

# The l3pdfmanagement package

## Managing central PDF resources

The L<sup>A</sup>T<sub>E</sub>X3 Project\*

Released XXXX-XX-XX

### 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 pdfLaTeX, LuaLaTeX, (x)dvipdfmx (LaTeX, XeLaTeX) 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<sup>A</sup>T<sub>E</sub>X 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 this 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.

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

---

\*E-mail: [latex-team@latex-project.org](mailto:latex-team@latex-project.org)

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}
```

---

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 dictionary<sup>1</sup>. 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.

---

```
\pdfmanagement_show:n \pdfmanagement_show:n {\resource path}
```

---

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.

---

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

---

<code>\pdfmanagement_remove:nn</code>	<code>\pdfmanagement_remove:nn {&lt;resource path&gt;} {&lt;name&gt;}</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} {&lt;name&gt;} {&lt;value&gt;}</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/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`.

---

<b>pdfmanagement:</b> <b>Pages</b>	<code>\pdfmanagement_add:nnn {Pages} {&lt;name&gt;} {&lt;value&gt;}</code>
------------------------------------	--

---

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”

---

<b>pdfmanagement:</b> <b>Page</b>	<code>\pdfmanagement_add:nnn {Page} {&lt;name&gt;} {&lt;value&gt;}</code>
-----------------------------------	---

---

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`.

---

<b>pdfmanagement:</b> <b>ThisPage</b>	<code>\pdfmanagement_add:nnn {ThisPage} {&lt;name&gt;} {&lt;value&gt;}</code>
---------------------------------------	---

---

New: 2020-04-12

Adds `/<name> <value>` at *shipout* to the page dictionary of the current page. Current page means here the *shipout* page. It is always stored globally. If `{<name>}` has already a value set in the `Page` dictionary it will be overwritten for this page. `<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. If a `<name>` is used twice, only the last `<value>` set will be used. With the engine pdf<sub>l</sub>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 `\pdfmanagement_show:n`.

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

---

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

---

Updated: 2020-04-10

---

Adds `/<name> <value>` to the page resource `<resource>`. `<resource>` 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. `<name>` should be a valid pdf name without the leading slash, `<value>` should be a valid pdf value for the resource. Any escaping or (re)encoding must be done explicitly. If a `<name>` is used twice, only the last `<value>` 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 <sup>2</sup>.



The primitive commands (e.g. `\pdfpageresources`) 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}{<Name>}{<Value>}
```

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.



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 `<Name>` 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.

---

<sup>2</sup>This is similar to how pgf handles this resources

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

Name	Value	Remark
Collection	objref or dict	the content should be build by external packages (see eg embedfile)
DPartRoot	objref or dict	PDF 2.0
Lang	string	e.g. (de-DE)
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 /SinglePage, /OneColumn, /TwoColumnLeft, /TwoColumnRight, /TwoPageLeft, /TwoPageRight
PageMode	name	one of /UseNone, /UseOutlines, /UseThumbs, /UseOC, /UseAttachments (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. /1.7
<i>&lt;unknown&gt;</i>		an unknown <i>&lt;name&gt;</i> 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 *<Name>* 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>&lt;name&gt;</i>	stream or array	pdf 1.5 probably unneeded
Catalog/AcroForm/DR/Font	<i>&lt;name&gt;</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 {pkg@intent}{dict}
\pdf_object_write:nn {pkg@intent}{...}
\pdfmanagement_add:nxx {Catalog} {OutputIntents}{\pdf_object_ref:n {pkg@intent}}
```

or

```
\pdf_object_now:nn {dict} { ... }
\pdfmanagement_add:nxx {Catalog} {OutputIntents}{\pdf_object_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 <*package>
3 %
4 \ProvidesExplPackage {l3pdfmanagement} {2020-04-08} {0.5}
5   {Main PDF Management}

```

### 2.1 Messages

```

6 \msg_new:nnn { pdfmanagement } { unknown-dict }
7             { The~PDF~management~resource~'#1'~is~unknown. }
8
9 \msg_new:nnn { pdfmanagement } { empty-value }
10            { The~value~for~#1~is~empty~and~will~be~ignored }
11
12 \msg_new:nnn { pdfmanagement } { no-removal }
13            { It~is~not~possible~to~remove~values~from~'#1'~.}
14
15 \msg_new:nnn { pdfmanagement } { no-show }
16            { It~is~not~possible~to~show~the~content~of~'#1'~.}
17
18 \msg_new:nnn { pdfmanagement } { show-dict }
19   {
20     The~PDF~resource~'#1'~
21     \tl_if_empty:nTF {#2}
22       { is~empty~\>~ . }
23       { contains~the~pairs~(without~outer~braces):~#2~ . }
24   }

```



```

25 \msg_new:nnn { pdfmanagement } { dict-already-defined }
26 {
27   The~path~'#1'~is~already~defined.
28 }
29 \msg_new:nnn { pdfmanagement } { inactive }
30 {
31   The~PDF~resources~management~is~not~active\\
32   command~'#1'~ignored.
33 }

```

`\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

```

34 \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

```

35 \prg_new_conditional:Npnn \__pdfmanagement_if_active: { p , T , F , TF }
36 {
37   \bool_if:NTF \g__pdfmanagement_active_bool
38   { \prg_return_true: }
39   { \prg_return_false: }
40 }
41 \prg_set_eq_conditional:NNn
42 \pdfmanagement_if_active: \__pdfmanagement_if_active: { p , T , F , TF }
43

