The l3pdffield module
Commands to create form fields
LaTeX PDF management testphase bundle

The LaTeX Project[*]

Version 0.95d, released 2021-05-14

1 l3pdffield Introduction

The implementation of form fields in hyperref has some bugs[1]. 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{l3pdffield-testphase}
```

The code is splitted into various submodules. `l3pdffield` contains the basic commands to create a form field. The code related to field types like checkboxes are in `l3pdffield-`*`type`*, for example `l3pdffield-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 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).

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

[1]see 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

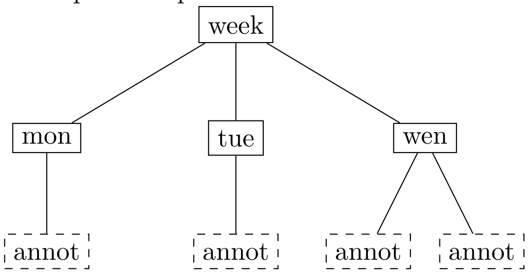
```
\RequirePackage{pdfmanagement-testphase}
\DeclareDocumentMetadata{uncompress}
\documentclass{article}
\usepackage{hyperref}
\usepackage{l3pdffield-testphase}
\begin{document}
\Form
```

2 Some background

A document can contain 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 superseed 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 XObjects [2] , 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.

3 Commands

`\pdffield_field:nn`
`\pdffield_field:Vn`

`\pdffield_field:nn{⟨key val list⟩}{⟨field ID⟩}`

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>}
```

[2]Such 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.

`\pdffield_annot:n`
`\pdffield_annot:V`

`\pdffield_field:nn{⟨key val list⟩}`

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_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`.

`\pdffield_setup:n`

`\pdffield_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.

`preset-checkbox`

`preset-checkbox={⟨key-val⟩}`

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 is 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`
`T`

`name = ⟨partial name⟩`
`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`.

`altname`
`TU`

`altname = ⟨string⟩`
`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`

Table 1: Keys for fields

`key`	value	required	inheritable	remark
`parent`	field ID	for non-root fields		
`style`	style name		defined with `create-style`	
`T, name`	string	mostly		
`TU, altname`	string			
`TM, mappingname`	string			
`FT`	name	terminal fields	yes	
`setFf, setfieldflags`	list of flags		yes	
`unsetFf, unsetfieldflags`	list of flags		yes	
`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 cho
`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)

`mappingname` `mappingname = ⟨string⟩`
`TM` `TM = ⟨string⟩`

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

`mappingname` `FT = Btn|Tx|Ch|Sig`
`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.

`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 currently only passed through `\pdf_string_from_unicode:nnN`, but this perhaps will have to change. The keys will be ignored if a pdfstandard is used that prohibits such actions.

`AA/K` `keystroke` `AA/K = ⟨string (ECMAScript)⟩`
`keystroke = ⟨string (ECMAScript)⟩`

This adds a keystroke action to the additional action dictionary. The value is passed through `\pdf_string_from_unicode:nnN`. 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` `format` `AA/F = ⟨string (ECMAScript)⟩`
`format = ⟨string (ECMAScript)⟩`

This adds a format action to the additional action dictionary. The value is passed through `\pdf_string_from_unicode:nnN`. 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` `validate` `AA/V = ⟨string (ECMAScript)⟩`
`validate = ⟨string (ECMAScript)⟩`

This adds a validate action to the additional action dictionary. The value is passed through `\pdf_string_from_unicode:nnN`. It is quite unclear if such an action make sense for non-terminal fields.

`AA/C` `calculate` `AA/C = ⟨string (ECMAScript)⟩`
`calculate = ⟨string (ECMAScript)⟩`

This adds a calculate action to the additional action dictionary. The value is passed through `\pdf_string_from_unicode:nnN`. It is quite unclear if such an action make sense for non-terminal fields.

`DA` `DA = ⟨string⟩`

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

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 l3pdffield-checkbox change this to allow to add appearances for various states.

