

The **l3pdf**field module

Commands to create form fields

L^AT_EX PDF management testphase bundle

The L^AT_EX Project*

Version 0.95d, released 2021-05-14

1 **l3pdf**field Introduction

The implementation of form fields in hyperref has some bugs¹. This package is a first step towards the goal to review and improve the code of form fields.

Like the `pdfmanagement-testphase` package itself it is a temporary package: the definite home of the code is not yet decided, and during the development changes in the interfaces are possible.

The package itself is currently loaded with

```
\usepackage{l3pdf
```

```
field-testphase}
```

The code is splitted into various submodules. `l3pdf`field contains the basic commands to create a form field. The code related to field types like checkboxes are in `l3pdf`field-type, for example `l3pdf`field-checkbox. Currently only checkboxes have been implemented, other form fields like pushbutton, radio buttons or text fields will follow later. The code doesn't rely on to initialize the form, but it can be used with hyperref.

The code requires the new PDF management. The code makes use of `l3pdf`xform to create the form Xobjects of the appearances. This code doesn't support yet the the dvips backend.

The code targets PDF 2.0. This doesn't mean that it won't work in older PDF versions, but it tries to implement requirements needed or recommended for 2.0; most importantly appearances are used by default everywhere and it deprecates `/NeedAppearances`.

Please keep in mind

- Not every PDF viewer supports form fields or all types and features.
- The handling can depend on settings in the PDF viewer. In adobe reader for example I had to disable an option to avoid that it tries to create an appearance itself.
- Standards like pdf/A disable some features of form fields like javascript actions (as you typically can't change the PDF).

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

¹see for example <https://github.com/latex3/hyperref/issues/94>

If `hyperref` is loaded before the package will suppress the deprecated `/NeedAppearances` setting. If `hyperref` is loaded later you should do it in the `\Form` options.

So a typical use together with `hyperref` could look like this

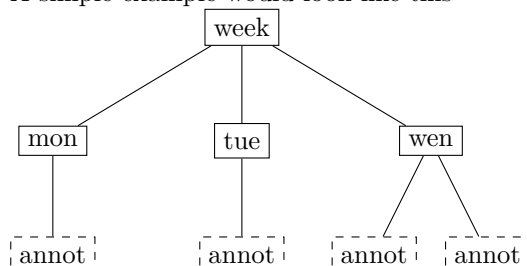
```
\RequirePackage{pdfmanagement-testphase}
\DeclareDocumentMetadata{uncompress}
\documentclass{article}
\usepackage{hyperref}
\usepackage{l3pdffield-testphase}
\begin{document}
\Form
```

2 Some background

A document can contain an arbitrary number of fields which can be organized in trees. The leaf fields in such a tree, the *terminal fields*, typically have widget annotations as kids which are then the actual, visual instances of the field, and allow to interact with the field. I will call such a tree a *fieldset*, nodes *fields* and the widget annotation *field annotations*.

If a field has only one child annotation the content of the field dictionary and the widget annotation dictionary can be merged—some examples in the PDF reference show such merged dictionaries—but the code here keeps them separate, at the end this is clearer.

A simple example would look like this



In many cases a fieldset consists of only one field along with its field annotation(s), but larger sets can be needed to build more complex interactions with javascript code. For example a datepicker can be built as a fieldset with various fields to represent the month and year choice and to select days.

Fields in a fieldset should have a name, for example `wen` or `week` in the example above. This name is the *partial name* of the field, the *full name* is then built from it by adding the names of the parents separated by periods. In the example above the partial name is `mon` and the full name `week.mon`. Partial names shouldn't contain periods. If two fields have the same name they will work in unison: if you enter text in one field, the text appears also in the other, such fields must have the same type and the same value and default value entry. If a field has no name it is considered to be a simple widget annotation and so only another representation of its parent.

All terminal fields should also have a type, e.g. `Btn` for a button field, or `Tx` for a textfield. The type can be set for the parent and then inherited. The fields in a fieldset can have different types.

2.1 The look of a field: Appearances and other settings

The look of widget annotation of a field can be set with various keys. The keys developed over time and some of them supersede older ones. There is for example the simple `/Border`, the more sophisticated `/BS` (“border style dictionary”), the “dynamic appearance dictionary” `MK`, with lots of keys, and the appearance dictionary `/AP` which may define as many as three separate appearances: the normal appearance (required), the rollover appearance and the down appearance. Such an appearance can be a simple form XObjects², but in some cases the annotation can have different *appearance states*: a checkbox for example can be checked or unchecked, in this case the appearances are dictionaries which map state names like `/Yes` and `/Off` to form XObjects.

The annotations cover a rectangular area on the page and form XObjects appearances are squeezed into this rectangle. So for the best result both should have the same ratio of width and height. Simple plain backgrounds can also be created in large size and reused for various annotations. Form XObjects used as appearances can not be rotated, if needed one has to create a new appearance.

In PDF 2.0 widget annotations must have at least a normal `/AP` appearance (unless the size of the annotation is zero) and the keys “*C, IC, Border, BS, BE, BM, CA, ca, H, DA, Q, DS, LE, LL, LLE, and Sy shall be ignored*”. But it is quite unclear if PDF Viewer honor this, and if this make sense e.g. for text fields which require a `DA` entry. It is also not clear how appearances and the entries of the `MK` dictionary are related in a form field. Tests with some PDF viewers are needed here.

3 Commands

```
\pdffield_field:nn \pdffield_field:nn{<key val list>}{<field ID>}
```

```
\pdffield_field:Vn
```

This creates a new field. `<field ID>` will be used to create and reference the needed objects but it is not the direct object name, so `pdf_object_ref:n` can not be used to access (and there will not clash with object names). It is recommended to start the name with a module prefix to avoid name clashes, so e.g. `mymodule/field/1` or `mymodule/field/week`.

The list of handled keys is described below. Typically the `<key val list>` should at least set the name `T`, fields that are kids in a fieldset must set the `parent` key, this should point to a field declared before.

The command is meant as a basic command to build more complex variants like checkbox or textfields. For this reason it doesn’t check if the combination of values and flags are sensible, and it uses as key names the names from the PDF reference. If you create a button field (`Btn`) and set `MaxLen` (which is only known for text fields), it will not complain.

Root fields (fields without parent) are added automatically to the `Catalog/AcroForm` dictionary with

```
\pdfmanagement_add:nnx{Catalog/AcroForm}{Fields}{<obj ref>}
```

²Such form XObjects are small pictures stored in the PDF which can be referenced in various part of the PDF. They can be created with the commands of the `l3pdfxform` package.

