# The l3pdffield-choice module
# Commands to create choice fields
# LaTeX PDF management bundle

The LaTeX Project*

Version 0.96t, released 2025-06-29

## 1 l3pdffield-choice Introduction

This is the documentation for choice fields, for general information about form fields check the documentation l3pdffield.

Please keep in mind

- Not every PDF viewer supports choice field.
- 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 features of form fields too (as you typically can't change the PDF).

## 2 Choice fields

Choice fields are drop down menus or scrollable lists where the user can select one or more entries. They can also contain a field where users can insert a free text. The export value and the displayed value can differ. Some values can be preselected.

This means that various data will have to be set, and the sorting matters. The module here will assume that the various values are stored in sequences:

```
\seq_const_from_clist:Nn \c_my_values_seq {hippo,duck,bear}
\seq_const_from_clist:Nn \c_my_displayvalues_seq {Sieglinde,,Bär}
\seq_const_from_clist:Nn \c_my_defaultvalues_seq {duck,Bär}
```

Only the first sequence is required. Empty values in the display sequence are possible, then the normal value is used.

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

## 2.1 Types

Choice fields can be a drop down menu (called `Combo`), which can contain an editable field.

`setfieldflags={Combo,Edit}` or `setfieldflags={Combo}`
If `Edit` is set, one can also set `DoNotSpellCheck`.
Or they can be a list.
`unsetfieldflags={Combo,Edit,DoNotSpellCheck}`
For both types it is possible to set or unset `MultiSelect` and `CommitOnSelChange`.

## 2.2 Appearance

By default the fields use a grey background as appearance, another appearance can be defined with `\pdffield_appearance:nn` (described in the documentation of l3pdffield) or directly with `\pdfxform_new:nnn` (in l3pdfxfom) and then used with the `appearance` as in other form fields.

## 2.3 Commands

`\pdffield_choice:n` `\pdffield_choice:n{`⟨*key val list*⟩`}`

This creates a choice field. The list of allowed keys is described below. The ⟨*key val list*⟩ should at least set the name, without it the default `choice` is used. Choice fields with the same name belong to the same field and if changed, the others are changed accordingly. As default appearance we reuse the text appearance: a simple gray background.

The first choice field setups the field and so should setup values and field flags.

## 2.4 Keys

The new choice command accept all field and annot keys from l3pdffield. A few keys are disabled or are forced to specific values. The `appearance` keys have a ... behaviour, other keys have other defaults than with the basic commands. Additionally there are a small number of keys specific to a choice field. `value` and `default` have a special meaning.

Disabled keys are

- `V`, `DV`, `AS`: they are set by the other keys.
- `FT` is overwritten.
- For fields only the field flags `ReadOnly`, `Required`, `NoExport`, `Combo`, `Edit`, `MultiSelect`, `Sort`, `DoNotSpellCheck` and `CommitOnSelChange` make sense. The code will force some values (e.g. disable `Edit` if `Combo` is false).

`preset-choice` `preset-choice = {`⟨*key-val-list*⟩`}`

This allows to set default keys for a choice field.

`type` `type = combo|combo-edit|list`

This key is a choice key that allows to set the three types.

name `name = ⟨partial name⟩`
T `T = ⟨partial name⟩`

These keys set the partial name of the field. They both do the same thing, use the one you are more comfortable with. The value shouldn't contain a period, be not empty and sensibly consist of simple chars. Additionally the value is used to create the field ID. This means that choice field with the same partial name are annotations with the same field as parent. The field ID is then internal and can not be used to attach another annotation. For explicit control of the field ID use the `fieldID` key.

value `value = {⟨comma separated list of values⟩}`
values `values = {⟨comma separated list of values⟩}`

With these keys you set the export value names. ⟨*comma separated list of values*⟩ should contain the values in the order as wanted. The values are passed through `\pdf_string_from_unicode:nnN` and can use unicode.

display-values `display-values = {⟨comma separated list of values⟩}`

With this key you set the display value names if they should be different from the export values. They should be given in the same order as the values. The values are passed through `\pdf_string_from_unicode:nnN` and can use unicode.