```

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

```

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

```

## 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<sub>l</sub>atex/lualatex can execute code after the last `\clearpage` which the dvi-based drivers have to add on a shipout page.

pdf/management/end\_run  
pdf/management/lastpage\_shipout  
pdf/management/thispage\_shipout

This hooks contain the code run in the three places.

```

68 \hook_new:n {pdf/management/end_run}
69 \hook_new:n {pdf/management/lastpage_shipout}
70 \hook_new:n {pdf/management/thispage_shipout}

(End definition for pdf/management/end_run, pdf/management/lastpage_shipout, and pdf/management/thispage_
shipout. These variables are documented on page ??.)

71 \hook_gput_code:nnn {pdf/management/thispage_shipout} {pdf}
72 {
73   \bool_if:NT \g__pdfmanagement_active_bool
74   {
75     \exp_args:NV \__pdf_backend_ThisPage_gpush:n { \g_shipout_readonly_int }
76     \exp_args:NV \__pdf_backend_PageResources_gpush:n { \g_shipout_readonly_int }
77   }
78 }
79
80 \hook_gput_code:nnn {pdf/management/lastpage_shipout} {pdf}
81 {
82   \bool_if:NT \g__pdfmanagement_active_bool
83   {
84     \__pdf_backend_PageResources_obj_gpush:          %ExtGState etc
85   }
86 }
87
88 \hook_gput_code:nnn {pdf/management/end_run} {pdf}
89 {
90   \bool_if:NT \g__pdfmanagement_active_bool
91   {
92     \__pdfmanagement_Pages_gpush:                    %pagesattr
93     \__pdfmanagement_Info_gpush:                     %pdfinfo
94     \__pdfmanagement_Catalog_gpush:
95   }
96 }
```

## 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

```

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

```

```

131 \cs_generate_variant:Nn \__pdfmanagement_handler_gput:nnn {nxx}
132
133 \cs_new_protected:Npn \__pdfmanagement_get:nnN #1 #2 #3 %path,key,macro
134 {
135   \exp_args:Nnx
136   \prop_get:cnN
137   { \__kernel_pdfdict_name:n { g__pdf_Core/#1 } }
138   { \str_convert_pdfname:n {#2} } #3
139 }
140
141
142 \cs_new_protected:Npn \__pdfmanagement_handler_gremove:nn #1 #2 %path,key
143 {
144   \pdfdict_if_exist:nTF { g__pdf_Core/#1 }
145   {
146     \cs_if_exist:cTF
147     { __pdfmanagement_handler/#1/?_gremove:n } %general, name independant handler
148     { \use:c {__pdfmanagement_handler/#1/?_gremove:n} {#2} }
149     {
150       \cs_if_exist:cTF
151       { __pdfmanagement_handler/#1/#2_gremove: }
152       { \use:c {__pdfmanagement_handler/#1/#2_gremove:} } %special handler
153       {
154         \exp_args:Nnx
155         \prop_gremove:cn
156         { \__kernel_pdfdict_name:n { g__pdf_Core/#1 } } { \str_convert_pdfname:n {#2} }
157       }
158     }
159   }
160   {
161     \msg_error:nnn { pdfmanagement } { unknown-dict } { #1 }
162   }
163 }
164
165 \cs_new_protected:Npn \__pdfmanagement_gremove:nn #1 #2 %path,key
166 {
167   \pdfdict_if_exist:nTF { g__pdf_Core/#1 }
168   {
169     \exp_args:Nnx
170     \prop_gremove:cn
171     { \__kernel_pdfdict_name:n { g__pdf_Core/#1 } } { \str_convert_pdfname:n {#2} }
172   }
173   {
174     \msg_error:nnn { pdfmanagement } { unknown-dict } { #1 }
175   }
176 }
177
178
179 \cs_new_protected:Npn \__pdfmanagement_show:Nn #1#2
180 {
181   \cs_if_exist:cTF
182   { __pdfmanagement_handler/#2/?_show: } %general, name independant handler
183   { \use:c {__pdfmanagement_handler/#2/?_show:} }
184   {

```

```

185     \prop_if_exist:cTF { \__kernel_pdfdict_name:n { g__pdf_Core/#2 } }
186     {
187         #1
188         { pdfmanagement } { show-dict }
189         { \tl_to_str:n {#2} }
190         { \prop_map_function:cN {\__kernel_pdfdict_name:n { g__pdf_Core/#2 } } \msg_show:nn }
191         { } { }
192     }
193     {
194         #1 { pdfmanagement } { unknown-dict } {#2}{-}{-}{-}
195     }
196 }
197 }
198
199 \cs_new_protected:Npn \__pdfmanagement_show:n #1 %path
200 {
201     \prop_show:c { \__kernel_pdfdict_name:n { g__pdf_Core/#1 } }
202 }
203
204 (End definition for \__pdfmanagement_handler_gput:nnn and others.)
205
206 \cs_new_protected:Npn \pdfmanagement_show:n #1
207 {
208     \__pdfmanagement_show:Nn \msg_show:nnxxxx {#1}
209 }
210
211 \cs_new_protected:Npn \pdfmanagement_remove:nn #1 #2
212 {
213     \pdfdict_if_exist:nTF { g__pdf_Core/#1 }
214     {
215         \__pdfmanagement_handler_gremove:nn { #1 } { #2 }
216     }
217     {
218         \msg_error:nnn{pdfmanagement}{unknown-dict}{#1}
219     }
220 }
221
222 \cs_new_protected:Npn \pdfmanagement_get:nnN #1 #2 #3
223 {
224     \pdfdict_if_exist:nTF { g__pdf_Core/#1 }
225     {
226         \__pdfmanagement_get:nnN { #1 } { #2 } #3
227     }
228     {
229         \msg_error:nnn{pdfmanagement}{unknown-dict}{#1}
230     }
231 }