<hr/> <hr/>	<code>\pdffield_annot:n</code>	<code>\pdffield_field:nn{<key val list>}</code>
<hr/>	<code>\pdffield_annot:V</code>	This creates a new field annotation. It is a widget annotation box created with <code>\pdfannot_widget_box:nnn</code> , and it is possible to add values to its dictionary by using <code>\pdfannot_dict_put:nnn {widget}...</code> . But to correctly setup the parent/kid relationship some additional wrapper code is needed. The command also setup dictionaries to fill the AP, MK and AA dictionaries.
<hr/> <hr/>	<code>\pdffield_appearance:n</code>	<code>\pdffield_appearance:nn{<name>}{<content>}</code>
<hr/>		This is a small wrapper around <code>\pdfxform_new:nnn</code> (which could be used too) to create an appearance. To avoid name clashes <code><name></code> should start with a module part, e.g. <code>mymodule/appearance/cross</code> .
<hr/> <hr/>	<code>\pdffield_setup:n</code>	<code>\pdffield_setup:n{<key-val>}</code>
<hr/>		This command allows to preset some field settings. It knows currently two keys:
<hr/> <hr/>	<code>create-style</code>	<code>create-style = {<name>}{<key-val>}</code>
<hr/>		This defines a style which can then be used with the <code>style</code> key. <code>{<key-val>}</code> can be an arbitrary collection of the keys of the module.
<hr/> <hr/>	<code>preset-checkbox</code>	<code>preset-checkbox={<key-val>}</code>
<hr/>		This allows to set default keys for a checkbox.

4 Field Keys

Table 1 summarize the keys which can be used. A number of keys have two names, the second is normally the name used by hyperref. Where it makes sense an empty value “unsets” a key.

<hr/>	<code>parent</code>	<code>parent = <field ID></code>
<hr/>		This declares the parent of the field. It is required if the field is not the root of the fieldset. The value is the field ID of the parent, the parent should have been already declared. It will add the reference to the parent field to the <code>/Parent</code> key, and also add reference of the kid as <code>/Kid</code> in the parent field.
<hr/>	<code>name</code>	<code>name = <partial name></code>
<hr/>	<code>T</code>	<code>T = <partial name></code>
<hr/>		This sets the partial name of the field. It shouldn't contain a period, be not empty and sensibly consist of simple ascii chars. It is normally required, see above. The value is passed through <code>\pdf_string_from_unicode:nnN</code> .
<hr/>	<code>altname</code>	<code>altname = <string></code>
<hr/>	<code>TU</code>	<code>TU = <string></code>
<hr/>		This sets an alternative name for user interaction. Unlike the name field it can use unicode or periods. The value is passed through <code>\pdf_string_from_unicode:nnN</code>

Table 1: Keys for fields

key	value	required	inheritable	remark
parent	field ID	for non-root fields		
style	style name		defined with <code>create-style</code>	
T, name	string	mostly		
TU, altname	string			
TM, mappingname	string			
FT	name	terminal fields	yes	
setFf,	list of flags		yes	
setfieldflags				
unsetFf,	list of flags		yes	
unsetfieldflags				
V	various		yes	
DV	various		yes	
MaxLen	integer	with Comb	yes	only textfields
Lock	object name			signature field
SV	object name			signature field
Opt	object name			buttons and checkboxes
TI	integer			list fields
I	object name			list fields
AA/K, keystroke	javascript			
AA/F, format	javascript			
AA/V, validate	javascript			
AA/C, calculate	javascript			
DA	string	yes	yes	variable text
Q	0, 1 or 2		yes	variable text
DS				(ignored)
RV				(ignored)

<hr/> mappingname	mappingname = <i><string></i>
<hr/> TM	TM = <i><string></i>
	This sets an alternative name for the export. The value is passed through <code>\pdf_string_from_unicode:nnN</code>
<hr/> mappingname	FT = Btn Tx Ch Sig
<hr/> TM	
	This sets the type of the field, the value should be one of Btn (button), Tx (text), Ch (choice), Sig (signature). The value is of relevance only for terminal fields, but it can be set in a parent and then inherited.
<hr/> setfieldflags	setfieldflags = <i><comma list of flags></i>
<hr/> setFf	setFf = <i><comma list of flags></i>
<hr/> unsetfieldflags	unsetfieldflags = all <i><comma list of flags></i>
<hr/> unsetFf	unsetFf = all <i><comma list of flags></i>
	These keys accept a list of flag names and then sets or unsets them, the resulting value is then used with the <code>/Ff</code> key. Depending on the field type some flags must be set or unset, other are optional or are ignored. The flag name can be given in PDF spelling (RadiosInUnison), in lowercase (radiosinunison), and as number. unsetFf and its alias unsetfieldflags know the special value all which clears all the fields. The list of flags are: ReadOnly , Required , NoExport , Multiline , Password , NoToggleToOff , Radio , Pushbotton , Combo , Edit , Sort , FileSelect , MultiSelect , DoNotSpellCheck , DoNotScroll , Comb , RadiosInUnison , RichText , CommitOnSelChange .
<hr/> V	V = <i><various></i>
	This sets the value of the field. Its format varies depending on the field type, so typically commands for the various type will have to preprocess and sanitize it. The value given here is x-expanded and then added to the dictionary! See the descriptions of individual field types for further information. (Pushbuttons for example don't have a value).
<hr/> DV	DV = <i><various></i>
	The default value, to which the field reverts when a reset-form action is executed. The format of this value is the same as that of DV .
<hr/> MaxLen	MaxLen = <i><integer></i>
	Only relevant for textfields. The value is an integer and describes the maximum length of the field's text in characters. Required if the Comb flag is used.
<hr/> Lock	MaxLen = <i><object name></i>
	Only relevant for signature fields. The value is an object name which should point to a dictionary that specifies a set of form fields that shall be locked when this signature field is signed. The exact format of the dictionary is described in the PDF reference.
<hr/> SV	SV = <i><object name></i>
	Only relevant for signature fields. The value is an object name which should point to a seed value dictionary. The exact format of the dictionary is described in the PDF reference.

<hr/>	Opt	Opt = <i><object name></i>	
			Only relevant for checkboxes, radiobuttons and choice fields. The value is an object name which should point to a array. The exact format of the array is described in the PDF reference.
<hr/>	TI	TI = <i><integer></i>	
			Only relevant for scrollable list boxes. The value is an integer, the top index (the index in the Opt array of the first option visible in the list). Default value: 0
<hr/>	I	I = <i><object name></i>	
			For choice fields that allow multiple selection (MultiSelect flag set). The value is an object name which should point to a array. The exact format of the array is described in the PDF reference (I have no idea what exactly should be added there, perhaps some future test will make it more understandable.)
			The following four keys are used to add javascript ("ECMAScript") code. The values are currently only passed through <code>\pdf_string_from_unicode:nnN</code> , but this perhaps will have to change. The keys will be ignored if a pdfstandard is used that prohibits such actions.
<hr/>	AA/K keystroke	AA/K = <i><string (ECMAScript)></i> keystroke = <i><string (ECMAScript)></i>	
			This adds a keystroke action to the additional action dictionary. The value is passed through <code>\pdf_string_from_unicode:nnN</code> . The action is meant for text and choice fields. It is quite unclear if such an action make sense for non-terminal fields.
<hr/>	AA/F format	AA/F = <i><string (ECMAScript)></i> format = <i><string (ECMAScript)></i>	
			This adds a format action to the additional action dictionary. The value is passed through <code>\pdf_string_from_unicode:nnN</code> . The action is meant for text and choice fields. It is quite unclear if such an action make sense for non-terminal fields.
<hr/>	AA/V validate	AA/V = <i><string (ECMAScript)></i> validate = <i><string (ECMAScript)></i>	
			This adds a validate action to the additional action dictionary. The value is passed through <code>\pdf_string_from_unicode:nnN</code> . It is quite unclear if such an action make sense for non-terminal fields.
<hr/>	AA/C calculate	AA/C = <i><string (ECMAScript)></i> calculate = <i><string (ECMAScript)></i>	
			This adds a calculate action to the additional action dictionary. The value is passed through <code>\pdf_string_from_unicode:nnN</code> . It is quite unclear if such an action make sense for non-terminal fields.
<hr/>	DA	DA = <i><string></i>	
			This contains instructions for the text in text fields. It is stored expanded and parentheses are added around the value.