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 = ⟨*comma list of flags*⟩
setF setF = ⟨*comma list of flags*⟩
unsetannotflags unsetannotflags = all | ⟨*comma list of flags*⟩
unsetF unsetF = all | ⟨*comma list of flags*⟩

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/* = ⟨*string (ECMAScript)*⟩

* should be one of `F`, `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 `/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 = ⟨*color expression*⟩ | [⟨*model*⟩]{⟨*values*⟩}
bordercolor bordercolor = ⟨*color expression*⟩ | [⟨*model*⟩]{⟨*values*⟩}

These colors the border. Internally currently RGB is used. The colors used in ⟨*color expression*⟩ must be known to the l3color commands.

MK/BG MK/BG = ⟨*color expression*⟩ | [⟨*model*⟩]{⟨*values*⟩}
backgroundcolor backgroundcolor = ⟨*color expression*⟩ | [⟨*model*⟩]{⟨*values*⟩}

These colors the background. Internally currently RGB is used. The colors used in ⟨*color expression*⟩ must be known to the l3color commands.

MK/CA
caption

`MK/CA = ⟨string⟩`
`caption = ⟨string⟩`

This sets a text for the caption. ⟨*string*⟩ 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/* = ⟨various⟩`

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

```
⟨*package⟩
⟨@@=pdffield⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesExplPackage{l3pdffield-testphase}{2021-05-14}{0.95d}%
  {form fields}
```

6.1 hyperref specific command

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

```
\csname HyField@NeedAppearancesfalse\endcsname % suppress NeedAppearances
```

6.2 local variables

```
\str_new:N \l__pdffield_tmpa_str
\tl_new:N  \l__pdffield_tmpa_tl
\tl_new:N  \l__pdffield_tmpa_keys_tl
\cs_new_protected:Npn \__pdffield_tmpa:n #1 {}
\cs_new_protected:Npn \__pdffield_tmpa:nn #1 #2 {}
\tl_new:N \l__pdffield_currentparent_tl
\tl_new:N \l__pdffield_fieldID_tl
```

6.3 messages

```
\msg_new:nnn {pdffield}{no-period}
  {
    The~field~name~‘#1‘~contains~a~period. \\
    This~is~not~allowed. ‘
  }
\msg_new:nnn {pdffield}{empty-name}
  {
    The~field~name~is~empty. \\
    This~is~not~allowed. ‘
  }
\msg_new:nnn {pdffield}{appearance-missing}
  {
    The~appearance~definition~‘#1‘~is~missing~for~the~#2~appearance.
  }
\msg_new:nnn {pdffield}{not-implemented}
  {
    Support~for~’/#1’~is~not~implemented\\
    The~key~is~ignored.
  }
\msg_new:nnn {pdffield}{key-disabled}
  {
    key~‘#2‘~is~disabled~and~ignored~in~the~‘#1‘~command.\\
    Use~key~‘#3‘~instead.
  }
\msg_new:nnn {pdffield}{parent-field-missing}
  {
    The~parent~field~’#1’~doesn’t~exist\\
    Create~it~with~\tl_to_str:n{\pdffield_field:nn}
  }
```

An auxiliary command to disable some keys

__pdffield_key_disable:nnn

```
\cs_new_protected:Npn \__pdffield_key_disable:nnn #1#2#3
 {
   \keys_define:nn {pdffield}
    {
      #2 .code:n =
       {
         \msg_warning:nnnnn {pdffield}{key-disabled}{#1}{#2}{#3}
       }
    }
 }
```

