

The **l3pdfmanagement** module

Managing central PDF resources

L^AT_EX PDF management testphase bundle

The L^AT_EX Project*

Version 0.95d, released 2021-05-14

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.

`\pdfmanagement_show:n` `\pdfmanagement_show:n` $\{\langle resource\ path\rangle\}$

New: 2020-04-08 This shows the content of the dictionary targetted by $\{\langle resource\ path\rangle\}$ 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.

`\pdfmanagement_remove:nn` `\pdfmanagement_remove:nn` $\{\langle resource\ path\rangle\}$ $\{\langle name\rangle\}$

New: 2020-04-07 Removes $\langle name\rangle$ and its associated $\langle value\rangle$ from the dictionary described with $\{\langle resource\ path\rangle\}$. The removal is global. If $\langle name\rangle$ 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.

`pdfmanagement: Info` `\pdfmanagement_add:nnn` $\{\text{Info}\}$ $\{\langle name\rangle\}$ $\{\langle value\rangle\}$

Adds $\langle name\rangle$ and the $\langle value\rangle$ to the Info dictionary. $\langle name\rangle$ 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. $\langle value\rangle$ 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. 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/pstopdf 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} {\langle name \rangle} {\langle value \rangle}

Adds /*\langle name \rangle* *\langle value \rangle* 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. *\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. Some backends expand the value but this should not be relied on. If a *\langle name \rangle* is used twice, only the last *\langle value \rangle* set will be used.

1.2.3 “Page” and “ThisPage”

pdfmanagement: Page \pdfmanagement_add:nnn {Page} {\langle name \rangle} {\langle value \rangle}

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*. *\langle name \rangle* should be a valid pdf name without the leading slash. Typical names used here are e.g. Rotate and CropBox. *\langle value \rangle* 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 *\langle name \rangle* is used twice, only the last *\langle value \rangle* 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 <code>/\langle name \rangle \langle value \rangle</code> at <i>shipout</i> to the page dictionary of the current page. Current page means here the <i>shipout</i> page. It is always stored globally. If <code>{\langle name \rangle}</code> has already a value set in the Page dictionary it will be overwritten for this page. <code>\langle name \rangle</code> should be a valid pdf name without the leading slash, <code>\langle value \rangle</code> should be a valid pdf value. Any escaping or (re)encoding must be done explicitly. If a <code>\langle name \rangle</code> is used twice, only the last <code>\langle value \rangle</code> set will be used. With the engine pdf _l atex (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 <code>\pdfmanagement_show:n</code> .
-----------------	---

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 `l3pdf file` 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	
	XFA	stream or array	pdf 1.5 probably unneeded
Catalog/AcroForm/DR	<i><name></i>		
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}{2021-05-14}{0.95d}
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.

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_lastpage_shipout_code_tl
84 {
85   \bool_if:NT \g__pdfmanagement_active_bool
86   {
87     \__pdf_backend_PageResources_obj_gpush: %ExtGState etc
88   }
89 }
90
91 \tl_gset:Nn \g__kernel_pdfmanagement_end_run_code_tl
92 {
93   \bool_if:NT \g__pdfmanagement_active_bool
94   {
95     \__pdfmanagement_Pages_gpush: %pagesattr
96     \__pdfmanagement_Info_gpush: %pdfinfo
97     \__pdfmanagement_Catalog_gpush:
98   }
99 }
```

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.

```

/Info % (\pdfinfo)
/Catalog % (\pdfcatalog)
/Catalog/AA %
/Catalog/AcroForm
/Catalog/OCProperties
/Catalog/OutputIntents
```

```

/Catalog/AcroForm/DR
/Catalog/AcroForm/DR/Font
/Catalog/MarkInfo
/Catalog/ViewerPreferences
/Pages % (\pagesattr)
/Page % (\pageattr)
/ThisPage % (\pageattr)
/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 simple cases it directly fill the property list, but if a handler exists this is called. It is important to use it only in places where this make sense.
 _pdfmanagement_get:nnN
 _pdfmanagement_gremove:nn
 _pdfmanagement_show:n

```

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

