

The **l3pdf** module
Commands to create form fields
L^AT_EX PDF management testphase bundle

The L^AT_EX Project*

Version 0.96j, released 2024-08-17

1 **l3pdf** 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-testphase}
```

The source code is split into various submodules. All code is combined in the `sty`, but the documentation is in individual PDF.

l3pdf This contains the basic commands and keys to create a form field.

l3pdf-checkbox The code to create checkboxes.

l3pdf-textfield The code to create text fields.

l3pdf-radiobutton The code to create radio buttons.

l3pdf-pushbutton The code to create push buttons.

l3pdf-choice The code to create choice fields (lists and drop-down/combo fields).

l3pdf-action Code related to actions, mostly submit and reset actions.

l3pdf-signature (not done yet) Code for signature fields

Form initialization (not done yet) The `\Form` command/environment of `hyperref` initialize a few things like fonts for text fields which should be moved. It is not strictly necessary to have this code, most examples work without it, but in case of problems it is possible to do the initialization by using the `hyperref` command.

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

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

The code requires the new PDF management. The code makes use of `l3pdfxform` 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).

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

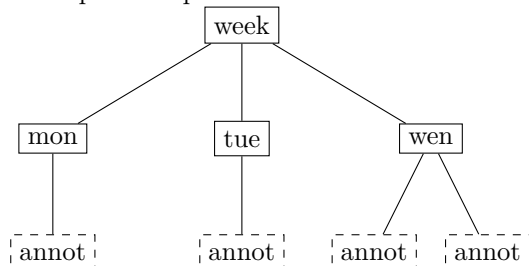
```
\DocumentMetadata{}
\documentclass{article}
\usepackage{hyperref}
\usepackage{l3pdffield-testphase}
\begin{document}
\Form
```

2 Some background

A document can contain a 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 than 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 maps 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.

2.2 Tagged PDF

Field annotations are (like link annotations) not part of the page stream. But they are obviously nevertheless meaningful content and must be consider if a PDF is “tagged”, that means if a structure is added.

According to the PDF references fields should be tagged by adding a `Form` structure element containing the object reference to a field annotations. Fields with more than one annotation like radio buttons need a `Form` structure for every one. Additional some cross references to structure relevant object like the parent tree are needed, for more info check the documentation of the `tagpdf` package.

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.

The commands of this module already contain the needed support. So if `tagpdf` is used and tagging activated the fields will be added as `Form` element to the structure where they are created. It is possible to deactivate tagging for a field annotation by setting the `tag` to false as described below.

If `lualatex` is used tagging require either that `tagpdf` is used with the option `global-mc`, or `mc-chunks` must be correctly closed manually, as the automatic code can't escape the grouping.

It is recommended to use the `TU/altname` key to give the field a readable name.

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:nne{Catalog/AcroForm}{Fields}{<obj ref>}
```

```
\pdffield_annot:n \pdffield_annot:n{<key val list>}
```

```
\pdffield_annot:V
```

This creates a new field annotation. It is a widget annotation box created with `\pdfannot_widget_box:nnn`, and it is possible to add values to its dictionary by using `\pdfannot_dict_put:nnn {widget}...`. 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.

```
\pdffield_annot_ref_last: \pdffield_annot_ref_last:
```

If a tagged PDF should be created, the object of the annotation of a field should be referenced in the `Form` structure element. This command allows to retrieve the reference to this object.

```
\pdffield_appearance:nn \pdffield_appearance:nn{<name>}{<content>}
```

This is a small wrapper around `\pdfxform_new:nnn` (which could be used too) to create an appearance. To avoid name clashes `<name>` should start with a module part, e.g. `mymodule/appearance/cross`.

`\pdfffield_setup:n` `\pdfffield_setup:n{<key-val>}`

This command allows to preset some field settings.

It knows currently two keys:

`create-style` `create-style = {<name>}{<key-val>}`

This defines a style which can then be used with the `style` key. `{<key-val>}` can be an arbitrary collection of the keys of the module.

`style` `style = {<style>}`

This uses a style define with the previous `create-style`.

`preset-checkbox` `preset-checkbox={<key-val>}`

This allows to set default keys for a checkbox.

`preset-radio` `preset-radio={<key-val>}`

This allows to set default keys for a radio button.

`preset-textfield` `preset-textfield={<key-val>}`

This allows to set default keys for a text field.

4 Special keys

`value` `value = {<value>}`

`default` `default={<value>}`

These two keys pass the value to a handler which can be redefined. Their exact behaviour depends on field type. Please check their documentation.

5 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.

`parent` `parent = <field ID>`

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 `/Parent` key, and also add reference of the kid as `/Kid` in the parent field.

`name` `name = <partial name>`

`T` `T = <partial name>`

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 `\pdf_string_from_unicode:nnN`.

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 ch
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)

altname altname = *<string>*
TU TU = *<string>*

This sets an alternative name for user interaction. Unlike the name field it can use unicode or periods. The value is passed through `\pdf_string_from_unicode:nnN`

mappingname mappingname = *<string>*
TM TM = *<string>*

This sets an alternative name for the export. The value is passed through `\pdf_string_from_unicode:nnN`

FT FT = Btn|Tx|Ch|Sig

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.

setfieldflags setfieldflags = *<comma list of flags>*
setFf setFf = *<comma list of flags>*
unsetfieldflags unsetfieldflags = all | *<comma list of flags>*
unsetFf unsetFf = all | *<comma list of flags>*

These keys accept a list of flag names and then sets or unsets them, the resulting value is then used with the `/Ff` 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**.

V V = *<various>*

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).

DV DV = *<various>*

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**.

MaxLen MaxLen = *<integer>*

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.

Lock MaxLen = *<object name>*

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.

SV SV = *<object name>*

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.

Opt Opt = *<object name>*

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.

TI TI = *<integer>*

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

I I = *<object name>*

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 expanded. It is recommended to store the javascript in a stream object and to pass the object reference, but passing a string (including parentheses) is possible too. The keys will be ignored if a pdfstandard is used that prohibits such actions.

AA/K AA/K = *<ECMAScript>*
keystroke keystroke = *<ECMAScript>*

This adds a keystroke action to the additional action dictionary. The action is meant for text and choice fields. It is quite unclear if such an action make sense for non-terminal fields.

AA/F AA/F = *<ECMAScript>*
format format = *<ECMAScript>*

This adds a format action to the additional action dictionary. The action is meant for text and choice fields. It is quite unclear if such an action make sense for non-terminal fields.

AA/V AA/V = *<ECMAScript>*
validate validate = *<ECMAScript>*

This adds a validate action to the additional action dictionary. It is quite unclear if such an action make sense for non-terminal fields.

AA/C AA/C = *<string (ECMAScript)>*
calculate calculate = *<string (ECMAScript)>*

This adds a calculate action to the additional action dictionary. It is quite unclear if such an action make sense for non-terminal fields. If an calculate action is used, the field will be added to the AcroForm/CO array to define the calculation order. The order can be controlled through the following key **sortkey**.

sortkey `sortkey = <string>`

This sets a sortkey for fields with calculate action. The sortkeys are sorted lexically with `\str_compare:nNnTF`. fields without sortkey will get an empty sortkey and so be at the begin, the order of fields with the same sortkey is not defined. The module only sorts fields created with the commands of this module, the sorting of fields created by `hyperref` is independent.