(*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.

```
\bitset_new:Nn \l__pdffield_Ff_bitset
 {
    ReadOnly          = 1,
    Required          = 2,
    NoExport          = 3,
    Multiline         = 13,%Tx
    Password          = 14,
    NoToggleToOff     = 15,%Btn, radio button
    Radio             = 16,%Btn: Radio:    15=1, 16=0
    Pushbutton        = 17,%Btn: Checkbox: 15=0, 16=0
                           %Btn: Pushbutton: 16=1
    Combo             = 18,%Ch: Combo=1 List=0
    Edit              = 19,%Ch, Combo=1 -> + edit field
    Sort              = 20,%Ch, not relevant for view...
    FileSelect        = 21,%Tx
    MultiSelect       = 22,%Ch
    DoNotSpellCheck   = 23,%Tx, Ch (if Combo + Edit set)
    DoNotScroll       = 24,%Tx
    Comb              = 25,%Tx, requires MaxLen in dict
    RadiosInUnison    = 26,%Btn Radio
    RichText          = 26,%Tx
    CommitOnSelChange = 27,
    readonly          = 1,
    required          = 2,
    noexport          = 3,
    multiline         = 13,%Tx
    password          = 14,
    notoggletooff     = 15,%Btn, radio button
    radio             = 16,%Btn: Radio:    15=1, 16=0
    pushbutton        = 17,%Btn: Checkbox: 15=0, 16=0
                           %Btn: Pushbutton: 16=1
    combo             = 18,%Ch: Combo=1 List=0
    edit              = 19,%Ch, Combo=1 -> + edit field
    sort              = 20,%Ch, not relevant for view...
    fileselect        = 21,%Tx
    multiselect       = 22,%Ch
    donotspellcheck   = 23,%Tx, Ch (if Combo + Edit set)
    donotscroll       = 24,%Tx
    comb              = 25,%Tx, requires MaxLen in dict
    radiosinunison    = 26,%Btn Radio
    richtext          = 26,%Tx
    commitonselchange = 27
  }

\bitset_new:Nn \l__pdffield_F_bitset
  {
    Invisible      = 1,
    Hidden         = 2,
    Print          = 3,
    NoZoom         = 4,
    NoRotate       = 5,
    NoView         = 6,
    ReadOnly       = 7,
    Locked         = 8,
    ToggleNoView   = 9,
    LockedContents = 10,
    invisible      = 1,
    hidden         = 2,
    print          = 3,
    nozoom         = 4,
    norotate       = 5,
    noview         = 6,
    readonly       = 7,
    locked         = 8,
    togglenoview   = 9,
    lockedcontents = 10
  }
```

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.

```
\pdfdict_new:n   {l__pdffield/field}
\pdfdict_new:n   {l__pdffield/field/AA}
\bool_new:N \l__pdffield_root_field_bool
```

__pdffield_field:n

`\__pdffield_field:n{⟨field ID⟩}`

```
\cs_new_protected:Npn \__pdffield_field:n #1
 {
   \pdf_object_new:nn {__pdffield/field/#1}      {dict}
   \pdf_object_new:nn {__pdffield/field/Kids/#1} {array}
   \tl_if_empty:NTF \l__pdffield_currentparent_tl
     {
       \pdfmanagement_add:nnx
         { Catalog / AcroForm }
         { Fields }
         {\pdf_object_ref:n {__pdffield/field/#1} }
     }
     {
       \exp_args:Ne
       \pdf_object_if_exist:nTF {__pdffield/field/\l__pdffield_currentparent_tl}
         {
           \pdfdict_put:nnx { l__pdffield/field }{Parent}
             {\exp_args:Ne \pdf_object_ref:n{__pdffield/field/\l__pdffield_currentparent_tl}
           \seq_gput_right:cx {g__pdffield_field/Kids/\l__pdffield_currentparent_tl _seq}
             { \exp_args:Ne \pdf_object_ref:n{__pdffield/field/#1}}
         }
         {
           \msg_error:nnx {pdffield}{parent-field-missing}{\l__pdffield_currentparent_tl}
         }
     }
   \seq_new:c {g__pdffield_field/Kids/#1_seq}
   \pdfdict_put:nnx {l__pdffield/field}
     {Kids}
     {
       \pdf_object_ref:n {__pdffield/field/Kids/#1}
     }
   \pdfdict_put:nnx {l__pdffield/field}
     {Ff}
     {\bitset_to_arabic:N \l__pdffield_Ff_bitset }
   \pdfdict_if_empty:nF{l__pdffield/field/AA}
     {
       \pdfmeta_standard_verify:nT
         {annot_widget_no_AA}
         {
           \pdf_object_unnamed_write:nx {dict}{\pdfdict_use:n {l__pdffield/field/AA}}
           \pdfdict_put:nnx
             {l__pdffield/field}
             {AA}
             {\pdf_object_ref_last:}
         }
     }
```

```
    \hook_gput_code:nnn {shipout/lastpage}{pdffield} %xetex needs this ...
      {
        \pdf_object_write:nx {__pdffield/field/Kids/#1}
          {
            \seq_use:cn{g__pdffield_field/Kids/#1_seq}{~}
          }
      }
    \pdf_object_write:nx {__pdffield/field/#1} { \pdfdict_use:n {l__pdffield/field} }
  }
\cs_new_protected:Npn \pdffield_field:nn #1 #2
  {
    \group_begin:
    \keys_set:nn { pdffield } {#1}
    \__pdffield_field:n {#2}
    \group_end:
  }
```