```

```

134 \cs_generate_variant:Nn \__pdfmanagement_handler_gput:nnn {nxx}
135
136 \cs_new_protected:Npn \__pdfmanagement_get:nnN #1 #2 #3 %path,key,macro
137 {
138   \exp_args:Nnx
139   \prop_get:cnN
140     { \__kernel_pdfdict_name:n { g__pdf_Core/#1 } }
141     { \str_convert_pdfname:n {#2} } #3
142 }
143
144
145 \cs_new_protected:Npn \__pdfmanagement_handler_gremove:nn #1 #2 %path,key
146 {
147   \pdfdict_if_exist:nTF { g__pdf_Core/#1 }
148     {
149       \cs_if_exist:cTF
150         { __pdfmanagement_handler/#1/?_gremove:n } %general, name independant handler
151         { \use:c {__pdfmanagement_handler/#1/?_gremove:n} {#2} }
152         {
153           \cs_if_exist:cTF
154             { __pdfmanagement_handler/#1/#2_gremove: }
155             { \use:c {__pdfmanagement_handler/#1/#2_gremove:} } %special handler
156             {
157               \exp_args:Nnx
158               \prop_gremove:cn
159                 { \__kernel_pdfdict_name:n { g__pdf_Core/#1 } }
160                 { \str_convert_pdfname:n {#2} }
161             }
162           }
163         }
164         {
165           \msg_error:nnn { pdfmanagement } { unknown-dict } { #1 }
166         }
167     }
168
169 \cs_new_protected:Npn \__pdfmanagement_gremove:nn #1 #2 %path,key
170 {
171   \pdfdict_if_exist:nTF { g__pdf_Core/#1 }
172     {
173       \exp_args:Nnx
174       \prop_gremove:cn
175         { \__kernel_pdfdict_name:n { g__pdf_Core/#1 } }
176         { \str_convert_pdfname:n{#2} }
177     }
178     {
179       \msg_error:nnn { pdfmanagement } { unknown-dict } { #1 }
180     }
181 }
182
183
184 \cs_new_protected:Npn \__pdfmanagement_show:Nn #1#2
185 {
186   \cs_if_exist:cTF
187     { __pdfmanagement_handler/#2/?_show: } %general, name independant handler

```

```

188 { \use:c {__pdfmanagement_handler/#2/?_show:} }
189 {
190   \prop_if_exist:cTF { \__kernel_pdffdict_name:n { g__pdf_Core/#2 } }
191   {
192     #1
193     { pdfmanagement } { show-dict }
194     { \tl_to_str:n {#2} }
195     {
196       \prop_map_function:cN
197       { \__kernel_pdffdict_name:n { g__pdf_Core/#2 } }
198       \msg_show_item:nn
199     }
200     { } { }
201   }
202   {
203     #1 { pdfmanagement } { unknown-dict } {#2}{-}{-}{-}
204   }
205 }
206 }
207
208 \cs_new_protected:Npn \__pdfmanagement_show:n #1 %path
209 {
210   \prop_show:c { \__kernel_pdffdict_name:n { g__pdf_Core/#1 } }
211 }

```

(End definition for __pdfmanagement_handler_gput:nnn and others.)

```

212 \cs_new_protected:Npn \pdfmanagement_show:n #1
213 {
214   \__pdfmanagement_show:Nn \msg_show:nnxxxx {#1}
215 }
216
217 \cs_new_protected:Npn \pdfmanagement_remove:nn #1 #2
218 {
219   \pdffdict_if_exist:nTF { g__pdf_Core/#1 }
220   {
221     \__pdfmanagement_handler_gremove:nn { #1 }{ #2 }
222   }
223   {
224     \msg_error:nnn{pdfmanagement}{unknown-dict}{#1}
225   }
226 }
227
228 \cs_new_protected:Npn \pdfmanagement_get:nnN #1 #2 #3
229 {
230   \pdffdict_if_exist:nTF { g__pdf_Core/#1 }
231   {
232     \__pdfmanagement_get:nnN { #1 }{ #2 } #3
233   }
234   {
235     \msg_error:nnn{pdfmanagement}{unknown-dict}{#1}
236   }
237 }

```

2.4 The Info dictionary

Initialization of the dictionary:

```
236 \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).