Table 2: Keys for field annotations

key	value	required	remark
parent	field ID	yes	
width	dim expression	(yes)	default is 0pt
height	dim expression	(yes)	default is 0pt
depth	dim expression	(yes)	default is 0pt
AP/N	appearance name	yes (in PDF 2.0)	
AP/R	appearance name	yes (in PDF 2.0)	
AP/D	appearance name	yes (in PDF 2.0)	
AS	name	yes (in PDF 2.0)	
setF	list of flags		
unsetF	list of flags		
AA/*	javascript	*= F, BI, D, U, E, X, PO, PC,PV, PI	
MK/*	various	*= R, BC, BG, CA, RC, AC, I, RI, IX, IF, TP	

Q Q = left|center|right
align align = left|center|right

The justification of the text.

DS These two keys are currently not implemented as it is unclear if there are of any use.
RV

5 Annot keys

Table 2 summarize the keys which can be used. A number of keys have alias names which are mentioned in the descriptions.

width width = *<dim expression>*
height height = *<dim expression>*
depth depth = *<dim expression>*

These keys allow to set the dimensions of the annotation. The value should be a command that expands to a dimension expression. By default all values are zero.

parent parent = *<field ID>*

This sets the parent. The value should be field ID of an already declared field.

AP/N AP/N = *<appearance name>*
AP/R AP/R = *<appearance name>*
AP/D AP/D = *<appearance name>*

This keys set the normal, rollover and down appearance. Alias names are **appearance**, **rollover-appearance** and **down-appearance**. The value is by default a simple name of an appearance/form Xobject but modules like **l3pdfld-checkbox** change this to allow to add appearances for various states.

AS AS = \langle appearance state name \rangle

This key sets the default appearance state. The value is a name without the starting slash (it is passed through `\pdf_name_from_unicode_e:n`), for checkbox for example `Yes`. If used it should typically have the same value as the V and DV key of the field.

setannotflags	setannotflags = \langle comma list of flags \rangle
setF	setF = \langle comma list of flags \rangle
unsetannotflags	unsetannotflags = all \langle comma list of flags \rangle
unsetF	unsetF = all \langle comma list of flags \rangle

These keys allow to set or unset the annot flags. They expect a comma lists of flag names. Allowed names `Invisible`, `Hidden`, `Print`, `NoZoom`, `NoRotate`, `NoView`, `ReadOnly`, `Locked`, `ToggleNoView`, `LockedContents`, or the lowercase variants or numbers.

AA/* AA/* = \langle string (ECMAScript) \rangle

* should be one of F, Bl, D, U, E, X, PD, PC, PV, PI. Alias names for the first six keys are `onfocus`, `onblur`, `onmousedown`, `onmouseup`, `onenter`, `onexit`. These keys adds then the respective key to the /AA dictionary of the field annotation object. Their value should be javascript code. The /AA dictionary is suppressed if a pdf/A standard is set.

For example

```
onenter={app.alert('Hello');}
```

The following keys add values to the *dynamic appearance dictionary* MK directory. This is only relevant for annotations with dynamic content, like e.g. textfields. The settings can also affect checkboxes and radio buttons if the (deprecated) `NeedAppearances` is set to true.

The MK dictionary can also be added by using `\pdfannot_dict_put:nnn{Widget}{MK}{...}` but the two methods should not be mixed.

MK/R	MK/R = 0 90 180 270
rotate	rotate = 0 90 180 270

These rotates the content of the annotation.

MK/BC	MK/BC = \langle color expression \rangle [\langle model \rangle]{ \langle values \rangle }
bordercolor	bordercolor = \langle color expression \rangle [\langle model \rangle]{ \langle values \rangle }

These colors the border. Internally currently RGB is used. The colors used in \langle color expression \rangle must be known to the `l3color` commands.

MK/BG	MK/BG = \langle color expression \rangle [\langle model \rangle]{ \langle values \rangle }
backgroundcolor	backgroundcolor = \langle color expression \rangle [\langle model \rangle]{ \langle values \rangle }

These colors the background. Internally currently RGB is used. The colors used in \langle color expression \rangle must be known to the `l3color` commands.

MK/CA	MK/CA = $\langle string \rangle$
caption	caption = $\langle string \rangle$

This sets a text for the caption. $\langle string \rangle$ is passed through `\pdf_string_from_unicode:nnN` and parentheses are added automatically. The font used seems to depend on the whims of the PDF reader: At least for checkboxes adobe reader quite insists to always use a symbol font and not a text font. It also shows always only one symbol, regardless how much one put in the string. `hyperref` uses the key names `checkboxsymbol` and `radiosymbol` for this setting.

The remaining key are useful for buttons only, currently no special syntax support is implemented. They will be handled when the code for push buttons is developed and tested.

MK/*	MK/* = $\langle various \rangle$
------	----------------------------------

These keys adds the various entries in the *dynamic appearance dictionary*. * should be one of RC, AC, I, RI, IX, IF, TP. The MK dictionary can also be added by using `\pdfannot_dict_put:nnn{Widget}{MK}{...}` but the two methods should not be mixed.

6 l3pdffield Implementation

```

1  $\langle *package \rangle$ 
2  $\langle @@=pdffield \rangle$ 
3 \NeedsTeXFormat{LaTeX2e}
4 \ProvidesExplPackage{l3pdffield-testphase}{2021-05-14}{0.95d}%
5   {form fields}

```

6.1 hyperref specific command

`hyperref` sets `NeedAppearances` by default. As this is deprecated we disable this.

```

6 \csname HyField@NeedAppearancesfalse\endcsname % suppress NeedAppearances

```

6.2 local variables

```

7 \str_new:N \l__pdffield_tmpa_str
8 \tl_new:N \l__pdffield_tmpa_tl
9 \tl_new:N \l__pdffield_tmpa_keys_tl
10 \cs_new_protected:Npn \__pdffield_tmpa:n #1 {}
11 \cs_new_protected:Npn \__pdffield_tmpa:nn #1 #2 {}
12 \tl_new:N \l__pdffield_currentparent_tl
13 \tl_new:N \l__pdffield_fieldID_tl

```

6.3 messages

```

14 \msg_new:nnn {pdffield}{no-period}
15 {
16   The~field~name~'#1'~contains~a~period. \\
17   This~is~not~allowed. '
18 }
19 \msg_new:nnn {pdffield}{empty-name}
20 {
21   The~field~name~is~empty. \\
22   This~is~not~allowed. '

```

```

23 }
24 \msg_new:nnn {pdffield}{appearance-missing}
25 {
26   The~appearance~definition~‘#1’~is~missing~for~the~#2~appearance.
27 }
28 \msg_new:nnn {pdffield}{not-implemented}
29 {
30   Support~for~’/#1’~is~not~implemented\\
31   The~key~is~ignored.
32 }
33 \msg_new:nnn {pdffield}{key-disabled}
34 {
35   key~‘#2’~is~disabled~and~ignored~in~the~‘#1’~command.\\
36   Use~key~‘#3’~instead.
37 }
38 \msg_new:nnn {pdffield}{parent-field-missing}
39 {
40   The~parent~field~‘#1’~doesn’t~exist\\
41   Create~it~with~\tl_to_str:n{\pdffield_field:nn}
42 }

```

An auxiliary command to disable some keys

`__pdffield_key_disable:nnn`

```

43 \cs_new_protected:Npn \__pdffield_key_disable:nnn #1#2#3
44 {
45   \keys_define:nn {pdffield}
46   {
47     #2 .code:n =
48     {
49       \msg_warning:nnnnn {pdffield}{key-disabled}{#1}{#2}{#3}
50     }
51   }
52 }

```

(End definition for `__pdffield_key_disable:nnn`.)