(*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.

```
\cs_new_protected:Npn \__pdffield_annot:
  {
    \pdfmeta_standard_verify:nF
      {annot_flags}
      {
        \bitset_set_true:Nn  \l__pdffield_F_bitset {Print}
        \bitset_set_false:Nn \l__pdffield_F_bitset {Hidden}
        \bitset_set_false:Nn \l__pdffield_F_bitset {Invisible}
        \bitset_set_false:Nn \l__pdffield_F_bitset {NoView}
      }
    \pdfannot_dict_put:nnx {widget}{F}{ \bitset_to_arabic:N \l__pdffield_F_bitset }
    \tl_if_empty:NF \l__pdffield_currentparent_tl
      {
         \exp_args:Ne
         \pdf_object_if_exist:nTF { __pdffield/field/\l__pdffield_currentparent_tl }
           {
             \pdfannot_dict_put:nnx {widget}{Parent}
               {
                 \exp_args:Ne
                   \pdf_object_ref:n{__pdffield/field/\l__pdffield_currentparent_tl}
               }
           }
           {
               \msg_error:nnx { pdffield }{parent-field-missing}{\l__pdffield_currentparent_t
           }
       }
    \mode_leave_vertical:
    \hbox_to_wd:nn
      { \l__pdffield_annot_wd_dim  }
      {
        \rule [-\l__pdffield_annot_dp_dim]{0pt}{\dim_eval:n{\l__pdffield_annot_ht_dim+\l__pdf
        \pdfannot_widget_box:nnn
           { \l__pdffield_annot_wd_dim }
           { \l__pdffield_annot_ht_dim }
           { \l__pdffield_annot_dp_dim }
         \hfill
      }
    \tl_if_empty:NF \l__pdffield_currentparent_tl
      {
        \seq_if_exist:cTF {g__pdffield_field/Kids/\l__pdffield_currentparent_tl _seq}
         {
           \seq_gput_right:cx
             {g__pdffield_field/Kids/\l__pdffield_currentparent_tl _seq}
             { \pdfannot_box_ref_last:}
         }
         {
           \msg_error:nnx { pdffield}{parent-field-missing}{\l__pdffield_currentparent_tl}
         }
      }
  }
\cs_new_protected:Npn \pdffield_annot:n #1
  {
    \group_begin:
    \keys_set:nn { pdffield } {#1}
    \__pdffield_annot:
    \group_end:
  }
```

(*End definition for* `\__pdffield_annot:`.)

6.7 auxiliary command for color keys

```
\cs_new_protected:Npn \__pdffield_color_set:nn #1 #2
 {
   \tl_if_head_eq_charcode:nNTF {#2}[ %]
    {
      \__pdffield_color_set_aux:nwn  { #1 } #2
    }
    {
      \color_set:nn {#1} {#2}
    }
 }

\cs_new_protected:Npn \__pdffield_color_set_aux:nwn #1 [#2] #3
  {
     \color_set:nnn {#1}{#2}{#3}
  }