```
237 % push to the register command / issue the special
238 \cs_new_protected:Npn \__pdfmanagement_Info_gpush:
239 {
240   \prop_map_function:cN
241     { \__kernel_pdfdict_name:n { g__pdf_Core/Info} }
242   \__pdf_backend_info_gput:nn
243   \prop_gclear:c { \__kernel_pdfdict_name:n { g__pdf_Core/Info} }
244 }
```

(End definition for __pdfmanagement_Info_gpush:.)

2.5 The Pages dictionary code

At first the initialisation

```
245 \pdfdict_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`

```
246 % push to the register command / issue the special
247 \cs_new_protected:Npn \__pdfmanagement_Pages_gpush:
248 {
249   \exp_args:Nx \__pdf_backend_Pages_primitive:n
250   {
251     \pdfdict_use:n { g__pdf_Core/Pages}
252   }
253 }
254
```

(End definition for __pdfmanagement_Pages_gpush:.)

2.6 The Page and ThisPage dictionary

At first the initialisation.

```
255 \pdfdict_new:n { g__pdf_Core/Page }
256 \pdfdict_new:n { g__pdf_Core/ThisPage }
257
258 %handler for pdfmanagement
259 \cs_new_protected:cpn { __pdfmanagement_handler/Page/?_gput:nn } #1 #2
260 {
261   \__pdf_backend_Page_gput:nn { #1 }{ #2 }
262 }
263 % remove:
264 \cs_new_protected:cpn { __pdfmanagement_handler/Page/?_gremove:n } #1
265 {
266   \__pdf_backend_Page_gremove:n { #1 }
267 }
```

```

268
269 % handler for pdfmanagement
270 \cs_new_protected:cpn { __pdfmanagement_handler/ThisPage/?_gput:nn } #1 #2
271 {
272   \prop_gput:cnn { \__kernel_pdfdict_name:n { g__pdf_Core/ThisPage } }{ #1 } { #2 }
273   \bool_if:NT \g__pdfmanagement_active_bool
274   {
275     \__pdf_backend_ThisPage_gput:nn { #1 }{ #2 }
276   }
277 }
278
279 \cs_new_protected:cpn { __pdfmanagement_handler/ThisPage/?_gremove:n } #1
280 {
281   \msg_warning:nnn { pdfmanagement } { no-removal }{ThisPage}
282 }
283
284 \cs_new_protected:cpn { __pdfmanagement_handler/ThisPage/?_show: }
285 {
286   \msg_warning:nnn { pdfmanagement } { no-show }{ThisPage}
287 }
288

```

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

```

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

```

```

319 {
320   \__pdf_backend_PageResources_gput:nnm {Pattern} { #1 }{ #2 }
321 }

```

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

```

This variables hold the list of the various types of entries. With it the various `_gput` commands are generated.

(End definition for \c__pdfmanagement_Catalog_toplevel_clist, \c__pdfmanagement_Catalog_sub_clist, and \c__pdfmanagement_Catalog_seq_clist.)

```

\__pdfmanagement_catalog_XX_gput:n

```

Various commands to handle subentries and special cases.

```

322 \pdfdict_new:n { g__pdf_Core/Catalog}
323
324 \clist_const:Nn \c__pdfmanagement_Catalog_toplevel_clist
325 {
326   Collection,
327   DPartRoot,
328   Lang,
329   Legal,
330   Metadata,
331   NeedsRendering,
332   OCProperties/D,
333   OpenAction,
334   PageLabels,
335   PageLayout,
336   PageMode,
337   Perms,
338   PieceInfo,
339   SpiderInfo,
340   StructTreeRoot,
341   Threads,
342   URI,
343   Version
344 }
345
346 \clist_const:Nn \c__pdfmanagement_Catalog_sub_clist
347 {
348   AA,
349   AcroForm,
350   AcroForm/DR,
351   AcroForm/DR/Font,
352   MarkInfo,
353   ViewerPreferences,
354   OCProperties
355 }
356
357 \clist_map_inline:Nn \c__pdfmanagement_Catalog_sub_clist
358 {
359   \pdfdict_new:n { g__pdf_Core/Catalog/#1}
360 }
361

```