DA `DA = <string>`

This contains instructions for the text in text fields. It is stored expanded and parentheses are added around the value.

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

The justification of the text.

DS
RV

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

fieldID `fieldID = <field ID>`

For experts only! This stores `<field ID>` in an internal variable. The variable is not used by the basic commands, but by the commands to create the various field types. Check their documentation for use cases.

6 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.

tag `tag = true|false`

This key is related to tagging and enables/disables the tagging.

parent `parent = <field ID>`

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

Table 2: Keys for field annotations

key	value	required	remark
parent	field ID	yes	
width	dim expression	(yes)	default is Opt
height	dim expression	(yes)	default is Opt
depth	dim expression	(yes)	default is Opt
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	

AP/N	AP/N = <i><appearance name></i>
appearance	appearance = <i><appearance name></i>
AP/R	AP/R = <i><rollover appearance name></i>
rollover-appearance	rollover-appearance = <i><rollover appearance name></i>
AP/D	AP/D = <i><down appearance name></i>
down-appearance	down appearance = <i><down appearance name></i>

This keys set the normal, rollover and down appearance. The names `appearance`, `rollover-appearance` and `down-appearance` are aliases. The value is by default a simple name of an appearance/form Xobject but modules like `l3pdf-field-checkbox` change this to allow to add appearances for various states. So check the documentation for the various field types for the exact format of the value.

AS AS = *<appearance state name>*

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 = <i><comma list of flags></i>
setF	setF = <i><comma list of flags></i>
unsetannotflags	unsetannotflags = all <i><comma list of flags></i>
unsetF	unsetF = all <i><comma list of flags></i>

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/* = $\langle ECMA\text{Script} \rangle$

* should be one of Fo, Bl, D, U, E, X, PO, 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 value is expanded but not escaped. It is recommended to store the code in a stream object and to use the object reference as value. 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.

MK/RC MK/RC = $\langle string \rangle$
rollover-caption rollover-caption = $\langle string \rangle$

This sets a text for the rollover-caption. $\langle string \rangle$ is passed through `\pdf_string_from_unicode:nnN` and parentheses are added automatically. The key should be used only with `pushbuttons`. It is unclear if is actually used by the PDF viewer, but the `pushbuttons` modules uses the argument also to setup the appearance.

MK/AC	MK/AC = $\langle string \rangle$
down-caption	down-caption = $\langle string \rangle$

This sets a text for the down-caption. $\langle string \rangle$ is passed through `\pdf_string_from_unicod`:`nnN` and parentheses are added automatically. The key should be used only with `pushbuttons`. It is unclear if is actually used by the PDF viewer, but the `pushbuttons` modules uses the argument also to setup the appearance.

The remaining key are like the two above useful for `pushbuttons` only. Currently no special syntax support is implemented. They will be handled if needed when the code for push buttons is developed and tested.

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

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

7 l3pdffield Implementation

```

1  $\langle *package \rangle$ 
2  $\langle @@=pdffield \rangle$ 
3 \NeedsTeXFormat{LaTeX2e}
4 \ProvidesExplPackage{l3pdffield-testphase}{2024-08-17}{0.96j}%
5 {form fields}

```

7.1 hyperref specific command

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

```

6 \csname HyField@NeedAppearancesfalse\endcsname % suppress NeedAppearances

```

7.2 local variables

Some tmp variables, and a variable for the current parent and the current fieldID.

```

\l__pdffield_tmpa_str
\l__pdffield_tmpb_str
\l__pdffield_tmpa_tl
\l__pdffield_tmpa_keys_tl
\l__pdffield_currentparent_tl
\l__pdffield_fieldID_tl
\l__pdffield_caption_tl
\l__pdffield_rollover_caption_tl
\l__pdffield_down_caption_tl
\g__pdffield_CO_sortkeys_prop
\l__pdffield_CO_sortkey_str
\g__pdffield_annot_ref_last_tl
\l__pdffield_tag_bool

```

```

7 \str_new:N \l__pdffield_tmpa_str
8 \str_new:N \l__pdffield_tmpb_str
9 \tl_new:N \l__pdffield_tmpa_tl
10 \tl_new:N \l__pdffield_tmpa_keys_tl
11 \tl_new:N \l__pdffield_currentparent_tl
12 \tl_new:N \l__pdffield_fieldID_tl
13 \tl_new:N \l__pdffield_caption_tl
14 \tl_new:N \l__pdffield_rollover_caption_tl
15 \tl_new:N \l__pdffield_down_caption_tl
16 \prop_new:N \g__pdffield_CO_sortkeys_prop
17 \seq_new:N \g__pdffield_CO_sortkeys_seq
18 \str_new:N \l__pdffield_CO_sortkey_str
19 \tl_new:N \g__pdffield_annot_ref_last_tl
20 \bool_new:N \l__pdffield_tag_bool
21 \bool_set_true:N \l__pdffield_tag_bool

```

(End of definition for `\l__pdffield_tmpa_str` and others.)

```

22 \cs_new_protected:Npn \__pdffield_tmpa:n #1 {}
23 \cs_new_protected:Npn \__pdffield_tmpa:nn #1 #2 {}

```

7.3 messages

```
24 \msg_new:nnn {pdffield}{no-period}
25 {
26   The~field~name~'#1'~contains~a~period. \\
27   This~is~not~allowed.
28 }
29 \msg_new:nnn {pdffield}{empty-name}
30 {
31   The~field~name~is~empty. \\
32   This~is~not~allowed.
33 }
34 \msg_new:nnn {pdffield}{appearance-missing}
35 {
36   The~appearance~definition~'#1'~is~missing~for~the~#2~appearance.
37 }
38 \msg_new:nnn {pdffield}{not-implemented}
39 {
40   Support~for~'#1'~is~not~implemented\\
41   The~key~is~ignored.
42 }
43 \msg_new:nnn {pdffield}{key-disabled}
44 {
45   key~'#2'~is~disabled~and~ignored~in~the~'#1'~command.\\
46   Use~key~'#3'~instead.
47 }
48 \msg_new:nnn {pdffield}{parent-field-missing}
49 {
50   The~parent~field~'#1'~doesn't~exist\\
51   Create~it~with~\tl_to_str:n{\pdffield_field:nn}
52 }
53 \msg_new:nnn {pdffield}{key-ignored}
54 {
55   key~'#1'~has~no~function~and~is~ignored
56 }
```

An auxiliary command to disable some keys

`_pdffield_key_disable:nnn`

```
57 \cs_new_protected:Npn \_pdffield_key_disable:nnn #1#2#3
58 {
59   \keys_define:nn {pdffield}
60   {
61     #2 .code:n =
62     {
63       \msg_warning:nnnn {pdffield}{key-disabled}{#1}{#2}{#3}
64     }
65   }
66 }
```

(End of definition for _pdffield_key_disable:nnn.)