```

## 2.4 The Info dictionary

Initialization of the dictionary:

```

227 \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).

```

228 % push to the register command / issue the special
229 \cs_new_protected:Npn \__pdfmanagement_Info_gpush:
230 {
231   \prop_map_function:cN { \__kernel_pdffdict_name:n { g__pdf_Core/Info } } \__pdf_backend_in
232   \prop_gclear:c { \__kernel_pdffdict_name:n { g__pdf_Core/Info } }
233 }

```

(End definition for \\_\_pdfmanagement\_Info\_gpush:.)

## 2.5 The Pages dictionary code

At first the initialisation

```

234 \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

```

235 % push to the register command / issue the special
236 \cs_new_protected:Npn \__pdfmanagement_Pages_gpush:
237 {
238   \exp_args:Nx \__pdf_backend_Pages_primitive:n
239   {
240     \pdffdict_use:n { g__pdf_Core/Pages}
241   }
242 }
243

```

(End definition for \\_\_pdfmanagement\_Pages\_gpush:.)

## 2.6 The Page and ThisPage dictionary

At first the initialisation.

```

244 \pdffdict_new:n { g__pdf_Core/Page }
245 \pdffdict_new:n { g__pdf_Core/ThisPage }
246
247 %handler for pdfmanagement
248 \cs_new_protected:cpn { __pdfmanagement_handler/Page/?_gput:nn } #1 #2
249 {
250   \__pdf_backend_Page_gput:nn { #1 }{ #2 }
251 }
252 % remove:
253 \cs_new_protected:cpn { __pdfmanagement_handler/Page/?_gremove:n } #1
254 {
255   \__pdf_backend_Page_gremove:n { #1 }
256 }
257
258 % handler for pdfmanagement
259 \cs_new_protected:cpn { __pdfmanagement_handler/ThisPage/?_gput:nn } #1 #2
260 {
261   \prop_gput:cnn { \__kernel_pdffdict_name:n { g__pdf_Core/ThisPage } }{ #1 } { #2 }
262   \bool_if:NT \g__pdfmanagement_active_bool
263   {
264     \__pdf_backend_ThisPage_gput:nn { #1 }{ #2 }
265   }
266 }

```

```

267
268 \cs_new_protected:cpn { __pdfmanagement_handler/ThisPage/?_gremove:n } #1
269 {
270   \msg_warning:nnn { pdfmanagement } { no-removal }{ThisPage}
271 }
272
273 \cs_new_protected:cpn { __pdfmanagement_handler/ThisPage/?_show: }
274 {
275   \msg_warning:nnn { pdfmanagement } { no-show }{ThisPage}
276 }
277

```

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

```

278 \clist_const:Nn \c__pdfmanagement_PageResources_clist
279 {
280   ExtGState,
281   ColorSpace,
282   Pattern,
283   Shading,
284 }
285
286 \clist_map_inline:Nn \c__pdfmanagement_PageResources_clist
287 {
288   \pdfdict_new:n { g__pdf_Core/Page/Resources/#1}
289 }
290 %
291 % setter: #1 is the name of the resource
292 \cs_new_protected:cpn { __pdfmanagement_handler/Page/Resources/ExtGState/?_gput:nn } #1 #2
293 {
294   \__pdf_backend_PageResources_gput:nnn {ExtGState} { #1 }{ #2 }
295 }
296
297 \cs_new_protected:cpn { __pdfmanagement_handler/Page/Resources/ColorSpace/?_gput:nn } #1 #2
298 {
299   \__pdf_backend_PageResources_gput:nnn {ColorSpace} { #1 }{ #2 }
300 }
301
302 \cs_new_protected:cpn { __pdfmanagement_handler/Page/Resources/Shading/?_gput:nn } #1 #2
303 {
304   \__pdf_backend_PageResources_gput:nnn {Shading} { #1 }{ #2 }
305 }
306
307 \cs_new_protected:cpn { __pdfmanagement_handler/Page/Resources/Pattern/?_gput:nn } #1 #2
308 {
309   \__pdf_backend_PageResources_gput:nnn {Pattern} { #1 }{ #2 }
310 }