```

362
363 \clist_const:Nn \c__pdfmanagement_Catalog_seq_clist
364 {
365     AF,
366     OCPProperties/OCGs,
367     OCPProperties/Configs,
368     OutputIntents,
369     Requirements,
370     AcroForm/Fields,
371     AcroForm/CO
372 }
373
374
375
376 \clist_map_inline:Nn \c__pdfmanagement_Catalog_seq_clist
377 {
378     \seq_new:c { g__pdfmanagement_/Catalog/#1_seq } % new name later
379     \cs_new_protected:cpn { __pdfmanagement_handler/Catalog/#1_gput:n } ##1
380     {
381         \seq_gput_right:cn { g__pdfmanagement_/Catalog/#1_seq } { ##1 }
382     }
383 }
384
385 \cs_new_protected:cpn { __pdfmanagement_handler/Catalog/OCProperties/D_gput:n } #1
386 {
387     \seq_gput_left:cn
388     { g__pdfmanagement_/Catalog/OCProperties/Configs_seq }
389     { #1 }
390 }

```

(End definition for __pdfmanagement_catalog_XX_gput:n.)

Building the catalog: Push order

__pdfmanagement_Catalog_gpush:

```

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

```

```

410 }
(End definition for \__pdfmanagement_Catalog_gpush:.)

```

Building catalog entries: AA

__pdfmanagement_/Catalog/AA_gpush:

```

411 \cs_new_protected:cpn { __pdfmanagement_/Catalog/AA_gpush: }
412 {
413   \prop_if_empty:cF
414   { \__kernel_pdfdict_name:n { g__pdf_Core/Catalog/AA } }
415   {
416     \__pdf_backend_object_new:nn { __pdfmanagement_/Catalog/AA } { dict }
417     \__pdf_backend_object_write:nx
418       { __pdfmanagement_/Catalog/AA }
419       { \pdfdict_use:n { g__pdf_Core/Catalog/AA } }
420     \exp_args:Nnx
421       \__pdf_backend_catalog_gput:nn
422       {AA}
423       {
424         \__pdf_backend_object_ref:n { __pdfmanagement_/Catalog/AA }
425       }
426   }
427 }

```

(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:

```

428 \cs_new_protected:cpn { __pdfmanagement_/Catalog/AcroForm_gpush: }
429 {
430   \seq_if_empty:cF { g__pdfmanagement_/Catalog/AcroForm/Fields_seq }
431   {
432     \__pdf_backend_object_new:nn { __pdfmanagement_/Catalog/AcroForm/Fields } { array }
433     \__pdf_backend_object_write:nx
434       { __pdfmanagement_/Catalog/AcroForm/Fields }
435       { \seq_use:cn { g__pdfmanagement_/Catalog/AcroForm/Fields_seq } {~} }
436     \exp_args:Nnnx
437       \prop_gput:cn %we have to use \prop here to avoid the handler ...
438       { \__kernel_pdfdict_name:n { g__pdf_Core/Catalog/AcroForm } }
439       { Fields }
440       { \__pdf_backend_object_ref:n { __pdfmanagement_/Catalog/AcroForm/Fields } }
441   }
442   \seq_if_empty:cF { g__pdfmanagement_/Catalog/AcroForm/CO_seq }
443   {
444     \__pdf_backend_object_new:nn { __pdfmanagement_/Catalog/AcroForm/CO } { array }
445     \exp_args:Nnx
446       \__pdf_backend_object_write:nn
447       { __pdfmanagement_/Catalog/AcroForm/CO }
448       { \seq_use:cn { g__pdfmanagement_/Catalog/AcroForm/CO_seq } {~} }
449     \exp_args:Nnnx

```