default `default = {⟨comma separated list of values⟩}`
default-values `default-values = {⟨comma separated list of values⟩}`

With this keys you set the values which are selected when the PDF is opened. The use of this keys is a bit dubious. It seems to work okay for combo fields, but not for list fields. According to the PDF reference the values should be the display values, but it is not clear if the PDF viewer really implement it that way or if the export value should be preferred. Depending on the values it could be needed to add also an `I` array to the field.

top-index `top-index = {⟨integer⟩}`

With this key you set for a scrollable list field the number of the value shown at the top. The first value has number 1 (that is the default). I'm not sure if viewers handle this correctly, so the setting should be used with care.

fieldID `fieldID = ⟨field ID⟩`

*For experts only!* This allows to give the radio field a specific ID. This is only useful in the context of a larger fieldset or if you want to attach another annotation to the field with `\pdffield_annot:n`. If used wrongly you can easily create invalid fieldset. It allows you to create to fields with the same partial name, but if you want to see both you need to ensure that their full names are different—for example by adding some parent fields.

parent `parent = ⟨field ID⟩`

This is only needed if the field should be part of a larger fieldset. The value should be a field ID of a field created previously with `\pdffield_field:nn`.

| | |
|---|---|
| width | width = ⟨*dim expression*⟩ |
| height | height = ⟨*dim expression*⟩ |
| depth | depth = ⟨*dim expression*⟩ |

These keys allow to set the dimensions of the choice field. The value should be a dimension expression. By default `width` and `height` and `depth` are ....

| | |
|---|---|
| AP/N | AP/N = ⟨*appearance name*⟩ |
| appearance | appearance = ⟨*appearance name*⟩ |
| AP/R | AP/R = ⟨*appearance name*⟩ |
| rollover-appearance | rollover-appearance = ⟨*appearance name*⟩ |
| AP/D | AP/D = ⟨*appearance name*⟩ |
| down-appearance | down-appearance = ⟨*appearance name*⟩ |

This adds appearances. ⟨*appearance name*⟩ should be the name of an form Xobjects. The default normal appearance is a simple gray background, the same as with text fields.

The font are handled in a similar way as in textfield. Most of the remarks made there are true for choice fields too.

| | |
|---|---|
| fontcolor | fontcolor = ⟨*color expression*⟩ \| [⟨*model*⟩]{⟨*values*⟩} |

This is sets the color of font in the textfield. Internally currently RGB is used. The colors used in ⟨*color expression*⟩ must be known to the l3color commands. The initial color is black. It is a *field* setting, that means instances share the color.

| | |
|---|---|
| fontsize | fontsize = ⟨*dim expression*⟩ |

This is sets the size of the font in the textfield. The default value is the current font size (`\f@size` in pt). It is a *field* setting, that means instances share the size.

| | |
|---|---|
| font | font = ⟨*symbolic font name*⟩ |

This is sets the font in the textfield. See the discussion at the begin of the documentation about some background. The default value is `Helv` and this name is setup if you load hyperref and issue the `\Form` command. The default value is the current font size (`\f@size` in pt). It is a *field* setting, that means instances share the font.

### 2.5 Using with hyperref

hyperref combines radio button and choice field in one command. This makes it difficult to replicate the hyperref commands here.

## 3 l3pdffield-choice Implementation

```
1 ⟨*package⟩
2 ⟨@@=pdffield⟩
3 \RequirePackage{l3draw}
```

### 3.1 Variables

\l__pdffield_choice_values_seq
\l__pdffield_choice_defaultvalues_seq
\l__pdffield_choice_displayvalues_seq
\l__pdffield_choice_opt_seq

variables to hold the value, the default value, and display variants. The opt seq will be used to build the content of opt

```
4 \seq_new:N  \l__pdffield_choice_values_seq
```

```
\seq_new:N  \l__pdffield_choice_defaultvalues_seq
\seq_new:N  \l__pdffield_choice_displayvalues_seq
\seq_new:N  \l__pdffield_choice_opt_seq
```

(*End of definition for* `\l__pdffield_choice_values_seq` *and others.*)

## 3.2 Appearances

(probably take from textfield ...)