```

### 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.

```

311 \pdfdict_new:n { g__pdf_Core/Catalog}
312
313 \clist_const:Nn \c__pdfmanagement_Catalog_toplevel_clist
314 {
315     Collection,
316     DPartRoot,
317     Lang,
318     Legal,
319     Metadata,
320     NeedsRendering,
321     OCProperties/D,
322     OpenAction,
323     PageLabels,
324     PageLayout,
325     PageMode,
326     Perms,
327     PieceInfo,
328     SpiderInfo,
329     StructTreeRoot,
330     Threads,
331     URI,
332     Version
333 }
334
335 \clist_const:Nn \c__pdfmanagement_Catalog_sub_clist
336 {
337     AA,
338     AcroForm,
339     AcroForm/DR,
340     AcroForm/DR/Font,
341     MarkInfo,
342     ViewerPreferences,
343     OCProperties
344 }
345
346 \clist_map_inline:Nn \c__pdfmanagement_Catalog_sub_clist
347 {
348     \pdfdict_new:n { g__pdf_Core/Catalog/#1}
349 }
350
351
352 \clist_const:Nn \c__pdfmanagement_Catalog_seq_clist
353 {
354     AF,
355     OCProperties/OCGs,
356     OCProperties/Configs,
357     OutputIntents,
358     Requirements,
359     AcroForm/Fields,
360     AcroForm/CO

```



```

361 }
362
363
364
365 \clist_map_inline:Nn \c__pdfmanagement_Catalog_seq_clist
366 {
367   \seq_new:c { g__pdfmanagement_/Catalog/#1_seq } % new name later
368   \cs_new_protected:cpn { __pdfmanagement_handler/Catalog/#1_gput:n } ##1
369   {
370     \seq_gput_right:cn { g__pdfmanagement_/Catalog/#1_seq } { ##1 }
371   }
372 }
373
374 \cs_new_protected:cpn { __pdfmanagement_handler/Catalog/OCProperties/D_gput:n } #1
375 {
376   \seq_gput_left:cn
377   { g__pdfmanagement_/Catalog/OCProperties/Configs_seq }
378   { #1 }
379 }

```

(End definition for \\_\_pdfmanagement\_catalog\_XX\_gput:n.)

## Building the catalog: Push order

\\_\_pdfmanagement\_Catalog\_gpush:

```

380 \cs_new_protected:Npn \__pdfmanagement_Catalog_gpush:
381 {
382   \use:c { __pdfmanagement_/Catalog/AA_gpush: }
383   \use:c { __pdfmanagement_/Catalog/AcroForm_gpush: }
384   \use:c { __pdfmanagement_/Catalog/AF_gpush: }
385   \use:c { __pdfmanagement_/Catalog/MarkInfo_gpush: }
386   \use:c { __pdfmanagement_/Catalog/OCProperties_gpush: }
387   \use:c { __pdfmanagement_/Catalog/OutputIntents_gpush: }
388   \use:c { __pdfmanagement_/Catalog/Requirements_gpush: }
389   \use:c { __pdfmanagement_/Catalog/ViewerPreferences_gpush: }
390   % output the single values:
391   \prop_map_function:cN { \__kernel_pdfdict_name:n { g__pdf_Core/Catalog } } \__pdf_backend
392   % output names tree:
393   \use:c { __pdfmanagement_/Catalog/Names/EmbeddedFiles_gpush: }
394 }

```

(End definition for \\_\_pdfmanagement\_Catalog\_gpush:.)

## Building catalog entries: AA

\\_\_pdfmanagement\_/Catalog/AA\_gpush:

```

395 \cs_new_protected:cpn { __pdfmanagement_/Catalog/AA_gpush: }
396 {
397   \prop_if_empty:cF
398   { \__kernel_pdfdict_name:n { g__pdf_Core/Catalog/AA } }
399   {
400     \__pdf_backend_object_new:nn { g__pdfmanagement_/Catalog/AA_obj } { dict }
401     \__pdf_backend_object_write:nx
402     { g__pdfmanagement_/Catalog/AA_obj }

```

```

403         { \pdfdict_use:n { g__pdf_Core/Catalog/AA } }
404     \exp_args:Nnx
405     \__pdf_backend_catalog_gput:nn
406     {AA}
407     {
408         \__pdf_backend_object_ref:n { g__pdfmanagement_/Catalog/AA_obj }
409     }
410 }
411 }