6.4 bitsets

A bitset for the field flag Ff and an internal copy of the annot bitset.

```

53 \bitset_new:Nn \l__pdffield_Ff_bitset
54 {
55   ReadOnly          = 1,
56   Required          = 2,
57   NoExport          = 3,
58   Multiline         = 13,%Tx
59   Password          = 14,
60   NoToggleToOff     = 15,%Btn, radio button
61   Radio             = 16,%Btn: Radio: 15=1, 16=0
62   Pushbutton        = 17,%Btn: Checkbox: 15=0, 16=0
63                     %Btn: Pushbutton: 16=1
64   Combo             = 18,%Ch: Combo=1 List=0
65   Edit              = 19,%Ch, Combo=1 -> + edit field
66   Sort              = 20,%Ch, not relevant for view...
67   FileSelect        = 21,%Tx

```

```

68     MultiSelect      = 22,%Ch
69     DoNotSpellCheck  = 23,%Tx, Ch (if Combo + Edit set)
70     DoNotScroll      = 24,%Tx
71     Comb             = 25,%Tx, requires MaxLen in dict
72     RadiosInUnison   = 26,%Btn Radio
73     RichText         = 26,%Tx
74     CommitOnSelChange = 27,
75     readonly         = 1,
76     required         = 2,
77     noexport         = 3,
78     multiline        = 13,%Tx
79     password         = 14,
80     notoggletooff    = 15,%Btn, radio button
81     radio            = 16,%Btn: Radio: 15=1, 16=0
82     pushbutton       = 17,%Btn: Checkbox: 15=0, 16=0
83                     %Btn: Pushbutton: 16=1
84     combo            = 18,%Ch: Combo=1 List=0
85     edit             = 19,%Ch, Combo=1 -> + edit field
86     sort             = 20,%Ch, not relevant for view...
87     fileselect       = 21,%Tx
88     multiselect      = 22,%Ch
89     donotspellcheck  = 23,%Tx, Ch (if Combo + Edit set)
90     donotscroll      = 24,%Tx
91     comb             = 25,%Tx, requires MaxLen in dict
92     radiosinunison   = 26,%Btn Radio
93     richtext         = 26,%Tx
94     commitonselchange = 27
95 }
96
97 \bitset_new:Nn \l__pdffield_F_bitset
98 {
99     Invisible        = 1,
100    Hidden            = 2,
101    Print             = 3,
102    NoZoom            = 4,
103    NoRotate          = 5,
104    NoView            = 6,
105    ReadOnly          = 7,
106    Locked            = 8,
107    ToggleNoView      = 9,
108    LockedContents    = 10,
109    invisible         = 1,
110    hidden            = 2,
111    print             = 3,
112    nozoom            = 4,
113    norotate          = 5,
114    noview            = 6,
115    readonly          = 7,
116    locked            = 8,
117    togglenoview      = 9,
118    lockedcontents    = 10
119 }

```

6.5 The field dictionary

The field dictionary is the main object. To be able to set values from the outside it will use a dictionary which can be filled by key-val.

```

120 \pdfdict_new:n {l__pdffield/field}
121 \pdfdict_new:n {l__pdffield/field/AA}
122 \bool_new:N \l__pdffield_root_field_bool

__pdffield_field:n      __pdffield_field:n{<field ID>}

123 \cs_new_protected:Npn __pdffield_field:n #1
124 {
125   \pdf_object_new:nn {__pdffield/field/#1} {dict}
126   \pdf_object_new:nn {__pdffield/field/Kids/#1} {array}
127   \tl_if_empty:NTF \l__pdffield_currentparent_tl
128   {
129     \pdfmanagement_add:nnx
130     { Catalog / AcroForm }
131     { Fields }
132     {\pdf_object_ref:n {__pdffield/field/#1} }
133   }
134   {
135     \exp_args:Ne
136     \pdf_object_if_exist:nTF {__pdffield/field/\l__pdffield_currentparent_tl}
137     {
138       \pdfdict_put:nnx { l__pdffield/field }{Parent}
139       {\exp_args:Ne \pdf_object_ref:n{__pdffield/field/\l__pdffield_currentparent_tl}
140       \seq_gput_right:cx {g__pdffield_field/Kids/\l__pdffield_currentparent_tl _seq}
141       { \exp_args:Ne \pdf_object_ref:n{__pdffield/field/#1}}
142     }
143     {
144       \msg_error:nnx {pdffield}{parent-field-missing}{\l__pdffield_currentparent_tl}
145     }
146   }
147   \seq_new:c {g__pdffield_field/Kids/#1_seq}
148   \pdfdict_put:nnx {l__pdffield/field}
149   {Kids}
150   {
151     \pdf_object_ref:n {__pdffield/field/Kids/#1}
152   }
153   \pdfdict_put:nnx {l__pdffield/field}
154   {Ff}
155   {\bitset_to_arabic:N \l__pdffield_Ff_bitset }
156   \pdfdict_if_empty:nF{l__pdffield/field/AA}
157   {
158     \pdfmeta_standard_verify:nT
159     {annot_widget_no_AA}
160     {
161       \pdf_object_unnamed_write:nx {dict}{\pdfdict_use:n {l__pdffield/field/AA}}
162       \pdfdict_put:nnx
163       {l__pdffield/field}
164       {AA}
165       {\pdf_object_ref_last:}
166     }
167   }

```

```

168 \hook_gput_code:nnn {shipout/lastpage}{pdffield} %xetex needs this ...
169 {
170   \pdf_object_write:nx {__pdffield/field/Kids/#1}
171   {
172     \seq_use:cn{g__pdffield_field/Kids/#1_seq}{~}
173   }
174 }
175 \pdf_object_write:nx {__pdffield/field/#1} { \pdfdict_use:n {l__pdffield/field} }
176 }
177 \cs_new_protected:Npn \pdffield_field:nn #1 #2
178 {
179   \group_begin:
180   \keys_set:nn { pdffield } {#1}
181   \__pdffield_field:n {#2}
182   \group_end:
183 }

```

(End definition for `__pdffield_field:n`.)

6.6 The annot dictionary

We assume that the annotation should really occupy space on the page and leave vertical mode.