```

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.

```
\cs_new_protected:Npn \__pdffield_value_handler:nN #1#2
  {
    \tl_set:Nn #2 {#1}
  }
\keys_define:nn { pdffield  }
  {
    ,parent .tl_set:N = \l__pdffield_currentparent_tl
    ,parent .groups:n = {field,annot}
    ,T .code:n =
      {
        \pdf_string_from_unicode:nnN {utf8/string-raw}{#1}\l__pdffield_tmpa_str
        \str_if_in:NnT \l__pdffield_tmpa_str {.}
          {
            \msg_error:nnx {pdffield}{no-period}{\l__pdffield_tmpa_str}
          }
        \str_if_empty:NTF\l__pdffield_tmpa_str
          {
            \msg_warning:nn {pdffield}{empty-name}
            \pdfdict_remove:nn { l__pdffield/field }{T}
          }
          {
            \pdfdict_put:nnx { l__pdffield/field }{T}{(\l__pdffield_tmpa_str)}
          }
      }
    ,T .value_required:n = true
    ,T .groups:n = {field}
    ,name .meta:n          = {T={#1}}
    ,name .value_required:n = true
    ,name .groups:n = {field}
    ,TU .groups:n = {field}
    ,TU .code:n =
      {
        \tl_if_empty:nTF {#1}
          {
            \pdfdict_remove:nn { l__pdffield/field }{TU}
          }
          {
            \pdf_string_from_unicode:nnN {utf8/string}{#1}\l__pdffield_tmpa_str
            \pdfdict_put:nnx { l__pdffield/field }{TU}{\l__pdffield_tmpa_str}
          }
      }
    ,TU .groups:n = {field}
    ,altname .meta:n      = {TU={#1}}
    ,altname .groups:n = {field}
    ,TM .code:n =
      {
        \tl_if_empty:nTF {#1}
          {
            \pdfdict_remove:nn { l__pdffield/field }{TM}
          }
          {
            \pdf_string_from_unicode:nnN {utf8/string}{#1}\l__pdffield_tmpa_str
            \pdfdict_put:nnx { l__pdffield/field }{TM}{\l__pdffield_tmpa_str}
          }
      }
    ,TM .groups:n = {field}
    ,mappingname .meta:n   = {TM={#1}}
    ,mappingname .groups:n = {field}
    ,FT .choices:nn =
      { Btn, Tx, Ch, Sig }
      {
        \pdfdict_put:nnn { l__pdffield/field }{FT}{ /#1 }
      }
    ,FT .groups:n = {field}
    ,V .code:n =
     {
       \tl_if_empty:nTF {#1}
         {
           \pdfdict_remove:nn { l__pdffield/field }{V}
         }
         {
           \__pdffield_value_handler:nN{#1}\l__pdffield_tmpa_str
           \pdfdict_put:nnx { l__pdffield/field }{V}{ \l__pdffield_tmpa_str }
         }
     }
    ,V .groups:n = {field}
    ,DV .code:n =
     {
       \tl_if_empty:nTF {#1}
         {
           \pdfdict_remove:nn { l__pdffield/field }{DV}
         }
         {
           \__pdffield_value_handler:nN{#1}\l__pdffield_tmpa_str
           \pdfdict_put:nnx { l__pdffield/field }{DV}{ \l__pdffield_tmpa_str }
         }
     }
    ,DV .groups:n = {field}
    ,MaxLen .code:n =
     {
       \tl_if_empty:nTF {#1}
         {
           \pdfdict_remove:nn { l__pdffield/field }{MaxLen}
         }
         {
           \pdfdict_put:nnx { l__pdffield/field }{MaxLen}{ #1 }
         }
      }
    ,MaxLen .groups:n = {field}
    ,Lock .code:n =
      {
        \tl_if_empty:nTF {#1}
          {
            \pdfdict_remove:nn { l__pdffield/field }{Lock}
          }
          {
            \pdfdict_put:nnx { l__pdffield/field }{Lock}{ \pdf_object_ref:n{#1} }
          }
      }
    ,Lock .groups:n = {field}
    ,SV .code:n =
      {
        \tl_if_empty:nTF {#1}
          {
            \pdfdict_remove:nn { l__pdffield/field }{SV}
          }
          {
            \pdfdict_put:nnx { l__pdffield/field }{SV}{ \pdf_object_ref:n{#1} }
          }
      }
    ,SV .groups:n = {field}
    ,Opt .code:n =
      {
        \tl_if_empty:nTF {#1}
          {
            \pdfdict_remove:nn { l__pdffield/field }{Opt}
          }
          {
            \pdfdict_put:nnx { l__pdffield/field }{Opt}{ \pdf_object_ref:n{#1} }
          }
      }
    ,Opt .groups:n = {field}
    ,TI .code:n =
      {
        \tl_if_empty:nTF {#1}
          {
            \pdfdict_remove:nn { l__pdffield/field }{TI}
          }
          {
            \pdfdict_put:nnx { l__pdffield/field }{TI}{ #1 }
          }
       }
    ,TI .groups:n = {field}
    ,I .code:n =
      {
        \tl_if_empty:nTF {#1}
          {
            \pdfdict_remove:nn { l__pdffield/field }{I}
          }
          {
            \pdfdict_put:nnx { l__pdffield/field }{I}{ \pdf_object_ref:n{#1} }
          }
      }
    ,I .groups:n = {field}
  }
```