7.4 bitsets

`\1__pdffield_Ff_bitset` The field and the annot bitset.

`\1__pdffield_F_bitset` 67 \bitset_new:Nn \1__pdffield_Ff_bitset

```

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

```

```

122   LockedContents = 10,
123   invisible      = 1,
124   hidden        = 2,
125   print         = 3,
126   nozoom       = 4,
127   norotate      = 5,
128   noview        = 6,
129   readonly      = 7,
130   locked        = 8,
131   togglenoview = 9,
132   lockedcontents = 10
133 }

```

(End of definition for \l__pdffield_Ff_bitset and \l__pdffield_F_bitset.)

7.5 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.

```

134 \pdfdict_new:n {l__pdffield/field}
135 \pdfdict_new:n {l__pdffield/field/AA}

```

```

\__pdffield_field:n      \__pdffield_field:n{<field ID>}
\pdffield_field:nn
136 \cs_new_protected:Npn \__pdffield_field:n #1
137 {
138   \pdf_object_new:n {__pdffield/field/#1}
139   \pdf_object_new:n {__pdffield/field/Kids/#1}
140   \tl_if_empty:NTF \l__pdffield_currentparent_tl
141     {
142       \pdfmanagement_add:nne
143         { Catalog / AcroForm }
144         { Fields }
145         {\pdf_object_ref:n {__pdffield/field/#1} }
146     }
147     {
148       \exp_args:Ne
149       \pdf_object_if_exist:nTF {__pdffield/field/\l__pdffield_currentparent_tl}
150         {
151           \pdfdict_put:nne { l__pdffield/field }{Parent}
152             {\exp_args:Ne \pdf_object_ref:n{__pdffield/field/\l__pdffield_currentparent_tl}
153               \seq_gput_right:ce {g__pdffield_field/Kids/\l__pdffield_currentparent_tl _seq}
154                 { \exp_args:Ne \pdf_object_ref:n{__pdffield/field/#1}}
155             }
156           {
157             \msg_error:nne {pdffield}{parent-field-missing}{\l__pdffield_currentparent_tl}
158           }
159         }
160       \seq_new:c {g__pdffield_field/Kids/#1_seq}
161       \pdfdict_put:nne {l__pdffield/field}
162         {Kids}
163         {
164           \pdf_object_ref:n {__pdffield/field/Kids/#1}
165         }

```

```

166 \pdfdict_put:nne {l__pdffield/field}
167 {Ff}
168 {\bitset_to_arabic:N \l__pdffield_Ff_bitset }
169 \pdfdict_if_empty:nF{l__pdffield/field/AA}
170 {
171   \pdfmeta_standard_verify:nT
172   {annot_widget_no_AA}
173   {
174     \pdf_object_unnamed_write:ne {dict}{\pdfdict_use:n {l__pdffield/field/AA}}
175     \pdfdict_put:nne
176       {l__pdffield/field}
177       {AA}
178     {\pdf_object_ref_last:}
179     \pdfdict_get:nnN {l__pdffield/field/AA}{C}\l__pdffield_tmpa_tl
180     \quark_if_no_value:NF \l__pdffield_tmpa_tl
181     {
182       \prop_gput:Nee\g__pdffield_CO_sortkeys_prop
183       { \pdf_object_ref:n {__pdffield/field/#1} }{ \l__pdffield_CO_sortkey_str }
184       \seq_gput_right:Ne\g__pdffield_CO_sortkeys_seq
185       { \pdf_object_ref:n {__pdffield/field/#1} }
186     }
187   }
188 }
189 \hook_gput_code:nnn {shipout/lastpage}{pdffield} %xetex needs this ...
190 {
191   \pdf_object_write:nne {__pdffield/field/Kids/#1} { array }
192   {
193     \seq_use:cn{g__pdffield_field/Kids/#1_seq}{~}
194   }
195 }
196 \pdf_object_write:nne {__pdffield/field/#1} { dict } { \pdfdict_use:n {l__pdffield/field}
197 }
198
199 \hook_gput_code:nnn {shipout/lastpage}{pdffield}
200 {
201   \prop_if_empty:NF \g__pdffield_CO_sortkeys_prop
202   {
203     \seq_gsort:Nn \g__pdffield_CO_sortkeys_seq
204     {
205       \str_compare:eNeTF
206         { \prop_item:Nn \g__pdffield_CO_sortkeys_prop {#1} }
207         >
208         { \prop_item:Nn \g__pdffield_CO_sortkeys_prop {#2} }
209         { \sort_return_swapped: }
210         { \sort_return_same: }
211     }
212     \pdfmanagement_add:nne
213     { Catalog / AcroForm }
214     { CO }
215     { \seq_use:Nn \g__pdffield_CO_sortkeys_seq{~} }
216   }
217 }
218
219 \cs_new_protected:Npn \pdffield_field:nn #1 #2

```



```

220 {
221   \group_begin:
222   \keys_set:nn { pdffield } {#1}
223   \__pdffield_field:n {#2}
224   \group_end:
225 }

```

(End of definition for `__pdffield_field:n` and `\pdffield_field:nn`. This function is documented on page 4.)

7.6 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.

`\pdffield_annot:n`

```

226 \cs_new_protected:Npn \__pdffield_annot:
227 {
228   \pdfmeta_standard_verify:nF
229   {annot_flags}
230   {
231     \bitset_set_true:Nn \l__pdffield_F_bitset {Print}
232     \bitset_set_false:Nn \l__pdffield_F_bitset {Hidden}
233     \bitset_set_false:Nn \l__pdffield_F_bitset {Invisible}
234     \bitset_set_false:Nn \l__pdffield_F_bitset {NoView}
235   }
236   \pdfannot_dict_put:nne {widget}{F}{ \bitset_to_arabic:N \l__pdffield_F_bitset }
237   \__pdffield_tag_add_struct_parent:
238   \tl_if_empty:NF \l__pdffield_currentparent_tl
239   {
240     \exp_args:Ne
241     \pdf_object_if_exist:nTF { __pdffield/field/\l__pdffield_currentparent_tl }
242     {
243       \pdfannot_dict_put:nne {widget}{Parent}
244       {
245         \exp_args:Ne
246         \pdf_object_ref:n{__pdffield/field/\l__pdffield_currentparent_tl}
247       }
248     }
249     {
250       \msg_error:nne { pdffield }{parent-field-missing}{\l__pdffield_currentparent_t
251     }
252   }
253   \mode_leave_vertical:
254   \__pdffield_tag_struct_begin:
255   \hbox_to_wd:nn
256   { \l__pdffield_annot_wd_dim }
257   {
258     \rule [-\l__pdffield_annot_dp_dim]{0pt}{\dim_eval:n{\l__pdffield_annot_ht_dim+\l__pd
259     \pdfannot_widget_box:nnn
260     { \l__pdffield_annot_wd_dim }
261     { \l__pdffield_annot_ht_dim }
262     { \l__pdffield_annot_dp_dim }
263     \hfill