`__pdffield_annot:` The command doesn't add grouping, so should only be used inside a group.

```

184 \cs_new_protected:Npn \__pdffield_annot:
185 {
186   \pdfmeta_standard_verify:nF
187   {annot_flags}
188   {
189     \bitset_set_true:Nn \l__pdffield_F_bitset {Print}
190     \bitset_set_false:Nn \l__pdffield_F_bitset {Hidden}
191     \bitset_set_false:Nn \l__pdffield_F_bitset {Invisible}
192     \bitset_set_false:Nn \l__pdffield_F_bitset {NoView}
193   }
194   \pdfannot_dict_put:nnx {widget}{F}{ \bitset_to_arabic:N \l__pdffield_F_bitset }
195   \tl_if_empty:NF \l__pdffield_currentparent_tl
196   {
197     \exp_args:Ne
198     \pdf_object_if_exist:nTF { __pdffield/field/\l__pdffield_currentparent_tl }
199     {
200       \pdfannot_dict_put:nnx {widget}{Parent}
201       {
202         \exp_args:Ne
203         \pdf_object_ref:n{__pdffield/field/\l__pdffield_currentparent_tl}
204       }
205     }
206     {
207       \msg_error:nnx { pdffield }{parent-field-missing}{\l__pdffield_currentparent_tl}
208     }
209   }
210   \mode_leave_vertical:
211   \hbox_to_wd:nn
212   { \l__pdffield_annot_wd_dim }

```

```

213 {
214   \rule [-\l__pdffield_annot_dp_dim]{0pt}{\dim_eval:n{\l__pdffield_annot_ht_dim+\l__pdffield_annot_dp_dim}}
215   \pdfannot_widget_box:nnn
216     { \l__pdffield_annot_wd_dim }
217     { \l__pdffield_annot_ht_dim }
218     { \l__pdffield_annot_dp_dim }
219   \hfill
220 }
221 \tl_if_empty:NF \l__pdffield_currentparent_tl
222 {
223   \seq_if_exist:cTF {g__pdffield_field/Kids/\l__pdffield_currentparent_tl _seq}
224   {
225     \seq_gput_right:cx
226       {g__pdffield_field/Kids/\l__pdffield_currentparent_tl _seq}
227       { \pdfannot_box_ref_last:}
228   }
229   {
230     \msg_error:nnx { pdffield}{parent-field-missing}{\l__pdffield_currentparent_tl}
231   }
232 }
233 }
234 \cs_new_protected:Npn \pdffield_annot:n #1
235 {
236   \group_begin:
237   \keys_set:nn { pdffield } {#1}
238   \__pdffield_annot:
239   \group_end:
240 }

```

(End definition for __pdffield_annot:.)

6.7 auxiliary command for color keys

```

241 \cs_new_protected:Npn \__pdffield_color_set:nn #1 #2
242 {
243   \tl_if_head_eq_charcode:nNTF {#2}[ %]
244   {
245     \__pdffield_color_set_aux:nwn { #1 } #2
246   }
247   {
248     \color_set:nn {#1} {#2}
249   }
250 }
251
252 \cs_new_protected:Npn \__pdffield_color_set_aux:nwn #1 [#2] #3
253 {
254   \color_set:nnn {#1}{#2}{#3}
255 }
256

```

6.8 Field keys

The names. The main name should not be empty, it is added to the dictionary when the field is created. A new name means a new field. The other names can only be set when the field is created, so we put them in the field group.

```

257 \cs_new_protected:Npn \__pdffield_value_handler:nN #1#2
258 {
259     \tl_set:Nn #2 {#1}
260 }
261 \keys_define:nn { pdffield }
262 {
263     ,parent .tl_set:N = \l__pdffield_currentparent_tl
264     ,parent .groups:n = {field,annot}
265     ,T .code:n =
266     {
267         \pdf_string_from_unicode:nnN {utf8/string-raw}{#1}\l__pdffield_tmpa_str
268         \str_if_in:NnT \l__pdffield_tmpa_str {.}
269         {
270             \msg_error:nxx {pdffield}{no-period}{\l__pdffield_tmpa_str}
271         }
272         \str_if_empty:NTF\l__pdffield_tmpa_str
273         {
274             \msg_warning:nn {pdffield}{empty-name}
275             \pdfdict_remove:nn { l__pdffield/field }{T}
276         }
277         {
278             \pdfdict_put:nxx { l__pdffield/field }{T}{(\l__pdffield_tmpa_str)}
279         }
280     }
281     ,T .value_required:n = true
282     ,T .groups:n = {field}
283     ,name .meta:n          = {T={#1}}
284     ,name .value_required:n = true
285     ,name .groups:n = {field}
286     ,TU .groups:n = {field}
287     ,TU .code:n =
288     {
289         \tl_if_empty:NTF {#1}
290         {
291             \pdfdict_remove:nn { l__pdffield/field }{TU}
292         }
293         {
294             \pdf_string_from_unicode:nnN {utf8/string}{#1}\l__pdffield_tmpa_str
295             \pdfdict_put:nxx { l__pdffield/field }{TU}{\l__pdffield_tmpa_str}
296         }
297     }
298     ,TU .groups:n = {field}
299     ,altname .meta:n      = {TU={#1}}
300     ,altname .groups:n = {field}
301     ,TM .code:n =
302     {
303         \tl_if_empty:NTF {#1}
304         {
305             \pdfdict_remove:nn { l__pdffield/field }{TM}
306         }
307         {
308             \pdf_string_from_unicode:nnN {utf8/string}{#1}\l__pdffield_tmpa_str
309             \pdfdict_put:nxx { l__pdffield/field }{TM}{\l__pdffield_tmpa_str}
310         }

```

```

311     }
312     ,TM .groups:n = {field}
313     ,mappingname .meta:n = {TM={#1}}
314     ,mappingname .groups:n = {field}
315     ,FT .choices:nn =
316     { Btn, Tx, Ch, Sig }
317     {
318         \pdfdict_put:nnn { l__pdffield/field }{FT}{ /#1 }
319     }
320     ,FT .groups:n = {field}
321     ,V .code:n =
322     {
323         \tl_if_empty:nTF {#1}
324         {
325             \pdfdict_remove:nn { l__pdffield/field }{V}
326         }
327         {
328             \__pdffield_value_handler:nN{#1}\l__pdffield_tmpa_str
329             \pdfdict_put:nnx { l__pdffield/field }{V}{ \l__pdffield_tmpa_str }
330         }
331     }
332     ,V .groups:n = {field}
333     ,DV .code:n =
334     {
335         \tl_if_empty:nTF {#1}
336         {
337             \pdfdict_remove:nn { l__pdffield/field }{DV}
338         }
339         {
340             \__pdffield_value_handler:nN{#1}\l__pdffield_tmpa_str
341             \pdfdict_put:nnx { l__pdffield/field }{DV}{ \l__pdffield_tmpa_str }
342         }
343     }
344     ,DV .groups:n = {field}
345     ,MaxLen .code:n =
346     {
347         \tl_if_empty:nTF {#1}
348         {
349             \pdfdict_remove:nn { l__pdffield/field }{MaxLen}
350         }
351         {
352             \pdfdict_put:nnx { l__pdffield/field }{MaxLen}{ #1 }
353         }
354     }
355     ,MaxLen .groups:n = {field}
356     ,Lock .code:n =
357     {
358         \tl_if_empty:nTF {#1}
359         {
360             \pdfdict_remove:nn { l__pdffield/field }{Lock}
361         }
362         {
363             \pdfdict_put:nnx { l__pdffield/field }{Lock}{ \pdf_object_ref:n{#1} }
364         }