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

```
\keys_define:nn { pdffield }
  {
    ,setFf .code:n =
      {
          \clist_map_inline:nn {#1}
           {
            \bitset_set_true:Nn \l__pdffield_Ff_bitset {##1}
          }
      }
    ,setFf .groups:n = {field}
    ,setfieldflags .meta:n =
      {setFf={#1}}
    ,setfieldflags .groups:n = {field}
    ,unsetFf .multichoice:
    ,unsetFf / all .code:n = { \bitset_clear:N \l__pdffield_Ff_bitset}
    ,unsetFf / unknown .code:n =
      {
        \bitset_set_false:Nn \l__pdffield_Ff_bitset {#1}
      }
    ,unsetFf .groups:n = {field}
    ,unsetfieldflags .meta:n = {unsetFf={#1}}
    ,unsetfieldflags .groups:n = {field}
  }

```

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

```
\cs_set_protected:Npn \__pdffield_tmpa:n #1  %
  {
    \keys_define:nn { pdffield  }
      {
         AA/#1 .code:n =
           {
             \pdf_string_from_unicode:nnN {utf8/string-raw}{##1}\l__pdffield_tmpa_str
             \str_if_empty:NTF \l__pdffield_tmpa_str
               {
                 \pdfdict_remove:nn {l__pdffield/field/AA}{#1}
               }
               {
                 \pdfdict_put:nnx {l__pdffield/field/AA}
                  {#1}
                  {<</S/JavaScript/JS(\l__pdffield_tmpa_str)>>}
               }
           },
        AA/#1 .groups:n  = {field}
      }
  }

\clist_map_inline:nn {K,F,V,C}{\__pdffield_tmpa:n{#1}}

\cs_set_protected:Npn \__pdffield_tmpa:nn #1 #2
  {
    \keys_define:nn { pdffield  }
      {
         #1 .meta:nn =
           { pdffield }{AA/#2={##1}},
        #1 .groups:n  = {field}
      }
  }
\__pdffield_tmpa:nn {keystroke}{K}
```

```
\__pdffield_tmpa:nn {format}   {F}
\__pdffield_tmpa:nn {validate} {V}
\__pdffield_tmpa:nn {calculate}{C}


\keys_define:nn { pdffield }
  {
    DA .code:n =
     {
       \tl_if_empty:nTF {#1}
         {
           \pdfdict_remove:nn { l__pdffield/field }{DA}
         }
         {
           \pdfdict_put:nnx { l__pdffield/field }{DA}{ (#1) }
         }
      }
    ,DA .groups:n = {field}
    ,Q .choices:nn = {left,center,right}
     {
       \pdfdict_put:nnx { l__pdffield/field }{Q}{ \int_eval:n{\l_keys_choice_int-1} }
     }
    ,Q / .code:n = { \pdfdict_remove:nn { l__pdffield/field }{Q} }
    ,Q .groups:n = {field}
    ,align .meta:n={Q=#1}
    ,DS .code:n =
     {
       \msg_warning:nnn {pdffield}{not-implemented}{DS}
     }
    ,DS .groups:n = {field}
    ,RV .code:n =
     {
       \msg_warning:nnn {pdffield}{not-implemented}{RV}
     }
    ,RV .groups:n = {field}
  }
```