```

```

264     }
265     \tl_gset:Ne \g__pdffield_annot_ref_last_tl { \pdfannot_box_ref_last: }
266     \exp_args:NV \__pdffield_tag_add_objr:n \g__pdffield_annot_ref_last_tl
267     \__pdffield_tag_struct_end:
268     \tl_if_empty:NF \l__pdffield_currentparent_tl
269     {
270         \seq_if_exist:cTF {g__pdffield_field/Kids/\l__pdffield_currentparent_tl _seq}
271         {
272             \seq_gput_right:ce
273             {g__pdffield_field/Kids/\l__pdffield_currentparent_tl _seq}
274             { \g__pdffield_annot_ref_last_tl }
275         }
276         {
277             \msg_error:nne { pdffield}{parent-field-missing}{\l__pdffield_currentparent_tl}
278         }
279     }
280 }
281 \cs_new_protected:Npn \pdffield_annot:n #1
282 {
283     \group_begin:
284     \keys_set:nn { pdffield } {#1}
285     \__pdffield_annot:
286     \group_end:
287 }

```

(End of definition for __pdffield_annot: and \pdffield_annot:n. This function is documented on page 4.)

\pdffield_annot_ref_last:

```

288 \cs_new:Npn \pdffield_annot_ref_last: { \g__pdffield_annot_ref_last_tl }

```

(End of definition for \pdffield_annot_ref_last:. This function is documented on page 4.)

7.7 Tagging

```

\__pdffield_tag_add_struct_parent:
\__pdffield_tag_add_objr:n
struct_begin:uuu\__pdffield_tag_struct_end:
289 \cs_new_protected:Npn \__pdffield_tag_add_struct_parent: {}
290 \cs_new_protected:Npn \__pdffield_tag_add_objr:n #1 {}
291 \cs_new_protected:Npn \__pdffield_tag_struct_begin: {}
292 \cs_new_protected:Npn \__pdffield_tag_struct_end: {}
293 \hook_gput_code:nnn {begindocument} { l3pdffield }
294 {
295     \cs_if_exist:NT \tag_if_active:T
296     {
297         \tag_if_active:T
298         {
299             \cs_set_protected:Npn \__pdffield_tag_add_struct_parent:
300             {
301                 \bool_if:NT \l__pdffield_tag_bool
302                 {
303                     \pdfannot_dict_put:nne {widget}{StructParent}{ \tag_struct_parent_int: }
304                 }
305             }
306             \cs_set_protected:Npn \__pdffield_tag_add_objr:n #1

```

```

307     {
308       \bool_if:NT \l__pdffield_tag_bool
309       {
310         \exp_args:Nne
311         \tag_struct_insert_annot:nn {#1}{ \tag_struct_parent_int: }
312       }
313     }
314   \cs_set_protected:Npn \__pdffield_tag_struct_begin:
315   {
316     \bool_if:NT \l__pdffield_tag_bool
317     {
318       \tag_mc_end_push:
319       \tag_struct_begin:n{tag=Form}
320     }
321   }
322   \cs_set_protected:Npn \__pdffield_tag_struct_end:
323   {
324     \bool_if:NT \l__pdffield_tag_bool
325     {
326       \tag_struct_end:
327       \tag_mc_begin_pop:n{ }
328     }
329   }
330 }
331 }
332 }

```

(End of definition for `__pdffield_tag_add_struct_parent:`, `__pdffield_tag_add_objr:n`, and `__pdffield_tag_struct_begin:` `__pdffield_tag_struct_end:.`)

7.8 auxiliary command for color keys

`__pdffield_color_set:nn`

```

333 \cs_new_protected:Npn \__pdffield_color_set:nn #1 #2
334 {
335   \tl_if_head_eq_charcode:nNTF {#2}[ %]
336   {
337     \__pdffield_color_set_aux:nwn { #1 } #2
338   }
339   {
340     \color_set:nn {#1} {#2}
341   }
342 }
343
344 \cs_new_protected:Npn \__pdffield_color_set_aux:nwn #1 [#2] #3
345 {
346   \color_set:nnn {#1}{#2}{#3}
347 }
348

```

(End of definition for `__pdffield_color_set:nn.`)

7.9 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.

```

\__pdffield_V_handler:nN Values (V and DV) need different handling in the various field types. So it uses a handler
which can be redefined locally. By default it simply stores the value in a tl var.
349 \cs_new_protected:Npn \__pdffield_V_handler:nN #1#2
350 {
351   \tl_set:Nn #2 {#1}
352 }

(End of definition for \__pdffield_V_handler:nN.)

parent
T
name
TU
altname
TM
mappingname
353 \keys_define:nn { pdffield }
354 {
355   ,parent .tl_set:N = \l__pdffield_currentparent_tl
356   ,parent .groups:n = {field,annot}
357   ,T .code:n =
358   {
359     \pdf_string_from_unicode:nnN {utf8/string-raw}{#1}\l__pdffield_tmpa_str
360     \str_if_in:NnT \l__pdffield_tmpa_str {.}
361     {
362       \msg_error:nne {pdffield}{no-period}{\l__pdffield_tmpa_str}
363     }
364     \str_if_empty:NTF\l__pdffield_tmpa_str
365     {
366       \msg_warning:nn {pdffield}{empty-name}
367       \pdfdict_remove:nn { l__pdffield/field }{T}
368     }
369     {
370       \pdfdict_put:nne { l__pdffield/field }{T}{(\l__pdffield_tmpa_str)}
371     }
372   }
373   ,T .value_required:n = true
374   ,T .groups:n = {field}
375   ,name .meta:n = {T={#1}}
376   ,name .value_required:n = true
377   ,name .groups:n = {field}
378   ,TU .groups:n = {field}
379   ,TU .code:n =
380   {
381     \tl_if_empty:nTF {#1}
382     {
383       \pdfdict_remove:nn { l__pdffield/field }{TU}
384     }
385     {
386       \pdf_string_from_unicode:nnN {utf16/hex}{#1}\l__pdffield_tmpa_str
387       \pdfdict_put:nne { l__pdffield/field }{TU}{\l__pdffield_tmpa_str}
388     }
389   }
390   ,TU .groups:n = {field}
391   ,altname .meta:n = {TU={#1}}

```

```

392 ,altname .groups:n = {field}
393 ,TM .code:n =
394 {
395   \tl_if_empty:nTF {#1}
396   {
397     \pdfdict_remove:nn { l__pdffield/field }{TM}
398   }
399   {
400     \pdf_string_from_unicode:nnN {utf16/hex}{#1}\l__pdffield_tmpa_str
401     \pdfdict_put:nne { l__pdffield/field }{TM}{\l__pdffield_tmpa_str}
402   }
403 }
404 ,TM .groups:n = {field}
405 ,mappingname .meta:n = {TM={#1}}
406 ,mappingname .groups:n = {field}
407 }

```

(End of definition for parent and others. These functions are documented on page 9.)

fieldID For some field types we need a fieldID.

```

408 \keys_define:nn { pdffield }
409 {
410   fieldID .tl_set:N = \l__pdffield_fieldID_tl
411 }

