# The l3pdfdict package – tools for PDF dictionaries

The LaTeX3 Project[*]

Released XXXX-XX-XX

## 1 l3pdfdict documentation

Many PDF objects are or contain dictionaries – structures containing a number of (pdf-)Name/value pairs. Examples are attributes of links, filespec dictionaries, xform dictionaries, the catalog, the info dictionary. The commands in this module offer an number of tools to handle such dictionaries. The module setups a namespace for the dictionary names and offers some commands to output dictionaries.

The dictionaries are implemented with property lists and in many respects they work like them with a few PDF specific changes:

- The keys are always converted with `\str_convert_pdfname` to get a correct PDF name;
- a key with a empty value can not be added, it will be ignored;
- there is a dedicated function to output the property as space separated list with keys with slash: `/key1 value1 /key2 value2`.

Local and global dictionaries can be created.

### 1.1 User Commands

`\pdfdict_new:n`

Updated: 2020-12-03

`\pdfdict_new:n {⟨dictionary name⟩}`

This function create a new local or global dictionary. Which one depends on ⟨*dictionary name*⟩: If it begins with the standard `g` the dictionary is global. With `l` the dictionary is local. Other begins will give an error. It is recommended to begin the name in the standard expl3 naming scheme with a module name, so `g_module_XXXX` or `g__module_XXXX`.

`\pdfdict_set_eq:nn`
`\pdfdict_gset_eq:nn`

New: 2020-06-16
Updated: 2020-12-03

```
\pdfdict_set_eq:nn  {⟨local dictionary name_1⟩} {⟨dictionary name_2⟩}
\pdfdict_gset_eq:nn {⟨global dictionary name_1⟩}{⟨dictionary name_2⟩}
```

This functions copy ⟨*dictionary name*$_2$⟩ into ⟨*local/global dictionary name*$_1$⟩ locally or globally. If the dictionary ⟨*local/global dictionary name*$_1$⟩ doesn't exist yet, it will be created. If ⟨*dictionary name*$_2$⟩ doesn't exist yet, an error will be raised.

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

`\pdfdict_put:nnn`
`\pdfdict_gput:nnn`

New: 2020-04-06

```
\pdfdict_put:nnn  {⟨local dictionary⟩}  {⟨name⟩} {⟨value⟩}
\pdfdict_gput:nnn {⟨global dictionary⟩} {⟨name⟩} {⟨value⟩}
```

This function puts key ⟨*name*⟩ and value ⟨*value*⟩ locally or globally in the ⟨*dictionary*⟩ created with `\pdfdict_new:n`. `{`⟨*name*⟩`}` should be a PDF Name without the starting slash. It will be stored with `\str_convert_pdfname`, so will be automatically correctly escaped in case it contains slashes, spaces or other chars not allowed in a PDF name. ⟨*value*⟩ should be a valid PDF value for this name in the target dictionary. The value is *neither* converted *nor* escaped automatically. If the value is blank nothing is added to the dictionary.

When adding a value keep in mind that the expansion behaviour of the backends can differ. Some backends expand a value always fully when writing to the PDF, with other backends commands could end as strings in the PDF. So one should neither rely on ⟨*value*⟩ to be expanded nor not expanded by the backend commands.

`\pdfdict_item:nn` ⋆
`\pdfdict_item:ne` ⋆

New: 2020-12-04

```
\pdfdict_item:nn {⟨key⟩} {⟨value⟩}
```

A simple command to output key-value as `/key value`. This is needed to output dictionaries in mapping commands. The command doesn't do any escaping, it expects that the name has been escaped when the value has been stored into the dictionary. If the value is blank nothing is output. The command is expandable if the content is it.

`\pdfdict_use:n` ⋆

Updated: 2020-12-03

```
\pdfdict_use:n {⟨dictionary⟩}
```

This outputs the property list of the dictionary as a list of `/key value` pairs. This can be used e.g. when writing a dictionary object with `\pdf_object_write:nx`

`\pdfdict_show:n`

Updated: 2020-12-03

```
\pdfdict_show:n  {⟨dictionary⟩}
```

This shows the content of ⟨*dictionary*⟩ in the log and on the terminal.