```

```

365     }
366     ,Lock .groups:n = {field}
367     ,SV .code:n =
368     {
369         \tl_if_empty:nTF {#1}
370         {
371             \pdfdict_remove:nn { l__pdffield/field }{SV}
372         }
373         {
374             \pdfdict_put:nxx { l__pdffield/field }{SV}{ \pdf_object_ref:n{#1} }
375         }
376     }
377     ,SV .groups:n = {field}
378     ,Opt .code:n =
379     {
380         \tl_if_empty:nTF {#1}
381         {
382             \pdfdict_remove:nn { l__pdffield/field }{Opt}
383         }
384         {
385             \pdfdict_put:nxx { l__pdffield/field }{Opt}{ \pdf_object_ref:n{#1} }
386         }
387     }
388     ,Opt .groups:n = {field}
389     ,TI .code:n =
390     {
391         \tl_if_empty:nTF {#1}
392         {
393             \pdfdict_remove:nn { l__pdffield/field }{TI}
394         }
395         {
396             \pdfdict_put:nxx { l__pdffield/field }{TI}{ #1 }
397         }
398     }
399     ,TI .groups:n = {field}
400     ,I .code:n =
401     {
402         \tl_if_empty:nTF {#1}
403         {
404             \pdfdict_remove:nn { l__pdffield/field }{I}
405         }
406         {
407             \pdfdict_put:nxx { l__pdffield/field }{I}{ \pdf_object_ref:n{#1} }
408         }
409     }
410     ,I .groups:n = {field}
411 }

```

Flags. We don't add lots of individual keys but map the key names directly

```

412 \keys_define:nn { pdffield }
413 {
414     ,setFf .code:n =
415     {
416         \clist_map_inline:nn {#1}
417         {

```

```

418         \bitset_set_true:Nn \l__pdffield_Ff_bitset {##1}
419     }
420 }
421 ,setFf .groups:n = {field}
422 ,setfieldflags .meta:n =
423     {setFf={#1}}
424 ,setfieldflags .groups:n = {field}
425 ,unsetFf .multichoice:
426 ,unsetFf / all .code:n = { \bitset_clear:N \l__pdffield_Ff_bitset}
427 ,unsetFf / unknown .code:n =
428     {
429         \bitset_set_false:Nn \l__pdffield_Ff_bitset {##1}
430     }
431 ,unsetFf .groups:n = {field}
432 ,unsetfieldflags .meta:n = {unsetFf={#1}}
433 ,unsetfieldflags .groups:n = {field}
434 }
435

```

Keys for the AA dictionary. They all trigger a javascript option. K=keystroke, F=format, V=validate, C=calculate

```

436 \cs_set_protected:Npn \__pdffield_tmpa:n #1 %
437 {
438     \keys_define:nn { pdffield }
439     {
440         AA/#1 .code:n =
441         {
442             \pdf_string_from_unicode:nnN {utf8/string-raw}{##1}\l__pdffield_tmpa_str
443             \str_if_empty:NTF \l__pdffield_tmpa_str
444             {
445                 \pdfdict_remove:nn {l__pdffield/field/AA}{#1}
446             }
447             {
448                 \pdfdict_put:nnx {l__pdffield/field/AA}
449                 {#1}
450                 {<</S/JavaScript/JS(\l__pdffield_tmpa_str)>>}
451             }
452         },
453         AA/#1 .groups:n = {field}
454     }
455 }
456
457 \clist_map_inline:nn {K,F,V,C}{\__pdffield_tmpa:n{#1}}
458
459 \cs_set_protected:Npn \__pdffield_tmpa:nn #1 #2
460 {
461     \keys_define:nn { pdffield }
462     {
463         #1 .meta:nn =
464         { pdffield }{AA/#2={##1}},
465         #1 .groups:n = {field}
466     }
467 }
468 \__pdffield_tmpa:nn {keystroke}{K}

```

```

469 \__pdffield_tmpa:nn {format} {F}
470 \__pdffield_tmpa:nn {validate} {V}
471 \__pdffield_tmpa:nn {calculate}{C}
472
473
474 \keys_define:nn { pdffield }
475 {
476   DA .code:n =
477   {
478     \tl_if_empty:nTF {#1}
479     {
480       \pdfdict_remove:nn { l__pdffield/field }{DA}
481     }
482     {
483       \pdfdict_put:nxx { l__pdffield/field }{DA}{ (#1) }
484     }
485   }
486   ,DA .groups:n = {field}
487   ,Q .choices:nn = {left,center,right}
488   {
489     \pdfdict_put:nxx { l__pdffield/field }{Q}{ \int_eval:n{\l_keys_choice_int-1} }
490   }
491   ,Q / .code:n = { \pdfdict_remove:nn { l__pdffield/field }{Q} }
492   ,Q .groups:n = {field}
493   ,align .meta:n={Q=#1}
494   ,DS .code:n =
495   {
496     \msg_warning:nnn {pdffield}{not-implemented}{DS}
497   }
498   ,DS .groups:n = {field}
499   ,RV .code:n =
500   {
501     \msg_warning:nnn {pdffield}{not-implemented}{RV}
502   }
503   ,RV .groups:n = {field}
504 }

```

6.9 Annotation keys

The size of the field annotation

```

505 \dim_new:N \l__pdffield_annot_ht_dim
506 \dim_new:N \l__pdffield_annot_wd_dim
507 \dim_new:N \l__pdffield_annot_dp_dim
508
509 \keys_define:nn { pdffield }
510 {
511   ,width .dim_set:N = \l__pdffield_annot_wd_dim
512   ,height .dim_set:N = \l__pdffield_annot_ht_dim
513   ,depth .dim_set:N = \l__pdffield_annot_dp_dim
514   ,width .initial:n = Opt
515   ,height .initial:n = Opt
516   ,depth .initial:n = Opt
517 }

```

```

518 \keys_define:nn { pdfffield }
519 {
520     %parent is defined in field
521     ,AS .code:n =
522     {
523         \tl_if_empty:nTF {#1}
524         {
525             \pdfannot_dict_remove:nn { widget }{AS}
526         }
527         {
528             \pdfannot_dict_put:nnx {widget}{AS}{\pdf_name_from_unicode_e:n{#1}}
529         }
530     }
531     ,AS .groups:n = annot
532 }
533 \cs_new_protected:Npn \__pdfffield_appearance_handler:nnn #1#2#3
534 {
535     \pdfxform_if_exist:nTF { #1 }
536     {
537         \pdfannot_dict_put:nnx {widget/AP}{#2}
538         {
539             \pdfxform_ref:n {#1}
540         }
541     }
542     {
543         \msg_error:nnnn{pdfffield}{appearance-missing}{#1}{#3}
544     }
545 }
546 \keys_define:nn { pdfffield }
547 {
548     AP/N .code:n =
549     {
550         \tl_if_empty:nTF {#1}
551         {
552             \pdfannot_dict_remove:nn { widget/AP }{N}
553         }
554         {
555             \__pdfffield_appearance_handler:nnn {#1}{N}{normal}
556         }
557     }
558     ,AP/N .groups:n = annot
559     ,appearance .meta:n = {AP/N={#1}}
560 }
561 \keys_define:nn { pdfffield }
562 {
563     AP/R .code:n =
564     {
565         \tl_if_empty:nTF {#1}
566         {
567             \pdfannot_dict_remove:nn { widget/AP }{R}
568         }
569         {
570             \__pdfffield_appearance_handler:nnn {#1}{R}{rollover}
571         }
572     }
573 }