6.9 Annotation keys

The size of the field annotation

```
\dim_new:N \l__pdffield_annot_ht_dim
\dim_new:N \l__pdffield_annot_wd_dim
\dim_new:N \l__pdffield_annot_dp_dim

\keys_define:nn { pdffield }
  {
    ,width  .dim_set:N = \l__pdffield_annot_wd_dim
    ,height .dim_set:N = \l__pdffield_annot_ht_dim
    ,depth  .dim_set:N = \l__pdffield_annot_dp_dim
    ,width  .initial:n = 0pt
    ,height .initial:n = 0pt
    ,depth  .initial:n = 0pt
  }
```

```
\keys_define:nn { pdffield }
 {
   %parent is defined in field
  ,AS .code:n =
    {
      \tl_if_empty:nTF {#1}
        {
          \pdfannot_dict_remove:nn { widget }{AS}
        }
        {
          \pdfannot_dict_put:nnx {widget}{AS}{\pdf_name_from_unicode_e:n{#1}}
        }
    }
  ,AS .groups:n = annot
 }
\cs_new_protected:Npn \__pdffield_appearance_handler:nnn #1#2#3
 {
   \pdfxform_if_exist:nTF {  #1 }
     {
       \pdfannot_dict_put:nnx {widget/AP}{#2}
         {
           \pdfxform_ref:n {#1}
         }
     }
     {
       \msg_error:nnnn{pdffield}{appearance-missing}{#1}{#3}
     }
 }
\keys_define:nn { pdffield }
  {
    AP/N .code:n =
      {
        \tl_if_empty:nTF {#1}
          {
            \pdfannot_dict_remove:nn { widget/AP }{N}
          }
          {
            \__pdffield_appearance_handler:nnn {#1}{N}{normal}
          }
        }
     ,AP/N .groups:n = annot
     ,appearance .meta:n = {AP/N={#1}}
  }
\keys_define:nn { pdffield }
  {
    AP/R .code:n =
      {
        \tl_if_empty:nTF {#1}
          {
            \pdfannot_dict_remove:nn { widget/AP }{R}
          }
          {
             \__pdffield_appearance_handler:nnn {#1}{R}{rollover}
          }
        }
   ,AP/R .groups:n = annot
   ,rollover-appearance .meta:n = {AP/R={#1}}
  }
\keys_define:nn { pdffield }
  {
    AP/D .code:n =
      {
        \tl_if_empty:nTF {#1}
          {
            \pdfannot_dict_remove:nn { widget/AP }{D}
          }
          {
             \__pdffield_appearance_handler:nnn {#1}{D}{rollover}
          }
        }
   ,AP/D .groups:n = annot
   ,down-appearance .meta:n = {AP/D={#1}}
  }

\keys_define:nn { pdffield  }
  {
    MK/R .choices:nn = {0,90,180,270}
     {
       \pdfannot_dict_put:nnx {widget/MK}{R}{#1}
     }
   ,MK/R / .code:n =
     {
        \pdfannot_dict_remove:nn { widget/MK }{R}
     }
   ,MK/R .groups:n = annot
   ,rotate .meta:n = {MK/R=#1}
  }

\keys_define:nn { pdffield  }
  {
    MK/BC .code:n =
     {
       \tl_if_empty:nTF {#1}
        {
          \pdfannot_dict_remove:nn { widget/MK }{BC}
        }
        {
          \__pdffield_color_set:nn {__pdffield/tmp}{#1}
          \color_export:nnN{__pdffield/tmp}{space-sep-rgb}\l__pdffield_tmpa_tl
          \pdfannot_dict_put:nnx {widget/MK}{BC}{[\l__pdffield_tmpa_tl]}
        }
     }
    ,MK/BC .groups:n = annot
   ,bordercolor .meta:n = {MK/BC=#1}
  }

\keys_define:nn { pdffield  }
  {
    MK/BG .code:n =
     {
       \tl_if_empty:nTF {#1}
        {
          \pdfannot_dict_remove:nn { widget/MK }{BG}
        }
        {
          \__pdffield_color_set:nn {__pdffield/tmp}{#1}
          \color_export:nnN{__pdffield/tmp}{space-sep-rgb}\l__pdffield_tmpa_tl
          \pdfannot_dict_put:nnx {widget/MK}{BG}{[\l__pdffield_tmpa_tl]}
        }
     }
    ,MK/BG .groups:n = annot
   ,bordercolor .meta:n = {MK/BG=#1}
  }


\keys_define:nn { pdffield  }
  {
    MK/CA .code:n =
     {
       \tl_if_empty:nTF {#1}
        {
          \pdfannot_dict_remove:nn { widget/MK }{CA}
        }
        {
          \pdf_string_from_unicode:nnN {utf8/string}{#1}\l__pdffield_tmpa_str
          \pdfannot_dict_put:nnx {widget/MK}{CA}{\l__pdffield_tmpa_str}
        }
     }
    ,MK/CA .groups:n = annot
   ,caption .meta:n = {MK/CA=#1}
  }

\cs_set_protected:Npn \__pdffield_tmpa:n #1
 {
   \keys_define:nn { pdffield  }
     {
       MK/#1 .code:n =
        {
          \tl_if_empty:nTF {##1}
            {
              \pdfannot_dict_remove:nn { widget/AP }{#1}
            }
            {
              \pdfannot_dict_put:nnx {widget/MK}{#1}{##1}
            }
        }
      ,MK/#1 .groups:n = annot
     }
 }

\clist_map_inline:nn {RC,AC,I,RI,IX,IF,TP}
  { \__pdffield_tmpa:n {#1} }
```