```

(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:

```

412 \cs_new_protected:cpn { __pdfmanagement_/Catalog/AcroForm_gpush: }
413 {
414     \seq_if_empty:cF { g__pdfmanagement_/Catalog/AcroForm/Fields_seq }
415     {
416         \__pdf_backend_object_new:nn { g__pdfmanagement_/Catalog/AcroForm/Fields_obj } { array }
417         \__pdf_backend_object_write:nx
418         { g__pdfmanagement_/Catalog/AcroForm/Fields_obj }
419         { \seq_use:cn { g__pdfmanagement_/Catalog/AcroForm/Fields_seq } {-} }
420     \exp_args:Nnnx
421     \prop_gput:cnn %we have to use \prop here to avoid the handler ...
422     { \__kernel_pdfdict_name:n { g__pdf_Core/Catalog/AcroForm } }
423     { Fields }
424     { \__pdf_backend_object_ref:n { g__pdfmanagement_/Catalog/AcroForm/Fields_obj } }
425 }
426 \seq_if_empty:cF { g__pdfmanagement_/Catalog/AcroForm/CO_seq }
427 {
428     \__pdf_backend_object_new:nn { g__pdfmanagement_/Catalog/AcroForm/CO_obj } { array }
429     \exp_args:Nnx
430     \__pdf_backend_object_write:nn
431     { g__pdfmanagement_/Catalog/AcroForm/CO_obj }
432     { \seq_use:cn { g__pdfmanagement_/Catalog/AcroForm/CO_seq } {-} }
433     \exp_args:Nnnx
434     \prop_gput:cnn %we have to use \prop here to avoid the handler ...
435     { \__kernel_pdfdict_name:n { g__pdf_Core/Catalog/AcroForm } }
436     { CO }
437     { \__pdf_backend_object_ref:n { g__pdfmanagement_/Catalog/AcroForm/CO_obj } }
438 }
439 \prop_if_empty:cF { \__kernel_pdfdict_name:n { g__pdf_Core/Catalog/AcroForm/DR/Font}}
440 {
441     \__pdf_backend_object_new:nn { g__pdfmanagement_/Catalog/AcroForm/DR/Font_obj } { dict }
442     \exp_args:Nnx
443     \__pdf_backend_object_write:nn
444     { g__pdfmanagement_/Catalog/AcroForm/DR/Font_obj }
445     { \pdfdict_use:n { g__pdf_Core/Catalog/AcroForm/DR/Font } }
446     \exp_args:Nnnx
447     \prop_gput:cnn %we have to use \prop here to avoid the handler ...
448     { \__kernel_pdfdict_name:n { g__pdf_Core/Catalog/AcroForm/DR } }

```

```

449         { Font }
450         { \_pdf_backend_object_ref:n { g\_pdfmanagement_/Catalog/AcroForm/DR/Font_obj } }
451     }
452     \prop_if_empty:cF { \_kernel_pdffdict_name:n { g\_pdf_Core/Catalog/AcroForm/DR} }
453     {
454         \_pdf_backend_object_new:nn { g\_pdfmanagement_/Catalog/AcroForm/DR_obj } {dict}
455         \exp_args:Nnx
456         \_pdf_backend_object_write:nn
457             { g\_pdfmanagement_/Catalog/AcroForm/DR_obj }
458             { \pdffdict_use:n { g\_pdf_Core/Catalog/AcroForm/DR } }
459         \exp_args:Nnnx
460         \prop_gput:cn %we have to use \prop here to avoid the handler ...
461         { \_kernel_pdffdict_name:n { g\_pdf_Core/Catalog/AcroForm } }
462         { DR }
463         { \_pdf_backend_object_ref:n { g\_pdfmanagement_/Catalog/AcroForm/DR_obj } }
464     }
465     \prop_if_empty:cF { \_kernel_pdffdict_name:n { g\_pdf_Core/Catalog/AcroForm } }
466     {
467         \_pdf_backend_object_new:nn { g\_pdfmanagement_/Catalog/AcroForm_obj } {dict}
468         \exp_args:Nnx
469         \_pdf_backend_object_write:nn
470             { g\_pdfmanagement_/Catalog/AcroForm_obj }
471             { \pdffdict_use:n { g\_pdf_Core/Catalog/AcroForm } }
472         \exp_args:Nnnx
473         \_pdfmanagement_handler_gput:nnn
474             { Catalog }
475             { AcroForm }
476             { \_pdf_backend_object_ref:n { g\_pdfmanagement_/Catalog/AcroForm_obj } }
477     }
478 }
479

```

(End definition for \\_pdfmanagement\_/Catalog/AcroForm\_gpush:.)

**Building catalog entries: AF** AF is an array.

\\_pdfmanagement\_/Catalog/AF\_gpush:

```

480 \cs_new_protected:cpn { \_pdfmanagement_/Catalog/AF_gpush: }
481 {
482     \seq_if_empty:cF
483     { g\_pdfmanagement_/Catalog/AF_seq }
484     {
485         \_pdf_backend_object_new:nn { g\_pdfmanagement_/Catalog/AF_obj } { array }
486         \exp_args:Nnx
487         \_pdf_backend_object_write:nn
488             { g\_pdfmanagement_/Catalog/AF_obj }
489             { \seq_use:cn { g\_pdfmanagement_/Catalog/AF_seq } {~} }
490         \exp_args:Nnx
491         \_pdf_backend_catalog_gput:nn
492             {AF}
493             {
494                 \_pdf_backend_object_ref:n {g\_pdfmanagement_/Catalog/AF_obj}
495             }
496     }

```

```
497 }
```

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

## Building catalog entries: MarkInfo

`__pdfmanagement_/Catalog/MarkInfo_gpush:`

```
498 \cs_new_protected:cpn { __pdfmanagement_/Catalog/MarkInfo_gpush: }
499 {
500   \prop_if_empty:cF
501   { \__kernel_pdfdict_name:n { g__pdf_Core/Catalog/MarkInfo } }
502   {
503     \__pdf_backend_object_new:nn { g__pdfmanagement_/Catalog/MarkInfo_obj } { dict }
504     \exp_args:Nnx
505     \__pdf_backend_object_write:nn
506     { g__pdfmanagement_/Catalog/MarkInfo_obj }
507     { \pdfdict_use:n { g__pdf_Core/Catalog/MarkInfo } }
508     \exp_args:Nnx
509     \__pdf_backend_catalog_gput:nn
510     {MarkInfo}
511     {
512       \__pdf_backend_object_ref:n {g__pdfmanagement_/Catalog/MarkInfo_obj}
513     }
514   }
515 }
```