```

450         \prop_gput:cnn %we have to use \prop here to avoid the handler ...
451         { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog/AcroForm } }
452         { CO }
453         { \__pdf_backend_object_ref:n { __pdfmanagement/Catalog/AcroForm/CO } }
454     }
455 \prop_if_empty:cF { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog/AcroForm/DR/Font}}
456 {
457     \__pdf_backend_object_new:nn { __pdfmanagement/Catalog/AcroForm/DR/Font } {dict}
458     \exp_args:Nnx
459     \__pdf_backend_object_write:nn
460     { __pdfmanagement/Catalog/AcroForm/DR/Font }
461     { \pdffdict_use:n { g__pdf_Core/Catalog/AcroForm/DR/Font } }
462     \exp_args:Nnnx
463     \prop_gput:cnn %we have to use \prop here to avoid the handler ...
464     { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog/AcroForm/DR } }
465     { Font }
466     { \__pdf_backend_object_ref:n { __pdfmanagement/Catalog/AcroForm/DR/Font } }
467 }
468 \prop_if_empty:cF { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog/AcroForm/DR}}
469 {
470     \__pdf_backend_object_new:nn { __pdfmanagement/Catalog/AcroForm/DR } {dict}
471     \exp_args:Nnx
472     \__pdf_backend_object_write:nn
473     { __pdfmanagement/Catalog/AcroForm/DR }
474     { \pdffdict_use:n { g__pdf_Core/Catalog/AcroForm/DR } }
475     \exp_args:Nnnx
476     \prop_gput:cnn %we have to use \prop here to avoid the handler ...
477     { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog/AcroForm } }
478     { DR }
479     { \__pdf_backend_object_ref:n { __pdfmanagement/Catalog/AcroForm/DR } }
480 }
481 \prop_if_empty:cF { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog/AcroForm} }
482 {
483     \__pdf_backend_object_new:nn { __pdfmanagement/Catalog/AcroForm } {dict}
484     \exp_args:Nnx
485     \__pdf_backend_object_write:nn
486     { __pdfmanagement/Catalog/AcroForm }
487     { \pdffdict_use:n { g__pdf_Core/Catalog/AcroForm } }
488     \exp_args:Nnnx
489     \__pdfmanagement_handler_gput:nnn
490     { Catalog }
491     { AcroForm }
492     { \__pdf_backend_object_ref:n { __pdfmanagement/Catalog/AcroForm } }
493 }
494 }
495

```

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

Building catalog entries: AF AF is an array.

__pdfmanagement_/Catalog/AF_gpush:

```

496 \cs_new_protected:cpn { __pdfmanagement_/Catalog/AF_gpush: }
497 {

```

```

498 \seq_if_empty:cF
499 { g__pdfmanagement_/Catalog/AF_seq }
500 {
501   \__pdf_backend_object_new:nn { __pdfmanagement_/Catalog/AF } { array }
502   \exp_args:Nnx
503   \__pdf_backend_object_write:nn
504     { __pdfmanagement_/Catalog/AF }
505     { \seq_use:cn { g__pdfmanagement_/Catalog/AF_seq } {~} }
506   \exp_args:Nnx
507   \__pdf_backend_catalog_gput:nn
508     {AF}
509     {
510       \__pdf_backend_object_ref:n {__pdfmanagement_/Catalog/AF}
511     }
512 }
513 }

```

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

Building catalog entries: MarkInfo

__pdfmanagement_/Catalog/MarkInfo_gpush:

```

514 \cs_new_protected:cpn { __pdfmanagement_/Catalog/MarkInfo_gpush: }
515 {
516   \prop_if_empty:cF
517   { \__kernel_pdfdict_name:n { g__pdf_Core/Catalog/MarkInfo } }
518   {
519     \__pdf_backend_object_new:nn { __pdfmanagement_/Catalog/MarkInfo } { dict }
520     \exp_args:Nnx
521     \__pdf_backend_object_write:nn
522       { __pdfmanagement_/Catalog/MarkInfo }
523       { \pdfdict_use:n { g__pdf_Core/Catalog/MarkInfo } }
524     \exp_args:Nnx
525     \__pdf_backend_catalog_gput:nn
526       {MarkInfo}
527       {
528         \__pdf_backend_object_ref:n {__pdfmanagement_/Catalog/MarkInfo}
529       }
530   }
531 }

```

(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:

```
532 % Catalog/OCProperties: OCGs + D is required
533 \cs_new_protected:cpn { __pdfmanagement_/Catalog/OCProperties_gpush: }
534 {
535   \int_compare:nNtT
536     {
537       ( \seq_count:c { g__pdfmanagement_/Catalog/OCProperties/OCGs_seq } ) *
538       ( \seq_count:c { g__pdfmanagement_/Catalog/OCProperties/Configs_seq } )
539     }
540     >
541     { 0 }
542     {
543       \__pdf_backend_object_new:nn { __pdfmanagement_/Catalog/OCProperties } { dict }
544       \seq_gpop_left:cN { g__pdfmanagement_/Catalog/OCProperties/Configs_seq } \l_tmpa_tl
545       \exp_args:Nnx
546         \__pdf_backend_object_write:nn { __pdfmanagement_/Catalog/OCProperties }
547         {
548           /OCGs~ [ \seq_use:cn { g__pdfmanagement_/Catalog/OCProperties/OCGs_seq } {~} ]
549           /D~ \l_tmpa_tl~
550           \seq_if_empty:cF { g__pdfmanagement_/Catalog/OCProperties/Configs_seq }
551           {
552             /Configs~
553             [ \seq_use:cn { g__pdfmanagement_/Catalog/OCProperties/Configs_seq } {~} ]
554           }
555         }
556       \exp_args:Nnx
557       \__pdf_backend_catalog_gput:nn
558       { OCProperties }
559       { \__pdf_backend_object_ref:n { __pdfmanagement_/Catalog/OCProperties } }
560     }
561   }
```

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

Building catalog entries: OutputIntents OutputIntents is an array.

pdfmanagement/Catalog/OutputIntents_gpush:

```
562 \cs_new_protected:cpn { __pdfmanagement_/Catalog/OutputIntents_gpush: }
563 {
564   \seq_if_empty:cF
565     { g__pdfmanagement_/Catalog/OutputIntents_seq }
566     {
567       \__pdf_backend_object_new:nn { __pdfmanagement_/Catalog/OutputIntents } { array }
568       \exp_args:Nnx
569       \__pdf_backend_object_write:nn
570       { __pdfmanagement_/Catalog/OutputIntents }
571       { \seq_use:cn { g__pdfmanagement_/Catalog/OutputIntents_seq } {~} }
572       \exp_args:Nnx
573       \__pdf_backend_catalog_gput:nn
574       { OutputIntents }
575       {
576         \__pdf_backend_object_ref:n { __pdfmanagement_/Catalog/OutputIntents }
577       }
578     }
```

```
579 }
```

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

Building catalog entries: Requirements Requirements is an array.

`__pdfmanagement_/Catalog/Requirements_gpush:`

```
580 \cs_new_protected:cpn { __pdfmanagement_/Catalog/Requirements_gpush: }
581 {
582   \seq_if_empty:cF
583   { g__pdfmanagement_/Catalog/Requirements_seq }
584   {
585     \__pdf_backend_object_new:nn { __pdfmanagement_/Catalog/Requirements } { array }
586     \exp_args:Nnx
587     \__pdf_backend_object_write:nn
588     { __pdfmanagement_/Catalog/Requirements }
589     { \seq_use:cn { g__pdfmanagement_/Catalog/Requirements_seq } {~} }
590     \exp_args:Nnx
591     \__pdf_backend_catalog_gput:nn
592     {Requirements}
593     {
594       \__pdf_backend_object_ref:n { __pdfmanagement_/Catalog/Requirements }
595     }
596   }
597 }
```

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

Building catalog entries: ViewerPreferences

`__pdfmanagement_/Catalog/ViewerPreferences_gpush:`