```

(End of definition for fieldID. This function is documented on page 9.)

```

FT
V
DV
MaxLen
Lock
SV
Opt
TI
I
412 \keys_define:nn{pdffield}
413 {
414   ,FT .choices:nn =
415   { Btn, Tx, Ch, Sig }
416   {
417     \pdfdict_put:nnn { l__pdffield/field }{FT}{ /#1 }
418   }
419   ,FT .groups:n = {field}
420   ,V .code:n =
421   {
422     \tl_if_empty:nTF {#1}
423     {
424       \pdfdict_remove:nn { l__pdffield/field }{V}
425     }
426     {
427       \__pdffield_V_handler:nn{#1}\l__pdffield_tmpa_str
428       \pdfdict_put:nne { l__pdffield/field }{V}{ \l__pdffield_tmpa_str }
429     }
430   }
431   ,V .groups:n = {field}
432   ,DV .code:n =
433   {
434     \tl_if_empty:nTF {#1}
435     {
436       \pdfdict_remove:nn { l__pdffield/field }{DV}
437     }

```

```

438     {
439         \_pdfffield_V_handler:nN{#1}\l__pdfffield_tmpa_str
440         \pdfdict_put:nne { l__pdfffield/field }{DV}{ \l__pdfffield_tmpa_str }
441     }
442 }
443 ,DV .groups:n = {field}
444 ,MaxLen .code:n =
445 {
446     \tl_if_empty:nTF {#1}
447     {
448         \pdfdict_remove:nn { l__pdfffield/field }{MaxLen}
449     }
450     {
451         \pdfdict_put:nne { l__pdfffield/field }{MaxLen}{ #1 }
452     }
453 }
454 ,MaxLen .groups:n = {field}
455 ,Lock .code:n =
456 {
457     \tl_if_empty:nTF {#1}
458     {
459         \pdfdict_remove:nn { l__pdfffield/field }{Lock}
460     }
461     {
462         \pdfdict_put:nne { l__pdfffield/field }{Lock}{ \pdf_object_ref:n{#1} }
463     }
464 }
465 ,Lock .groups:n = {field}
466 ,SV .code:n =
467 {
468     \tl_if_empty:nTF {#1}
469     {
470         \pdfdict_remove:nn { l__pdfffield/field }{SV}
471     }
472     {
473         \pdfdict_put:nne { l__pdfffield/field }{SV}{ \pdf_object_ref:n{#1} }
474     }
475 }
476 ,SV .groups:n = {field}
477 ,Opt .code:n =
478 {
479     \tl_if_empty:nTF {#1}
480     {
481         \pdfdict_remove:nn { l__pdfffield/field }{Opt}
482     }
483     {
484         \pdfdict_put:nne { l__pdfffield/field }{Opt}{ \pdf_object_ref:n{#1} }
485     }
486 }
487 ,Opt .groups:n = {field}
488 ,TI .code:n =
489 {
490     \tl_if_empty:nTF {#1}
491     {

```

```

492         \pdfdict_remove:nn { l__pdffield/field }{TI}
493     }
494     {
495         \pdfdict_put:nne { l__pdffield/field }{TI}{ #1 }
496     }
497 }
498 ,TI .groups:n = {field}
499 ,I .code:n =
500 {
501     \tl_if_empty:nTF {#1}
502     {
503         \pdfdict_remove:nn { l__pdffield/field }{I}
504     }
505     {
506         \pdfdict_put:nne { l__pdffield/field }{I}{ \pdf_object_ref:n{#1} }
507     }
508 }
509 ,I .groups:n = {field}
510 }

```

(End of definition for FT and others. These functions are documented on page 7.)

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

```

setFf
setfieldflags 511 \keys_define:nn { pdffield }
unsetFf        512 {
unsetfieldflags 513     ,setFf .code:n =
514     {
515         \clist_map_inline:nn {#1}
516         {
517             \bitset_set_true:Nn \l__pdffield_Ff_bitset {##1}
518         }
519     }
520     ,setFf .groups:n = {field}
521     ,setfieldflags .meta:n =
522     {setFf={#1}}
523     ,setfieldflags .groups:n = {field}
524     ,unsetFf .multichoice:
525     ,unsetFf / all .code:n = { \bitset_clear:N \l__pdffield_Ff_bitset}
526     ,unsetFf / unknown .code:n =
527     {
528         \bitset_set_false:Nn \l__pdffield_Ff_bitset {#1}
529     }
530     ,unsetFf .groups:n = {field}
531     ,unsetfieldflags .meta:n = {unsetFf={#1}}
532     ,unsetfieldflags .groups:n = {field}
533 }
534

```

(End of definition for setFf and others. These functions are documented on page 7.)

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

```

AA/F 535 \cs_set_protected:Npn \__pdffield_tmpa:n #1 %
format 536 {

```

AA/V
validate
AA/C
calculate

```

537 \keys_define:nn { pdfffield }
538 {
539   AA/#1 .code:n =
540   {
541     \tl_if_empty:nTF {#1}
542     {
543       \pdfdict_remove:nn {l__pdfffield/field/AA}{#1}
544     }
545     {
546       \pdfdict_put:nne {l__pdfffield/field/AA}
547       {#1}
548       {<</S/JavaScript/JS\c_space_tl #1>>}
549     }
550   },
551   AA/#1 .groups:n = {field}
552 }
553 }
554
555 \clist_map_inline:nn {K,F,V,C}{\__pdfffield_tmpa:n{#1}}
556
557 \cs_set_protected:Npn \__pdfffield_tmpa:nn #1 #2
558 {
559   \keys_define:nn { pdfffield }
560   {
561     #1 .meta:nn =
562     { pdfffield }{AA/#2={#1}},
563     #1 .groups:n = {field}
564   }
565 }
566 \__pdfffield_tmpa:nn {keystroke}{K}
567 \__pdfffield_tmpa:nn {format} {F}
568 \__pdfffield_tmpa:nn {validate} {V}
569 \__pdfffield_tmpa:nn {calculate}{C}
570
571 \keys_define:nn {pdfffield}
572 {
573   sortkey .code:n = {\str_set:Ne \l__pdfffield_CO_sortkey_str {\tl_to_str:n{#1}}}
574 }

```

(End of definition for AA/K and others. These functions are documented on page 8.)

DA The following keys are related to textfield and their format.

Q

align

DS

RV

```

575 \keys_define:nn { pdfffield }
576 {
577   DA .code:n =
578   {
579     \tl_if_empty:nTF {#1}
580     {
581       \pdfdict_remove:nn { l__pdfffield/field }{DA}
582     }
583     {
584       \pdfdict_put:nne { l__pdfffield/field }{DA}{ (#1) }
585     }
586   }

```



```

587 ,DA .groups:n = {field}
588 ,Q .choices:nn = {left,center,right}
589 {
590   \pdfdict_put:nne { l__pdffield/field }{Q}{ \int_eval:n{\l_keys_choice_int-1} }
591 }
592 ,Q / .code:n = { \pdfdict_remove:nn { l__pdffield/field }{Q} }
593 ,Q .groups:n = {field}
594 ,align .meta:n={Q=#1}
595 ,DS .code:n =
596 {
597   \msg_warning:nnn {pdffield}{not-implemented}{DS}
598 }
599 ,DS .groups:n = {field}
600 ,RV .code:n =
601 {
602   \msg_warning:nnn {pdffield}{not-implemented}{RV}
603 }
604 ,RV .groups:n = {field}
605 }

```

(End of definition for DA and others. These functions are documented on page 9.)