## 3.3 Creating the field

A field should be created if the name doesn't exist yet

`\__pdffield_choice_field:n`

```
\cs_new_protected:Npn \__pdffield_choice_field:n #1 %name
  {
    \pdf_object_if_exist:nF {__pdffield/field/__pdffield/choice/#1}
      {
```

We must at first build the Opt array.

```
        \seq_clear:N \l__pdffield_choice_opt_seq
        \seq_map_indexed_inline:Nn \l__pdffield_choice_values_seq
          {
            \pdf_string_from_unicode:nnN{utf16/hex}{##2}\l__pdffield_tmpa_str
            \tl_set:Ne \l__pdffield_tmpa_tl {\seq_item:Nn \l__pdffield_choice_displayvalues_s
            \tl_if_empty:NTF \l__pdffield_tmpa_tl
              {
                \seq_put_right:NV \l__pdffield_choice_opt_seq \l__pdffield_tmpa_str
              }
              {
                \exp_args:NnV
                \pdf_string_from_unicode:nnN{utf16/hex}\l__pdffield_tmpa_tl\l__pdffield_tmpb_
                \seq_put_right:Ne \l__pdffield_choice_opt_seq
                  { [ \l__pdffield_tmpa_str\c_space_tl\l__pdffield_tmpb_str] }
              }
          }
        \pdf_object_unnamed_write:ne {array}{\seq_use:Nn\l__pdffield_choice_opt_seq {~}}
        \pdfdict_put:nne { l__pdffield/field }{Opt}  { \pdf_object_ref_last: }
```

Now we handle the V value. If MultiSelect is set, we use the full displayvalues seq, if not only the first value.

```
        \int_compare:nNnTF {\bitset_item:Nn \l__pdffield_Ff_bitset {MultiSelect}} = {1}
          {
            \tl_clear:N \l__pdffield_tmpa_tl
            \seq_map_inline:Nn \l__pdffield_choice_defaultvalues_seq
             {
                \pdf_string_from_unicode:nnN{utf16/hex}{##1}\l__pdffield_tmpa_str
                \tl_put_right:NV \l__pdffield_tmpa_tl \l__pdffield_tmpa_str
                \pdfdict_put:nne { l__pdffield/field }{V}  { [ \l__pdffield_tmpa_tl ] }
             }
```

```
          }
          {
            \tl_set:Ne \l__pdffield_tmpa_tl {\seq_item:Nn \l__pdffield_choice_defaultvalues_s
            \exp_args:NnV
              \pdf_string_from_unicode:nnN{utf16/hex}\l__pdffield_tmpa_tl\l__pdffield_tmpb_st
            \pdfdict_put:nne { l__pdffield/field }{V}  { \l__pdffield_tmpb_str }
          }
```

now we create the field and set it as parent for the following annotation.

```
        \__pdffield_field:n { __pdffield/choice/#1 }
      }
    \keys_set:nn {pdffield}{parent={__pdffield/choice/#1}}
  }
\cs_generate_variant:Nn \__pdffield_choice_field:n {V}
```

(*End of definition for* `\__pdffield_choice_field:n`.)

## 3.4 Assembling the choice field

`\__pdffield_choice:n`

The argument are key-val settings. At first we map the handlers. As default appearance we reuse the text appearance: a simple gray background.

```
\cs_new_protected:Npn \__pdffield_choice:n #1
  {
    \group_begin:
    \cs_set_eq:NN\__pdffield_value_handler:n      \__pdffield_choice_value_handler:n
    \cs_set_eq:NN\__pdffield_default_handler:n    \__pdffield_choice_default_handler:n
    \__pdffield_textfield_default_appearance:
```