```
598 \cs_new_protected:cpn { __pdfmanagement_/Catalog/ViewerPreferences_gpush: }
599 {
600   \prop_if_empty:cF
601   { \__kernel_pdfdict_name:n { g__pdf_Core/Catalog/ViewerPreferences } }
602   {
603     \__pdf_backend_object_new:nn { __pdfmanagement_/Catalog/ViewerPreferences } { dict }
604     \exp_args:Nnx
605     \__pdf_backend_object_write:nn
606     { __pdfmanagement_/Catalog/ViewerPreferences }
607     { \pdfdict_use:n { g__pdf_Core/Catalog/ViewerPreferences } }
608     \exp_args:Nnx
609     \__pdf_backend_catalog_gput:nn
610     {ViewerPreferences}
611     {
612       \__pdf_backend_object_ref:n {__pdfmanagement_/Catalog/ViewerPreferences}
613     }
614   }
615 }
```

(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.

```
616 \pdfdict_new:n { g__pdf_Core/Catalog/Names }
617
618 \cs_new_protected:cpn { __pdfmanagement_handler/Catalog/Names/EmbeddedFiles_gput:n } #1
619 {
620   \__pdf_backend_NamesEmbeddedFiles_add:n { #1 }
621 }
```

(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:

```
622 %
623 \cs_new_protected:cpn { __pdfmanagement_/Catalog/Names/EmbeddedFiles_gpush: }
624 {
625   \seq_if_empty:NF \g__pdf_backend_EmbeddedFiles_seq
626   {
627     \exp_args:Nx \__pdf_backend_NamesEmbeddedFiles_gpush:n
628     {
629       \seq_use:Nn \g__pdf_backend_EmbeddedFiles_seq {~}
630     }
631   }
632 }
```

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

__pdfmanagement_handler/Catalog/?_show:

A handler to show the catalog.

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

```

658         \iow_term:x
659         {
660             The-Catalog-subdirectory~'##1'~contains-the~single-value~entries
661             \prop_map_function:cN
662             { \_kernel_pdfdict_name:n { g__pdf_Core/Catalog/##1 }}
663             \msg_show_item:nn
664         }
665     }
666 }
667 \tl_show:x {\tl_to_str:n{\pdfmanagement_show:n{Catalog}}}}
668 }
```

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

2.7 xform / Properties

```
669 \pdfdict_new:n { g__pdf_Core/Xform/Resources/Properties}
670 \</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	E
\\ 24, 33	exp commands:
	\exp_args:Nnnx 436, 449, 462, 475, 488
	\exp_args:Nnx 119, 138, 157, 173, 420, 445, 458, 471, 484, 502, 506, 520, 524, 545, 556, 568, 572, 586, 590, 604, 608
B	\exp_args:NV 78, 79
bool commands:	\exp_args:Nx 249, 627
\bool_if:NTF 39, 76, 85, 93, 273	
\bool_new:N 36	
C	
\clearpage 10	
clist commands:	H
\clist_const:Nn ... 289, 324, 346, 363	Handler 616
\clist_map_inline:Nn 297, 357, 376, 643, 654	hook commands:
cs commands:	\hook_gput_code:nnn 53
\cs_generate_variant:Nn 70, 134	\hook_new:n 46
\cs_if_exist:NTF 111, 115, 149, 153, 186	
\cs_new_protected:Npn 47, 102, 136, 145, 169, 184, 208, 212, 216, 226, 238, 247, 259, 264, 270, 279, 284, 303, 308, 313, 318, 379, 385, 391, 411, 428, 496, 514, 533, 562, 580, 598, 618, 623, 633	I
	int commands:
	\int_compare:nNnTF 535
	iow commands:
	\iow_newline: 637
	\iow_term:n 635, 647, 658
	K
	kernel internal commands:
	_kernel_pdfdict_name:n 121, 140, 159, 175, 190, 197, 210, 241, 243,
D	
\DeclareDocumentMetadata 9	