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

**Building catalog entries: OCPProperties** 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/OCPProperties_gpush:`

```
516 % Catalog/OCPProperties: OCGs + D is required
517 \cs_new_protected:cpn { __pdfmanagement_/Catalog/OCPProperties_gpush: }
518 {
519   \int_compare:nNnT
520   {
521     ( \seq_count:c { g__pdfmanagement_/Catalog/OCPProperties/OCGs_seq } ) *
522     ( \seq_count:c { g__pdfmanagement_/Catalog/OCPProperties/Configs_seq } )
523   }
524   >
525   { 0 }
526   {
527     \__pdf_backend_object_new:nn { g__pdfmanagement_/Catalog/OCPProperties_obj } { dict }
528     \seq_gpop_left:cN { g__pdfmanagement_/Catalog/OCPProperties/Configs_seq } \l_tmpa_tl
529     \exp_args:Nnx
```

```

530     \__pdf_backend_object_write:nn {g__pdfmanagement_/Catalog/OCProperties_obj}
531     {
532         /OCGs~[ \seq_use:cn { g__pdfmanagement_/Catalog/OCProperties/OCGs_seq } {-} ]
533         /D~\l_tmpa_tl~
534         \seq_if_empty:cF { g__pdfmanagement_/Catalog/OCProperties/Configs_seq }
535         {
536             /Configs~
537             [ \seq_use:cn { g__pdfmanagement_/Catalog/OCProperties/Configs_seq } {-} ]
538         }
539     }
540     \exp_args:Nnx
541     \__pdf_backend_catalog_gput:nn
542     { OCProperties }
543     { \__pdf_backend_object_ref:n {g__pdfmanagement_/Catalog/OCProperties_obj} }
544 }
545 }

```

(End definition for \\_\_pdfmanagement\_/Catalog/OCProperties\_gpush:.)

**Building catalog entries: OutputIntents** OutputIntents is an array.

\_pdfmanagement\_/Catalog/OutputIntents\_gpush:

```

546 \cs_new_protected:cpn { __pdfmanagement_/Catalog/OutputIntents_gpush: }
547 {
548     \seq_if_empty:cF
549     { g__pdfmanagement_/Catalog/OutputIntents_seq }
550     {
551         \__pdf_backend_object_new:nn { g__pdfmanagement_/Catalog/OutputIntents_obj } { array }
552         \exp_args:Nnx
553         \__pdf_backend_object_write:nn
554         { g__pdfmanagement_/Catalog/OutputIntents_obj }
555         { \seq_use:cn { g__pdfmanagement_/Catalog/OutputIntents_seq } {-} }
556         \exp_args:Nnx
557         \__pdf_backend_catalog_gput:nn
558         {OutputIntents}
559         {
560             \__pdf_backend_object_ref:n {g__pdfmanagement_/Catalog/OutputIntents_obj}
561         }
562     }
563 }

```

(End definition for \\_\_pdfmanagement\_/Catalog/OutputIntents\_gpush:.)

**Building catalog entries: Requirements** Requirements is an array.

\_pdfmanagement\_/Catalog/Requirements\_gpush:

```

564 \cs_new_protected:cpn { __pdfmanagement_/Catalog/Requirements_gpush: }
565 {
566     \seq_if_empty:cF
567     { g__pdfmanagement_/Catalog/Requirements_seq }
568     {
569         \__pdf_backend_object_new:nn { g__pdfmanagement_/Catalog/Requirements_obj } { array }
570         \exp_args:Nnx
571         \__pdf_backend_object_write:nn

```

```

572         { g__pdfmanagement_/Catalog/Requirements_obj }
573         { \seq_use:cn { g__pdfmanagement_/Catalog/Requirements_seq } {-} }
574     \exp_args:Nnx
575     \__pdf_backend_catalog_gput:nn
576     {Requirements}
577     {
578         \__pdf_backend_object_ref:n { g__pdfmanagement_/Catalog/Requirements_obj }
579     }
580 }
581 }

```

(End definition for \\_\_pdfmanagement\_/Catalog/Requirements\_gpush:.)

## Building catalog entries: ViewerPreferences

management\_/Catalog/ViewerPreferences\_gpush:

```

582 \cs_new_protected:cpn { __pdfmanagement_/Catalog/ViewerPreferences_gpush: }
583 {
584     \prop_if_empty:cF
585     { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog/ViewerPreferences } }
586     {
587         \__pdf_backend_object_new:nn { g__pdfmanagement_/Catalog/ViewerPreferences_obj } { di
588         \exp_args:Nnx
589         \__pdf_backend_object_write:nn
590         { g__pdfmanagement_/Catalog/ViewerPreferences_obj }
591         { \pdffdict_use:n { g__pdf_Core/Catalog/ViewerPreferences } }
592         \exp_args:Nnx
593         \__pdf_backend_catalog_gput:nn
594         {ViewerPreferences}
595         {
596             \__pdf_backend_object_ref:n {g__pdfmanagement_/Catalog/ViewerPreferences_obj}
597         }
598     }
599 }

```

(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.

```

600 \pdffdict_new:n { g__pdf_Core/Catalog/Names }
601
602 \cs_new_protected:cpn { __pdfmanagement_handler/Catalog/Names/EmbeddedFiles_gput:n } #1
603 {
604     \__pdf_backend_NamesEmbeddedFiles_add:n { #1 }
605 }

```

(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:

```

606 %
607 \cs_new_protected:cpn { __pdfmanagement_/Catalog/Names/EmbeddedFiles_gpush: }
608 {

```

```

609 \seq_if_empty:NF \g__pdf_backend_EmbeddedFiles_seq
610 {
611     \exp_args:Nx \__pdf_backend_NamesEmbeddedFiles_gpush:n
612     {
613         \seq_use:Nn \g__pdf_backend_EmbeddedFiles_seq {~}
614     }
615 }
616 }

```

(End definition for \\_\_pdfmanagement\_/Catalog/Names/EmbeddedFiles\_gpush:.)