Setting up the defaults.

```
    \keys_set:nn {pdffield}
      {
         fieldID=
        ,name=choice
        ,appearance = pdffield/textfield/default
        ,width  = 3cm
        ,height = \normalbaselineskip
        ,type=combo-edit
        ,__pdffield/preset/choice
        ,#1
        ,FT= Ch
      }
     \__pdffield_choice_set_type:
     \keys_set:nn {pdffield}
      {
        ,DA=
           {
              \pdf_name_from_unicode_e:n{\l__pdffield_DA_fontname_tl}
              \c_space_tl
              \dim_to_decimal_in_bp:n{\l__pdffield_DA_fontsize_dim}
              \c_space_tl
              Tf
```

```
              \c_space_tl
              \l__pdffield_DA_fontcolor_tl
              \c_space_tl
              %\l__pdffield_text_DAextra_tl
            }
      }
```

If the fieldID has not been set explicitly, we use the name/T key

```
   \tl_if_empty:NT\l__pdffield_fieldID_tl
      {
        \pdfdict_get:nnN {l__pdffield/field}{T}\l__pdffield_fieldID_tl
        \tl_put_left:Nn \l__pdffield_fieldID_tl {__pdffield/choice/}
      }
```

Now we build the field

```
    \__pdffield_choice_field:V\l__pdffield_fieldID_tl
```

And add the annotation.

```
    \__pdffield_annot:
    \group_end:
  }
```

(*End of definition for* `\__pdffield_choice:n`.)

## 3.5 Keys and handlers

Most keys are inherited simply the ones from the generic field and annot keys. We define a group key, as the name is better. The value key sets the export value. default the button which is checked on. At first the two handlers

`\__pdffield_choice_value_handler:n`
`\__pdffield_choice_default_handler:n`
`display-values`

```
\cs_new_protected:Npn \__pdffield_choice_value_handler:n #1
  {
    \seq_set_from_clist:Nn \l__pdffield_choice_values_seq {#1}
  }
\cs_new_protected:Npn \__pdffield_choice_default_handler:n #1
  {
    \seq_set_from_clist:Nn \l__pdffield_choice_defaultvalues_seq {#1}
  }
\keys_define:nn{pdffield}
  {
     values .meta:n = {value={#1}}
    ,default-values .meta:n = {default={#1}}
    ,display-values .code:n =
      {
        \seq_set_from_clist:Nn \l__pdffield_choice_displayvalues_seq {#1}
      }
   ,top-index .code:n =
      {
        \pdfdict_put:nne {l__pdffield/field}{TI}{\int_eval:n{#1-1}}
      }
  }
```

(*End of definition for* `\__pdffield_choice_value_handler:n`, `\__pdffield_choice_default_handler:n`, *and* `display-values`. *This function is documented on page 3.*)

type

```
\cs_new_protected:Npn \__pdffield_choice_set_type: {}

\keys_define:nn { pdffield }
  {
    type .choice:
   ,type / combo      .code:n =
     {
       \cs_set_protected:Npn\__pdffield_choice_set_type:
        {
          \keys_set:nn{pdffield}{setFf={Combo},unsetFf={Edit,DoNotSpellCheck}}
        }
     }
   ,type / combo-edit .code:n =
     {
       \cs_set_protected:Npn\__pdffield_choice_set_type:
         {
           \keys_set:nn{pdffield}{setFf={Combo,Edit}}
         }
     }
   ,type / list       .code:n =
     {
       \cs_set_protected:Npn\__pdffield_choice_set_type:
         {
           \keys_set:nn{pdffield}{unsetFf={Combo,Edit,},unsetFf={Edit,DoNotSpellCheck}}
         }
     }
  }
```

(*End of definition for* `type`. *This function is documented on page 2.*)

__pdffield_choice_appearance_handler:nnn

The handler for the appearances stores only the code as it must be executed rather late.

```
\cs_new_protected:Npn \__pdffield_choice_appearance_handler:nnn #1 #2 #3 %name, type, text
  {
  }

```

(*End of definition for* `\__pdffield_choice_appearance_handler:nnn`.)

## 3.6 User commands

\pdffield_choice:n

```
\cs_set_eq:NN \pdffield_choice:n \__pdffield_choice:n
⟨/package⟩
```

(*End of definition for* `\pdffield_choice:n`. *This function is documented on page 2.*)

# 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.