Flags.

```
\keys_define:nn { pdffield  }
  {
    ,setF .code:n =
      {
          \clist_map_inline:nn {#1}
           {
             \bitset_set_true:Nn \l__pdffield_F_bitset {##1}
           }
      }
    ,setF .groups:n = annot
    ,setannotflags .meta:nn =
      { pdffield }{setF={#1}}
    ,setannotflags .groups:n = annot
    ,unsetF .multichoice:
    ,unsetF / all .code:n = { \bitset_clear:N \l__pdffield_F_bitset}
    ,unsetF / unknown .code:n =
      {
        \bitset_set_false:Nn \l__pdffield_F_bitset {#1}
      }
    ,unsetF .groups:n = annot
    ,unsetannotflags .meta:nn =
      { pdffield }{unsetF= {#1} }
    ,unsetannotflags .groups:n = annot
  }

```

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

```
\cs_set_protected:Npn \__pdffield_tmpa:n #1  %
  {
    \keys_define:nn { pdffield }
      {
         AA/#1 .code:n =
           {
             \pdf_string_from_unicode:nnN {utf8/string-raw}{##1}\l__pdffield_tmpa_str
             \str_if_empty:NTF \l__pdffield_tmpa_str
               {
                 \pdfannot_dict_remove:nn {widget/AA}{#1}
               }
               {
                 \pdfannot_dict_put:nnx {widget/AA}
                  {#1}
                  {<</S/JavaScript/JS(\l__pdffield_tmpa_str)>>}
               }
           },
         ,AA/#1 .groups:n = annot
      }
  }

\clist_map_inline:nn {Fo,Bl,D,U,E,X,PO,PC,PV,PI}{\__pdffield_tmpa:n{#1}}

\cs_set_protected:Npn \__pdffield_tmpa:nn #1 #2
```

```
  {
    \keys_define:nn { pdffield }
      {
         #1 .meta:nn =
           { pdffield }{AA/#2={##1}},
         #1 .groups:n = {annot}
      }
  }
\__pdffield_tmpa:nn {onfocus}  {Fo}
\__pdffield_tmpa:nn {onblur}   {Bl}
\__pdffield_tmpa:nn {onmousedown}{D}
\__pdffield_tmpa:nn {onmouseup}{U}
\__pdffield_tmpa:nn {onenter}  {E}
\__pdffield_tmpa:nn {onexit}   {X}
```

6.10 Appearances

\pdffield_appearance:nn

```
\cs_new_protected:Npn \pdffield_appearance:nn #1 #2
  {
     \pdfxform_new:nnn {#1}{}{#2}
  }

\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

```
\keys_define:nn { pdffield / setup }
  {
    ,create-style .code:n  = { \__pdffield_style_create:nn #1 }
    ,preset-checkbox .code:n =
      {
        \keys_define:nn { pdffield }
         {
           __pdffield/preset/checkbox .meta:n = {#1},
         }
      }
   ,preset-textfield .code:n =
      {
        \keys_define:nn { pdffield }
         {
           __pdffield/preset/textfield .meta:n = {#1},
         }
      }
  }
\keys_set:nn{ pdffield / setup }{preset-checkbox={}}
\keys_set:nn{ pdffield / setup }{preset-textfield={}}

\cs_new_protected:Npn \__pdffield_style_create:nn #1#2
  {
    \keys_define:nn { pdffield }
      {
          __pdffield/style/#1 .meta:n = {#2},
        }
    }


\cs_new_protected:Npn \pdffield_setup:n #1
    {
       \keys_set:nn{ pdffield / setup }{#1}
    }

\keys_define:nn { pdffield }
    {
      style .code:n = {\keys_set:nn {pdffield}{__pdffield/style/#1={#1}}}
    }
```
⟨/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.