\_\_pdfmanagement\_handler/Catalog/?\_show: A handler to show the catalog.

```

617 \cs_new_protected:cpn {__pdfmanagement_handler/Catalog/?_show:}
618 {
619     \iow_term:x
620     {
621         \iow_newline:
622         The~Catalog~contains~in~the~top~level~the~single~value~entries
623         \prop_map_function:cN {\__kernel_pdffdict_name:n { g__pdf_Core/Catalog }} \msg_show_it
624     }
625     \clist_map_inline:Nn \c__pdfmanagement_Catalog_seq_clist
626     {
627         \seq_if_empty:cF { g__pdfmanagement_/Catalog/##1_seq }
628         {
629             \iow_term:x
630             {
631                 The~'##1'~array~contains~the~entries
632                 \seq_map_function:cN { g__pdfmanagement_/Catalog/AF_seq } \msg_show_item:n
633             }
634         }
635     }
636     \clist_map_inline:Nn \c__pdfmanagement_Catalog_sub_clist
637     {
638         \prop_if_empty:cF { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog/##1 } }
639         {
640             \iow_term:x
641             {
642                 The~Catalog~subdirectory~'##1'~contains~the~single~value~entries
643                 \prop_map_function:cN {\__kernel_pdffdict_name:n { g__pdf_Core/Catalog/##1 }}
644             }
645         }
646     }
647     \tl_show:x {\tl_to_str:n{\pdfmanagement_show:n{Catalog}}}
648 }

```

(End definition for \_\_pdfmanagement\_handler/Catalog/?\_show:.)