```

```

572     }
573     ,AP/R .groups:n = annot
574     ,rollover-appearance .meta:n = {AP/R={#1}}
575 }
576 \keys_define:nn { pdffield }
577 {
578     AP/D .code:n =
579     {
580         \tl_if_empty:nTF {#1}
581         {
582             \pdfannot_dict_remove:nn { widget/AP }{D}
583         }
584         {
585             \__pdffield_appearance_handler:nnn {#1}{D}{rollover}
586         }
587     }
588     ,AP/D .groups:n = annot
589     ,down-appearance .meta:n = {AP/D={#1}}
590 }
591
592 \keys_define:nn { pdffield }
593 {
594     MK/R .choices:nn = {0,90,180,270}
595     {
596         \pdfannot_dict_put:nnx {widget/MK}{R}{#1}
597     }
598     ,MK/R / .code:n =
599     {
600         \pdfannot_dict_remove:nn { widget/MK }{R}
601     }
602     ,MK/R .groups:n = annot
603     ,rotate .meta:n = {MK/R=#1}
604 }
605
606 \keys_define:nn { pdffield }
607 {
608     MK/BC .code:n =
609     {
610         \tl_if_empty:nTF {#1}
611         {
612             \pdfannot_dict_remove:nn { widget/MK }{BC}
613         }
614         {
615             \__pdffield_color_set:nn {__pdffield/tmp}{#1}
616             \color_export:nnN{__pdffield/tmp}{space-sep-rgb}\l__pdffield_tmpa_tl
617             \pdfannot_dict_put:nnx {widget/MK}{BC}{[\l__pdffield_tmpa_tl]}
618         }
619     }
620     ,MK/BC .groups:n = annot
621     ,bordercolor .meta:n = {MK/BC=#1}
622 }
623
624 \keys_define:nn { pdffield }
625 {

```

```

626 MK/BG .code:n =
627 {
628   \tl_if_empty:nTF {#1}
629   {
630     \pdfannot_dict_remove:nn { widget/MK }{BG}
631   }
632   {
633     \__pdffield_color_set:nn {__pdffield/tmp}{#1}
634     \color_export:nnN{__pdffield/tmp}{space-sep-rgb}\l__pdffield_tmpa_tl
635     \pdfannot_dict_put:nnx {widget/MK}{BG}{[\l__pdffield_tmpa_tl]}
636   }
637 }
638 ,MK/BG .groups:n = annot
639 ,bordercolor .meta:n = {MK/BG=#1}
640 }
641
642
643 \keys_define:nn { pdffield }
644 {
645   MK/CA .code:n =
646   {
647     \tl_if_empty:nTF {#1}
648     {
649       \pdfannot_dict_remove:nn { widget/MK }{CA}
650     }
651     {
652       \pdf_string_from_unicode:nnN {utf8/string}{#1}\l__pdffield_tmpa_str
653       \pdfannot_dict_put:nnx {widget/MK}{CA}{\l__pdffield_tmpa_str}
654     }
655   }
656   ,MK/CA .groups:n = annot
657   ,caption .meta:n = {MK/CA=#1}
658 }
659
660 \cs_set_protected:Npn \__pdffield_tmpa:n #1
661 {
662   \keys_define:nn { pdffield }
663   {
664     MK/#1 .code:n =
665     {
666       \tl_if_empty:nTF {##1}
667       {
668         \pdfannot_dict_remove:nn { widget/AP }{#1}
669       }
670       {
671         \pdfannot_dict_put:nnx {widget/MK}{#1}{##1}
672       }
673     }
674     ,MK/#1 .groups:n = annot
675   }
676 }
677
678 \clist_map_inline:nn {RC,AC,I,RI,IX,IF,TP}
679 { \__pdffield_tmpa:n {#1} }

```

Flags.

```
680 \keys_define:nn { pdfffield }
681 {
682   ,setF .code:n =
683   {
684     \clist_map_inline:nn {#1}
685     {
686       \bitset_set_true:Nn \l__pdfffield_F_bitset {##1}
687     }
688   }
689   ,setF .groups:n = annot
690   ,setannotflags .meta:nn =
691   { pdfffield }{setF={#1}}
692   ,setannotflags .groups:n = annot
693   ,unsetF .multichoice:
694   ,unsetF / all .code:n = { \bitset_clear:N \l__pdfffield_F_bitset}
695   ,unsetF / unknown .code:n =
696   {
697     \bitset_set_false:Nn \l__pdfffield_F_bitset {#1}
698   }
699   ,unsetF .groups:n = annot
700   ,unsetannotflags .meta:nn =
701   { pdfffield }{unsetF= {#1} }
702   ,unsetannotflags .groups:n = annot
703 }
704
```

Keys for the AA dictionary. They all trigger a javascript option. Fo = onfocus, Bl = onblur, D = onmousedown, U = onmouseup, E = onenter, X = onexit, PO = pageopen, PC = pageclose, PV = pagevisible, PI = pageinvisible

```
705 \cs_set_protected:Npn \__pdfffield_tmpa:n #1 %
706 {
707   \keys_define:nn { pdfffield }
708   {
709     AA/#1 .code:n =
710     {
711       \pdf_string_from_unicode:nnN {utf8/string-raw}{##1}\l__pdfffield_tmpa_str
712       \str_if_empty:NTF \l__pdfffield_tmpa_str
713       {
714         \pdfannot_dict_remove:nn {widget/AA}{#1}
715       }
716       {
717         \pdfannot_dict_put:nnx {widget/AA}
718         {#1}
719         {<</S/JavaScript/JS(\l__pdfffield_tmpa_str)>>}
720       }
721     },
722     ,AA/#1 .groups:n = annot
723   }
724 }
725
726 \clist_map_inline:nn {Fo,Bl,D,U,E,X,PO,PC,PV,PI}{\__pdfffield_tmpa:n{#1}}
727
728 \cs_set_protected:Npn \__pdfffield_tmpa:nn #1 #2
```

```

729 {
730   \keys_define:nn { pdffield }
731   {
732     #1 .meta:nn =
733       { pdffield }{AA/#2={##1}},
734     #1 .groups:n = {annot}
735   }
736 }
737 \__pdffield_tmpa:nn {onfocus} {Fo}
738 \__pdffield_tmpa:nn {onblur} {Bl}
739 \__pdffield_tmpa:nn {onmousedown}{D}
740 \__pdffield_tmpa:nn {onmouseup}{U}
741 \__pdffield_tmpa:nn {onenter} {E}
742 \__pdffield_tmpa:nn {onexit} {X}

```

6.10 Appearances

`\pdffield_appearance:nn`

```

743 \cs_new_protected:Npn \pdffield_appearance:nn #1 #2
744 {
745   \pdfxform_new:nnn {#1}{#2}
746 }
747
748 \cs_set_eq:NN \pdffield_store_appearance:nn\pdffield_appearance:nn

```

(End definition for `\pdffield_appearance:nn`. This function is documented on page 4.)