7.10 Annotation keys

The size of the field annotation

```

\l__pdffield_annot_ht_dim
\l__pdffield_annot_wd_dim
\l__pdffield_annot_dp_dim
606 \dim_new:N \l__pdffield_annot_ht_dim
607 \dim_new:N \l__pdffield_annot_wd_dim
608 \dim_new:N \l__pdffield_annot_dp_dim

```

(End of definition for \l__pdffield_annot_ht_dim, \l__pdffield_annot_wd_dim, and \l__pdffield_annot_dp_dim.)

```

width The size of the field annotation.
height
depth
609 \keys_define:nn { pdffield }
610 {
611   ,width .dim_set:N = \l__pdffield_annot_wd_dim
612   ,height .dim_set:N = \l__pdffield_annot_ht_dim
613   ,depth .dim_set:N = \l__pdffield_annot_dp_dim
614   ,width .initial:n = Opt
615   ,height .initial:n = Opt
616   ,depth .initial:n = Opt
617 }

```

(End of definition for width, height, and depth. These functions are documented on page 9.)

```

tag to disable tagging locally
618 \keys_define:nn { pdffield }
619 {
620   ,tag .bool_set:N = \l__pdffield_tag_bool
621 }

```

(End of definition for tag. This function is documented on page 9.)

`_pdffield_appearance_handler:nnn` Appearances have to be handled in various ways, so we use a handler, that the field types can redefine if needed.

```

622 \cs_new_protected:Npn \_pdffield_appearance_handler:nnn #1#2#3
623 {
624   \pdfxform_if_exist:nTF { #1 }
625   {
626     \pdfannot_dict_put:nne {widget/AP}{#2}
627     {
628       \pdfxform_ref:n {#1}
629     }
630   }
631   {
632     \msg_error:nnnn{pdffield}{appearance-missing}{#1}{#3}
633   }
634 }

```

(End of definition for _pdffield_appearance_handler:nnn.)

```

AS The key for the default appearance and the various types.
AP/N
appearance
AP/R
rollover-appearance
AP/D
down-appearance
635 \keys_define:nn { pdffield }
636 {
637   %parent is defined in field
638   ,AS .code:n =
639   {
640     \tl_if_empty:nTF {#1}
641     {
642       \pdfannot_dict_remove:nn { widget }{AS}
643     }
644     {
645       \pdfannot_dict_put:nne {widget}{AS}{\pdf_name_from_unicode_e:n{#1}}
646     }
647   }
648   ,AS .groups:n = annot
649 }
650 \keys_define:nn { pdffield }
651 {
652   AP/N .code:n =
653   {
654     \tl_if_empty:nTF {#1}
655     {
656       \pdfannot_dict_remove:nn { widget/AP }{N}
657     }
658     {
659       \_pdffield_appearance_handler:nnn {#1}{N}{normal}
660     }
661   }
662   ,AP/N .groups:n = annot
663   ,appearance .meta:n = {AP/N={#1}}
664   ,appearance .groups:n = annot
665 }
666 \keys_define:nn { pdffield }
667 {
668   AP/R .code:n =
669   {

```

```

670     \tl_if_empty:nTF {#1}
671     {
672       \pdfannot_dict_remove:nn { widget/AP }{R}
673     }
674     {
675       \__pdfffield_appearance_handler:nnn {#1}{R}{rollover}
676     }
677   }
678   ,AP/R .groups:n = annot
679   ,rollover-appearance .meta:n = {AP/R={#1}}
680   ,rollover-appearance .groups:n = annot
681 }
682 \keys_define:nn { pdfffield }
683 {
684   AP/D .code:n =
685   {
686     \tl_if_empty:nTF {#1}
687     {
688       \pdfannot_dict_remove:nn { widget/AP }{D}
689     }
690     {
691       \__pdfffield_appearance_handler:nnn {#1}{D}{down}
692     }
693   }
694   ,AP/D .groups:n = annot
695   ,down-appearance .meta:n = {AP/D={#1}}
696   ,down-appearance .groups:n = annot
697 }

```

(End of definition for AS and others. These functions are documented on page 10.)

```

MK/R This are the keys for the dynamic appearance. A number are not handled yet fully.
rotate
MK/BC
bordercolor
MK/BG
backgroundcolor
MK/CA
caption
698 \keys_define:nn { pdfffield }
699 {
700   MK/R .choices:nn = {0,90,180,270}
701   {
702     \pdfannot_dict_put:nne {widget/MK}{R}{#1}
703   }
704   ,MK/R / .code:n =
705   {
706     \pdfannot_dict_remove:nn { widget/MK }{R}
707   }
708   ,MK/R .groups:n = annot
709   ,rotate .meta:n = {MK/R=#1}
710 }
711
712 \keys_define:nn { pdfffield }
713 {
714   MK/BC .code:n =
715   {
716     \tl_if_empty:nTF {#1}
717     {
718       \pdfannot_dict_remove:nn { widget/MK }{BC}
719     }

```

```

720     {
721     \_pdfffield_color_set:nn {__pdfffield/tmp}{#1}
722     \color_export:nnN{__pdfffield/tmp}{space-sep-rgb}\l__pdfffield_tmpa_tl
723     \pdfannot_dict_put:nne {widget/MK}{BC}{[\l__pdfffield_tmpa_tl]}
724     }
725   }
726   ,MK/BC .groups:n = annot
727   ,bordercolor .meta:n = {MK/BC=#1}
728 }
729
730 \keys_define:nn { pdfffield }
731 {
732   MK/BG .code:n =
733   {
734     \tl_if_empty:nTF {#1}
735     {
736       \pdfannot_dict_remove:nn { widget/MK }{BG}
737     }
738     {
739       \_pdfffield_color_set:nn {__pdfffield/tmp}{#1}
740       \color_export:nnN{__pdfffield/tmp}{space-sep-rgb}\l__pdfffield_tmpa_tl
741       \pdfannot_dict_put:nne {widget/MK}{BG}{[\l__pdfffield_tmpa_tl]}
742     }
743   }
744   ,MK/BG .groups:n = annot
745   ,backgroundcolor .meta:n = {MK/BG=#1}
746 }
747
748
749 \keys_define:nn { pdfffield }
750 {
751   MK/CA .code:n =
752   {
753     \tl_set:Nn \l__pdfffield_caption_tl {#1}
754     \tl_if_empty:nTF {#1}
755     {
756       \pdfannot_dict_remove:nn { widget/MK }{CA}
757     }
758     {
759       \pdf_string_from_unicode:nnN {utf8/string}{#1}\l__pdfffield_tmpa_str
760       \pdfannot_dict_put:nne {widget/MK}{CA}{[\l__pdfffield_tmpa_str]}
761     }
762   }
763   ,MK/CA .groups:n = annot
764   ,caption .meta:n = {MK/CA=#1}
765 }
766
767 \keys_define:nn { pdfffield }
768 {
769   MK/RC .code:n =
770   {
771     \tl_set:Nn \l__pdfffield_rollover_caption_tl {#1}
772     \tl_if_empty:nTF {#1}
773     {