## 2.7 xform / Properties

```

649 \pdffdict_new:n { g__pdf_Core/Xform/Resources/Properties}
650 </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		K	
\\	22, 31	kernel internal commands:	
		\\_kernel_pdffdict_name:n	118, 137, 156, 171, 185, 190, 201, 231, 232, 261, 391, 398, 422, 435, 439, 448, 452, 461, 465, 501, 585, 623, 638, 643
B		M	
bool commands:		msg commands:	
\\bool_if:NTF	37, 73, 82, 90, 262	\\msg_error:nnn	59, 125, 161, 174, 214, 224
\\bool_new:N	34	\\msg_new:nnn	6, 9, 12, 15, 18, 25, 29
C		\\msg_none:nnn	103
\\clearpage	10	\\msg_show:nnnnnn	205
clist commands:		\\msg_show_item:n	632
\\clist_const:Nn	278, 313, 335, 352	\\msg_show_item:nn	190, 623, 643
\\clist_map_inline:Nn	286, 346, 365, 625, 636	\\msg_warning:nnn	63, 270, 275
cs commands:			
\\cs_generate_variant:Nn	67, 131	N	
\\cs_if_exist:NTF	108, 112, 146, 150, 181	\\newpage	4
\\cs_new_protected:Npn	45, 99, 133, 142, 165, 179, 199, 203, 207, 217, 229, 236, 248, 253, 259, 268, 273, 292, 297, 302, 307, 368, 374, 380, 395, 412, 480, 498, 517, 546, 564, 582, 602, 607, 617	P	
D		pdf internal commands:	
\\DeclareDocumentMetadata	9	\\__pdf_backend_catalog_gput:nn	391, 405, 491, 509, 541, 557, 575, 593
E		\\g__pdf_backend_EmbeddedFiles_-seq	609, 613
exp commands:		\\g__pdf_backend_end_run_tl	14
\\exp_args:Nnnx	420, 433, 446, 459, 472	\\__pdf_backend_info_gput:nn	231
\\exp_args:Nnx	116, 135, 154, 169, 404, 429, 442, 455, 468, 486, 490, 504, 508, 529, 540, 552, 556, 570, 574, 588, 592	\\__pdf_backend_NamesEmbeddedFiles_-add:n	604
\\exp_args:Nv	75, 76	\\__pdf_backend_NamesEmbeddedFiles_-gpush:n	611
\\exp_args:Nx	238, 611	\\__pdf_backend_object_new:nn	400, 416, 428, 441, 454, 467, 485, 503, 527, 551, 569, 587
H		\\__pdf_backend_object_ref:n	408, 424, 437, 450, 463, 476, 494, 512, 543, 560, 578, 596
Handler	600	\\__pdf_backend_object_write:nn	401, 417, 430, 443, 456, 469, 487, 505, 530, 553, 571, 589
hook commands:		\\__pdf_backend_Page_gput:nn	250
\\hook_gput_code:nnn	51, 71, 80, 88	\\__pdf_backend_Page_gremove:n	255
\\hook_new:n	44, 68, 69, 70	\\__pdf_backend_PageResources_-gpush:n	76
I		\\__pdf_backend_PageResources_-gput:nnn	294, 299, 304, 309
int commands:			
\\int_compare:nNnTF	519		
iow commands:			
\\iow_newline:	621		
\\iow_term:n	619, 629, 640		



\_pdf_backend_PageResources_- obj_gpush: . . . . .	84	\_pdfmanagement_/Catalog/OutputIntents_- gpush: . . . . .	546
\_pdf_backend_Pages_primitive:n	238	\_pdfmanagement_/Catalog/Requirements_- gpush: . . . . .	564
\_pdf_backend_ThisPage_gpush:n	75	\_pdfmanagement_/Catalog/ViewerPreferences_- gpush: . . . . .	582
\_pdf_backend_ThisPage_gput:nn	264	\_pdfmanagement_active_bool . . . . .	34, 37, 73, 82, 90, 262
pdf/management/end commands:		\_pdfmanagement_Catalog_gpush: . . . . .	94, 380, 380
pdf/management/end_run . . . . .	68	\_c\_pdfmanagement_Catalog_seq_- clist . . . . .	311, 352, 365, 625
pdf/management/lastpage commands:		\_c\_pdfmanagement_Catalog_sub_- clist . . . . .	311, 335, 346, 636
pdf/management/lastpage_shipout .	68	\_c\_pdfmanagement_Catalog_- toplevel_clist . . . . .	311, 313
pdf/management/thispage commands:		\_pdfmanagement_catalog_XX_- gput:n . . . . .	311
pdf/management/thispage_shipout .	68	\_pdfmanagement_get:nnN	97, 133, 221
\pdfcatalog . . . . .	1, 2	\_pdfmanagement_gremove:nn	97, 165
pdfdict commands:		\_pdfmanagement_handler/Catalog/?_- show: . . . . .	617
\pdfdict_if_exist:nTF . . . . .	49, 106, 144, 167, 209, 219	\_pdfmanagement_handler_- gput:nnn . . . . .	11, 55, 97, 99, 131, 473
\pdfdict_new:n . . . . .	227, 234, 244, 245, 288, 311, 348, 600, 649	\_pdfmanagement_handler_- gremove:nn . . . . .	142, 211
\pdfdict_use:n . . . . .	240, 403, 445, 458, 471, 507, 591	\_pdfmanagement_if_active: .	35, 42
\pdfinfo . . . . .	1, 2	\_pdfmanagement_if_active:TF . . . . .	47
pdfmanagement commands:		\_pdfmanagement_Info_gpush: . . . . .	13, 93, 228, 229
pdfmanagement:Info . . . . .	3	\_c\_pdfmanagement_PageResources_- clist . . . . .	278, 286
pdfmanagement:Page . . . . .	4	\_pdfmanagement_Pages_gpush: . . . . .	92, 235, 236
pdfmanagement:Page/Resources/ColorSpace . . . . .	5	\_pdfmanagement_show:n . . . . .	97, 199
pdfmanagement:Page/Resources/ExtGState . . . . .	5	\_pdfmanagement_show:Nn . . . . .	179, 205
pdfmanagement:Page/Resources/Pattern . . . . .	5	\pdfpageattr . . . . .	1, 2, 4
pdfmanagement:Page/Resources/Shading . . . . .	5	\pdfpageresources . . . . .	1, 2, 5
pdfmanagement:Pages . . . . .	4	\pdfpagesattr . . . . .	1, 2, 4
pdfmanagement:ThisPage . . . . .	4	prg commands:	
\pdfmanagement_add:nnn . . . . .	2, 3, 4, 4, 4, 5, 45, 63, 67	\prg_new_conditional:Npnn . . . . .	35
\pdfmanagement_get:nnN . . . . .	217	\prg_return_false: . . . . .	39
\pdfmanagement_if_active: . . . . .	42	\prg_return_true: . . . . .	38
\pdfmanagement_if_active:TF . . . . .	2	\prg_set_eq_conditional:NNn . . . . .	41
\pdfmanagement_if_active_p: . . . . .	2	\prop . . . . .	421, 434, 447, 460
\pdfmanagement_remove:nn . . . . .	3, 4, 207	prop commands:	
\pdfmanagement_show:n . . . . .	2, 4, 203, 647	\prop_gclear:N . . . . .	232
pdfmanagement internal commands:		\prop_get:NnN . . . . .	136
\_pdfmanagement_/Catalog/AA_- gpush: . . . . .	395	\prop_gput:Nnn . . . . .	117, 261, 421, 434, 447, 460
\_pdfmanagement_/Catalog/AcroForm_- gpush: . . . . .	412	\prop_gremove:Nn . . . . .	155, 170
\_pdfmanagement_/Catalog/AF_- gpush: . . . . .	480	\prop_if_empty:NTF . . . . .	397, 439, 452, 465, 500, 584, 638
\_pdfmanagement_/Catalog/MarkInfo_- gpush: . . . . .	498	\prop_if_exist:NTF . . . . .	185
\_pdfmanagement_/Catalog/Names/EmbeddedFiles_- gpush: . . . . .	606		
\_pdfmanagement_/Catalog/OCProperties_- gpush: . . . . .	516		