272, 406, 414, 438, 451, 455, 464, 468, 477, 481, 517, 601, 640, 656, 662	
\g__kernel_pdfmanagement_end_- run_code_tl 73, 91	
\g__kernel_pdfmanagement_- lastpage_shipout_code_tl .. 72, 83	
\g__kernel_pdfmanagement_- thispage_shipout_code_tl .. 71, 74	
\g__kernel_pdfmanagement_- thispage_shipout_code_- tl_uuuuuu\g__kernel_pdfmanagement_- lastpage_shipout_code_- tl_uuuuuu\g__kernel_pdfmanagement_- end_run_code_tl 71	
M	
msg commands:	
\msg_error:nnn 61, 128, 165, 179, 223, 233	
\msg_new:nnn .. 8, 11, 14, 17, 20, 27, 31	
\msg_none:nnn 106	
\msg_show:nnnnnn 214	
\msg_show_item:n 650	
\msg_show_item:nn 198, 641, 663	
\msg_warning:nnn 65, 281, 286	
N	
\newpage 4	
P	
pdf internal commands:	
__pdf_backend_catalog_gput:nn .. 407, 421, 507, 525, 557, 573, 591, 609	
\g__pdf_backend_EmbeddedFiles_- seq 625, 629	
\g__pdf_backend_end_run_tl 14	
__pdf_backend_info_gput:nn ... 242	
__pdf_backend_NamesEmbeddedFiles_- add:n 620	
__pdf_backend_NamesEmbeddedFiles_- gpush:n 627	
__pdf_backend_object_new:nn 416, 432, 444, 457, 470, 483, 501, 519, 543, 567, 585, 603	
__pdf_backend_object_ref:n 424, 440, 453, 466, 479, 492, 510, 528, 559, 576, 594, 612	
__pdf_backend_object_write:nn 417, 433, 446, 459, 472, 485, 503, 521, 546, 569, 587, 605	
__pdf_backend_Page_gput:nn ... 261	
__pdf_backend_Page_gremove:n .. 266	
__pdf_backend_PageResources_- gpush:n 79	
__pdf_backend_PageResources_- gput:nnn 305, 310, 315, 320	
__pdf_backend_PageResources_- obj_gpush: 87	
__pdf_backend_Pages_primitive:n 249	
__pdf_backend_ThisPage_gpush:n . 78	
__pdf_backend_ThisPage_gput:nn 275	
\pdfcatalog 1, 2	
pdfdict commands:	
\pdfdict_if_exist:nTF 51, 109, 147, 171, 218, 228	
\pdfdict_new:n 236, 245, 255, 256, 299, 322, 359, 616, 669	
\pdfdict_use:n 251, 419, 461, 474, 487, 523, 607	
\pdfinfo 1, 2	
pdfmanagement commands:	
pdfmanagement:Info 3	
pdfmanagement:Page 4	
pdfmanagement:Page/Resources/ColorSpace 5	
pdfmanagement:Page/Resources/ExtGState 5	
pdfmanagement:Page/Resources/Pattern 5	
pdfmanagement:Page/Resources/Shading 5	
pdfmanagement:Pages 4	
pdfmanagement:ThisPage 5	
\pdfmanagement_add:nnn 2-5, 47, 66, 70	
\pdfmanagement_get:nnN 226	
\pdfmanagement_if_active: 44	
\pdfmanagement_if_active:TF 2	
\pdfmanagement_if_active_p: 2	
\pdfmanagement_remove:nn .. 3, 4, 216	
\pdfmanagement_show:n . 3, 5, 212, 667	
pdfmanagement internal commands:	
__pdfmanagement_/Catalog/AA_- gpush: 411	
__pdfmanagement_/Catalog/AcroForm_- gpush: 428	
__pdfmanagement_/Catalog/AF_- gpush: 496	
__pdfmanagement_/Catalog/MarkInfo_- gpush: 514	
__pdfmanagement_/Catalog/Names/EmbeddedFiles_- gpush: 622	
__pdfmanagement_/Catalog/OCProperties_- gpush: 532	
__pdfmanagement_/Catalog/OutputIntents_- gpush: 562	
__pdfmanagement_/Catalog/Requirements_- gpush: 580	

[illegible]