```

```

774     \pdfannot_dict_remove:nn { widget/MK }{RC}
775   }
776   {
777     \pdf_string_from_unicode:nnN {utf8/string}{#1}\l__pdffield_tmpa_str
778     \pdfannot_dict_put:nne {widget/MK}{RC}{\l__pdffield_tmpa_str}
779   }
780 }
781 ,MK/RC .groups:n = annot
782 ,rollover-caption .meta:n = {MK/RC=#1}
783 }
784
785 \keys_define:nn { pdffield }
786 {
787   MK/AC .code:n =
788   {
789     \tl_set:Nn \l__pdffield_down_caption_tl {#1}
790     \tl_if_empty:nTF {#1}
791     {
792       \pdfannot_dict_remove:nn { widget/MK }{AC}
793     }
794     {
795       \pdf_string_from_unicode:nnN {utf8/string}{#1}\l__pdffield_tmpa_str
796       \pdfannot_dict_put:nne {widget/MK}{AC}{\l__pdffield_tmpa_str}
797     }
798   }
799   ,MK/AC .groups:n = annot
800   ,down-caption .meta:n = {MK/AC=#1}
801 }

```

(End of definition for MK/R and others. These functions are documented on page 11.)

MK/I The following keys are pushbuttons only. Currently there is no special handling involved
MK/RI as it is unclear if they are useful.

MK/IX

MK/IF

MK/TP

```

802 \cs_set_protected:Npn \__pdffield_tmpa:n #1
803 {
804   \keys_define:nn { pdffield }
805   {
806     MK/#1 .code:n =
807     {
808       \tl_if_empty:nTF {##1}
809       {
810         \pdfannot_dict_remove:nn { widget/MK }{#1}
811       }
812       {
813         \pdfannot_dict_put:nne {widget/MK}{#1}{##1}
814       }
815     }
816     ,MK/#1 .groups:n = annot
817   }
818 }
819 }
820
821 \clist_map_inline:nn {I,RI,IX,IF,TP}
822 { \__pdffield_tmpa:n {#1} }

```

(End of definition for MK/I and others. These functions are documented on page 12.)

Flags.

```

setF
setannotflags 823 \keys_define:nn { pdfffield }
unsetF         824 {
unsetannotflags 825   ,setF .code:n =
                826   {
                827     \clist_map_inline:nn {#1}
                828     {
                829       \bitset_set_true:Nn \l__pdfffield_F_bitset {##1}
                830     }
                831   }
                832   ,setF .groups:n = annot
                833   ,setannotflags .meta:nn =
                834     { pdfffield }{setF={#1}}
                835   ,setannotflags .groups:n = annot
                836   ,unsetF .multichoice:
                837   ,unsetF / all .code:n = { \bitset_clear:N \l__pdfffield_F_bitset}
                838   ,unsetF / unknown .code:n =
                839     {
                840       \bitset_set_false:Nn \l__pdfffield_F_bitset {#1}
                841     }
                842   ,unsetF .groups:n = annot
                843   ,unsetannotflags .meta:nn =
                844     { pdfffield }{unsetF= {#1} }
                845   ,unsetannotflags .groups:n = annot
                846 }
                847
```

(End of definition for setF and others. These functions are documented on page 10.)

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

```

AA/Fo
onfocus 848 \cs_set_protected:Npn \__pdfffield_tmpa:n #1 %
AA/Bl    849 {
onblur   850   \keys_define:nn { pdfffield }
AA/D     851   {
onmousedown 852     AA/#1 .code:n =
AA/U     853     {
onmouseup  854       \tl_if_empty:nTF {#1}
AA/E     855       {
onenter   856         \pdfannot_dict_remove:nn {widget/AA}{#1}
AA/X     857       }
AA/X     858       {
onexit    859         \pdfannot_dict_put:nme {widget/AA}
AA/PO    860         {#1}
pageopen  861         {<</S/JavaScript/JS\c_space_tl##1>>}
AA/PC    862       }
pageclose 863     },
AA/PV    864     ,AA/#1 .groups:n = annot
pagevisible 865   }
AA/PI
pageinvisible
```

```

866 }
867
868 \clist_map_inline:nn {Fo,Bl,D,U,E,X,PO,PC,PV,PI}{\_pdfffield_tmpa:n{#1}}
869
870 \cs_set_protected:Npn \_pdfffield_tmpa:nn #1 #2
871 {
872   \keys_define:nn { pdfffield }
873   {
874     #1 .meta:nn =
875     { pdfffield }{AA/#2={##1}},
876     #1 .groups:n = {annot}
877   }
878 }
879 \_pdfffield_tmpa:nn {onfocus} {Fo}
880 \_pdfffield_tmpa:nn {onblur} {Bl}
881 \_pdfffield_tmpa:nn {onmousedown}{D}
882 \_pdfffield_tmpa:nn {onmouseup}{U}
883 \_pdfffield_tmpa:nn {onenter} {E}
884 \_pdfffield_tmpa:nn {onexit} {X}

```

(End of definition for AA/Fo and others. These functions are documented on page ??.)

7.11 Appearances

`\pdfffield_appearance:nn`

`\pdfffield_store_appearance:nn`

```

885 \cs_new_protected:Npn \pdfffield_appearance:nn #1 #2
886 {
887   \pdfxform_new:nnn {#1}{#2}
888 }
889
890 \cs_set_eq:NN \pdfffield_store_appearance:nn\pdfffield_appearance:nn

```

(End of definition for `\pdfffield_appearance:nn` and `\pdfffield_store_appearance:nn`. These functions are documented on page 4.)

7.12 Setup command

`create-style`
`preset-checkbox`
`preset-radio`
`preset-textfield`

```

891 \keys_define:nn { pdfffield / setup }
892 {
893   ,create-style .code:n = { \_pdfffield_style_create:nn #1 }
894   ,preset-checkbox .code:n =
895   {
896     \keys_define:nn { pdfffield }
897     {
898       __pdfffield/preset/checkbox .meta:n = {#1},
899     }
900   }
901   ,preset-radiobutton .code:n =
902   {
903     \keys_define:nn { pdfffield }
904     {
905       __pdfffield/preset/radiobutton .meta:n = {#1},
906     }

```

```

907     }
908     ,preset-textfield .code:n =
909     {
910         \keys_define:nn { pdfffield }
911         {
912             __pdfffield/preset/textfield .meta:n = {#1},
913         }
914     }
915     ,preset-pushbutton .code:n =
916     {
917         \keys_define:nn { pdfffield }
918         {
919             __pdfffield/preset/pushbutton .meta:n = {#1},
920         }
921     }
922     ,preset-choice .code:n =
923     {
924         \keys_define:nn { pdfffield }
925         {
926             __pdfffield/preset/choice .meta:n = {#1},
927         }
928     }
929 }
930 \keys_set:nn{ pdfffield / setup }{preset-checkbox={}}
931 \keys_set:nn{ pdfffield / setup }{preset-textfield={}}
932 \keys_set:nn{ pdfffield / setup }{preset-radiobutton={}}
933 \keys_set:nn{ pdfffield / setup }{preset-pushbutton={}}
934 \keys_set:nn{ pdfffield / setup }{preset-choice={}}

```

(End of definition for create-style and others. These functions are documented on page 5.)