6.11 Setup command

```

749 \keys_define:nn { pdffield / setup }
750 {
751   ,create-style .code:n = { \__pdffield_style_create:nn #1 }
752   ,preset-checkbox .code:n =
753   {
754     \keys_define:nn { pdffield }
755     {
756       __pdffield/preset/checkbox .meta:n = {#1},
757     }
758   }
759   ,preset-textfield .code:n =
760   {
761     \keys_define:nn { pdffield }
762     {
763       __pdffield/preset/textfield .meta:n = {#1},
764     }
765   }
766 }
767 \keys_set:nn{ pdffield / setup }{preset-checkbox={}}
768 \keys_set:nn{ pdffield / setup }{preset-textfield={}}
769
770 \cs_new_protected:Npn \__pdffield_style_create:nn #1#2
771 {
772   \keys_define:nn { pdffield }
773   {

```

```

774     __pdfffield/style/#1 .meta:n = {#2},
775   }
776 }
777
778
779 \cs_new_protected:Npn \pdfffield_setup:n #1
780 {
781   \keys_set:nn{ pdfffield / setup }{#1}
782 }
783
784 \keys_define:nn { pdfffield }
785 {
786   style .code:n = {\keys_set:nn {pdfffield}{__pdfffield/style/#1={#1}}}
787 }
788 \end{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	
<code>\</code>	16, 21, 30, 35, 40
A	
<code>AA/*</code>	9
<code>AA/C</code>	7
<code>AA/F</code>	7
<code>AA/K</code>	7
<code>AA/V</code>	7
<code>align</code>	8
<code>altname</code>	4
<code>AP/D</code>	8
<code>AP/N</code>	8
<code>AP/R</code>	8
<code>AS</code>	9
B	
<code>backgroundcolor</code>	9
bitset commands:	
<code>\bitset_clear:N</code>	426, 694
<code>\bitset_new:Nn</code>	53, 97
<code>\bitset_set_false:Nn</code>	190, 191, 192, 429, 697
<code>\bitset_set_true:Nn</code>	189, 418, 686
<code>\bitset_to_arabic:N</code>	155, 194
bool commands:	
<code>\bool_new:N</code>	122
<code>bordercolor</code>	9
C	
<code>calculate</code>	7
caption	
<code>\caption</code>	10
clist commands:	
<code>\clist_map_inline:nn</code>	416, 457, 678, 684, 726
color commands:	
<code>\color_export:nnN</code>	616, 634
<code>\color_set:nn</code>	248
<code>\color_set:nnn</code>	254
<code>create-style</code>	4
cs commands:	
<code>\cs_new_protected:Npn</code>	10, 11, 43, 123, 177, 184, 234, 241, 252, 257, 533, 743, 770, 779
<code>\cs_set_eq:NN</code>	748
<code>\cs_set_protected:Npn</code>	436, 459, 660, 705, 728
<code>\csname</code>	6
D	
<code>DA</code>	7
<code>depth</code>	8
dim commands:	
<code>\dim_eval:n</code>	214
<code>\dim_new:N</code>	505, 506, 507
<code>DS</code>	8
<code>DV</code>	6
E	
<code>\endcsname</code>	6
exp commands:	
<code>\exp_args:Ne</code>	135, 139, 141, 197, 202

	F		<code>\NeedsTeXFormat</code>	3
<code>\Form</code>	2		O	
<code>format</code>	7		<code>Opt</code>	7
	G		P	
group commands:		parent	4, 8	
<code>\group_begin:</code>	179, 236	pdf commands:		
<code>\group_end:</code>	182, 239	<code>\pdf_name_from_unicode_e:n</code>	528	
	H	<code>\pdf_object_if_exist:nTF</code>	136, 198	
hbox commands:		<code>\pdf_object_new:nn</code>	125, 126	
<code>\hbox_to_wd:nn</code>	211	<code>\pdf_object_ref:n</code>	132,	
height	8	139, 141, 151, 203, 363, 374, 385, 407		
<code>\hfill</code>	219	<code>\pdf_object_ref_last:</code>	165	
hook commands:		<code>\pdf_object_unnamed_write:nn</code>	161	
<code>\hook_gput_code:nnn</code>	168	<code>\pdf_object_write:nn</code>	170, 175	
	I	<code>\pdf_string_from_unicode:nnN</code>	4,	
I	7	6, 7, 10, 267, 294, 308, 442, 652, 711		
int commands:		pdfannot commands:		
<code>\int_eval:n</code>	489	<code>\pdfannot_box_ref_last:</code>	227	
	K	<code>\pdfannot_dict_put:nnn</code>	194, 200,	
keys commands:		528, 537, 596, 617, 635, 653, 671, 717		
<code>\l_keys_choice_int</code>	489	<code>\pdfannot_dict_remove:nn</code>	525, 552,	
<code>\keys_define:nn</code>	45, 261,	567, 582, 600, 612, 630, 649, 668, 714		
412, 438, 461, 474, 509, 518, 546,		<code>\pdfannot_widget_box:nnn</code>	4, 215	
561, 576, 592, 606, 624, 643, 662,		pdfdict commands:		
680, 707, 730, 749, 754, 761, 772, 784		<code>\pdfdict_if_empty:nTF</code>	156	
<code>\keys_set:nn</code>	180, 237, 767, 768, 781, 786	<code>\pdfdict_new:n</code>	120, 121	
keystroke	7	<code>\pdfdict_put:nnn</code>	138, 148, 153, 162,	
	L	278, 295, 309, 318, 329, 341, 352,		
Lock	6	363, 374, 385, 396, 407, 448, 483, 489		
	M	<code>\pdfdict_remove:nn</code>		
mappingname	6 275, 291, 305, 325, 337, 349,		
MaxLen	6	360, 371, 382, 393, 404, 445, 480, 491		
MK/*	10	<code>\pdfdict_use:n</code>	161, 175	
MK/BC	9	pdffield commands:		
MK/BG	9	<code>\pdffield_annot:n</code>	4, 234	
MK/CA	10	<code>\pdffield_appearance:nn</code>		
MK/R	9 4, 743, 743, 748		
mode commands:		<code>\pdffield_field:nn</code>	3, 4, 41, 177	
<code>\mode_leave_vertical:</code>	210	<code>\pdffield_setup:n</code>	4, 779	
msg commands:		<code>\pdffield_store_appearance:nn</code>	748	
<code>\msg_error:nnn</code>	144, 207, 230, 270	pdffield internal commands:		
<code>\msg_error:nnnn</code>	543	<code>__pdffield_annot:</code>	184, 184, 238	
<code>\msg_new:nnn</code>	14, 19, 24, 28, 33, 38	<code>\l__pdffield_annot_dp_dim</code>		
<code>\msg_warning:nn</code>	274 214, 218, 507, 513		
<code>\msg_warning:nnn</code>	496, 501	<code>\l__pdffield_annot_ht_dim</code>		
<code>\msg_warning:nnnnn</code>	49 214, 217, 505, 512		
	N	<code>\l__pdffield_annot_wd_dim</code>		
name	4 212, 216, 506, 511		
		<code>__pdffield_appearance_handler:nnn</code>		
	 533, 555, 570, 585		
		<code>__pdffield_color_set:nn</code>	241, 615, 633	
		<code>__pdffield_color_set_aux:nwn</code>		
	 245, 252		