`__pdfffield_style_create:nn`

```

935 \cs_new_protected:Npn \__pdfffield_style_create:nn #1#2
936 {
937     \keys_define:nn { pdfffield }
938     {
939         __pdfffield/style/#1 .meta:n = {#2},
940     }
941 }
942

```

(End of definition for __pdfffield_style_create:nn.)

`\pdfffield_setup:n`
`style`

```

943 \cs_new_protected:Npn \pdfffield_setup:n #1
944 {
945     \keys_set:nn{ pdfffield / setup }{#1}
946 }
947
948 \keys_define:nn { pdfffield }
949 {
950     style .code:n = {\keys_set:nn {pdfffield}{__pdfffield/style/#1={#1}}}
951 }

```

(End of definition for \pdfffield_setup:n and style. These functions are documented on page 5.)

8 Value keys

value
default

```

952 \cs_new_protected:Npn \__pdffield_value_handler:n #1
953 {
954   \msg_info:nnn {pdffield}{key-ignored}{value}
955 }
956 \cs_new_protected:Npn \__pdffield_default_handler:n #1
957 {
958   \msg_info:nnn {pdffield}{key-ignored}{default}
959 }
960 \keys_define:mn {pdffield}
961 {
962   value .code:n = { \__pdffield_value_handler:n {#1} }
963   ,default .code:n = { \__pdffield_default_handler:n {#1}}
964 }

```

(End of definition for value and others. These functions are documented on page 5.)

```
965 </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.

A		bool commands:	
AA/*	11	\bool_if:NTF	301, 308, 316, 324
AA/BI	848	\bool_set_true:N	21
AA/C	8, <u>535</u>	bordercolor	11, <u>698</u>
AA/D	848	C	
AA/E	848	calculate	8, <u>535</u>
AA/F	8, <u>535</u>	caption	11, <u>698</u>
AA/Fo	848	create-style	5, <u>891</u>
AA/K	8, <u>535</u>	D	
AA/PC	848	DA	9, <u>575</u>
AA/PI	848	default	5, <u>952</u>
AA/PO	848	depth	9, <u>609</u>
AA/PV	848	down-appearance	10, <u>635</u>
AA/U	848	down-caption	12
AA/V	8, <u>535</u>	DS	9, <u>575</u>
AA/X	848	DV	7, <u>412</u>
align	9, <u>575</u>	F	
altname	7, <u>353</u>	fieldID	9, <u>408</u>
AP/D	10, <u>635</u>	\Form	2
AP/N	10, <u>635</u>	format	8, <u>535</u>
AP/R	10, <u>635</u>	FT	7, <u>412</u>
appearance	10, <u>635</u>	H	
AS	10, <u>635</u>	height	9, <u>609</u>
B			
backgroundcolor	11, <u>698</u>		

	I		
I	8, 412	
	K		
keystroke	8, 535	
	L		
Lock	7, 412	
	M		
mappingname	7, 353	
MaxLen	7, 412	
MK/AC	12	
MK/BC	11, 698	
MK/BG	11, 698	
MK/CA	11, 698	
MK/I	12, 802	
MK/IF	12, 802	
MK/IX	12, 802	
MK/R	11, 698	
MK/RC	11	
MK/RI	12, 802	
MK/TP	12, 802	
	N		
name	5, 353	
	O		
onblur	848	
onenter	848	
onexit	848	
onfocus	848	
onmousedown	848	
onmouseup	848	
Opt	8, 412	
	P		
pageclose	848	
pageinvisible	848	
pageopen	848	
pagevisible	848	
parent	5, 9, 353	
pdf commands:			
\pdf_string_from_unicode:nnN	5, 7, 11, 12	
pdfannot commands:			
\pdfannot_widget_box:nnn	4	
pdfdict commands:			
\pdfdict_get:nnN	179	
pdffield commands:			
\pdffield_annot:n	4, 226 , 281	
\pdffield_annot_ref_last:	4, 288 , 288	
\pdffield_appearance:nn	4, 885 , 885 , 890	
\pdffield_field:nn	4, 51, 136 , 219	
\pdffield_setup:n	5, 943 , 943	
\pdffield_store_appearance:nn	885 , 890	
pdffield internal commands:			
_pdffield_annot:	226 , 226 , 285	
\l_pdffield_annot_dp_dim	258, 262 , 606 , 613	
\l_pdffield_annot_ht_dim	258, 261 , 606 , 612	
\g_pdffield_annot_ref_last_tl	7, 265 , 266 , 274 , 288	
\l_pdffield_annot_wd_dim	256, 260 , 606 , 611	
_pdffield_appearance_handler:nnn	622 , 622 , 659 , 675 , 691	
\l_pdffield_caption_tl	7, 753	
\l_pdffield_CO_sortkey_str	7, 183 , 573	
\g_pdffield_CO_sortkeys_prop	7, 182 , 201 , 206 , 208	
\g_pdffield_CO_sortkeys_seq	17, 184 , 203 , 215	
_pdffield_color_set:nn	333 , 333 , 721 , 739	
_pdffield_color_set_aux:nwn	337 , 344	
\l_pdffield_currentparent_tl	7, 140 , 149 , 152 , 153 , 157 , 238 , 241 , 246 , 250 , 268 , 270 , 273 , 277 , 355	
_pdffield_default_handler:n	952 , 956 , 963	
\l_pdffield_down_caption_tl	7, 789	
\l_pdffield_F_bitset	67 , 231 , 232 , 233 , 234 , 236 , 829 , 837 , 840	
\l_pdffield_Ff_bitset	67 , 168 , 517 , 525 , 528	
_pdffield_field:n	15, 136 , 136 , 223	
\l_pdffield_fieldID_tl	7, 410	
_pdffield_key_disable:nnn	57 , 57	
\l_pdffield_rollover_caption_tl	7, 771	
_pdffield_style_create:nn	893 , 935 , 935	
_pdffield_tag_add_objr:n	266 , 289 , 290 , 306	
_pdffield_tag_add_struct_-parent:	237 , 289 , 289 , 299	
\l_pdffield_tag_bool	7, 301 , 308 , 316 , 324 , 620	
_pdffield_tag_struct_begin:	254 , 291 , 314	
_pdffield_tag_struct_begin:_-pdffield_tag_struct_end:	289	