`\pdfdict_if_exist_p:n` ⋆
`\pdfdict_if_exist:nTF` ⋆

Updated: 2020-12-03

```
\pdfdict_if_exist:n {⟨dictionary⟩}
```

This tests if the dictionary exists.

`\pdfdict_if_empty_p:n` ⋆
`\pdfdict_if_empty:nTF` ⋆

Updated: 2020-12-03

```
\pdfdict_if_empty:n {⟨dictionary⟩}
```

This tests if the dictionary is empty. The result is false if the dictionary doesn't exist.

`\pdfdict_get:nnN`

New: 2020-07-06

```
\pdfdict_get:nnN {⟨dictionary⟩}  {⟨name⟩} ⟨tl var⟩
```

Recovers the ⟨*value*⟩ stored by `\pdfdict_put:nnn` or `\pdfdict_gput:nnn` for ⟨*name*⟩ and places this in the ⟨*token list variable*⟩. If ⟨*name*⟩ is not found then the ⟨*token list variable*⟩ is set to the special marker `\q_no_value`. ⟨*name*⟩ is first converted with `\str_convert_pdfname`. The ⟨*token list variable*⟩ is set within the current TeX group.

`\pdfdict_remove:nn`
`\pdfdict_gremove:nn`

Updated: 2020-12-03

```
\pdfdict_remove:nn  {⟨local dictionary⟩}  {⟨name⟩}
\pdfdict_gremove:nn {⟨global dictionary⟩} {⟨name⟩}
```

Removes ⟨*name*⟩ and its associated ⟨*value*⟩ from the `{`⟨*dictionary*⟩`}` The removal is local from local dictionaries and global from global dictionaries. If ⟨*name*⟩ is not found no change occurs, *i.e* there is no need to test for the existence of a name before trying to remove it. ⟨*name*⟩ is first converted with `\str_convert_pdfname`.

## 2 l3pdfdict implementation

```
⟨@@=pdfdict⟩
⟨*package⟩
\ProvidesExplPackage {l3pdfdict} {2020-12-03} {0.6}
  {Tools for PDF dictionaries}
⟨/package⟩
```

### 2.1 messages

```
⟨*package⟩
\cs_new:Npn \__pdfdict_get_type:n #1
  {
    \str_case_e:nn { \str_head:n{#1} }
     {
       {g}{global}
       {l}{local}
     }
  }
\msg_new:nnn  { pdfdict } { show-dict }
  { %#1: name of the dictionary
    %#2: expanded content
    %#3: type
    The~#3~dictionary~'#1'~
    \tl_if_empty:nTF {#2}
      { is~empty \\>~ . }
      { contains~the~pairs~(without~outer~braces): #2 . }
  }
\msg_new:nnn  { pdfdict } { unknown-dict }
  {
    The~dictionary~'#1'~is~unknown.
  }
\msg_new:nnn  { pdfdict } { dict-already-defined  }
  {
    The~#2~dictionary~'#1'~is~already~defined.
  }
\msg_new:nnn  { pdfdict } { empty-value }
              { The~value~#1~for~#2~is~blank~and~will~be~ignored }

\msg_new:nnn  { pdfdict } { invalid-name }
              { Name~'#1'~is~not~valid\\
                Names~of~dictionaries~should~start~with~'g_'~or~'l_' }

```

### 2.2 Creating dictionaries

\g__pdfdict_names_seq
\g__pdfdict_gnames_seq

Two seq to store the used names for diagnostics.

```
⟨*package⟩
\seq_new:N \g__pdfdict_lnames_seq
\seq_new:N \g__pdfdict_gnames_seq
⟨/package⟩
```

(*End definition for* \g__pdfdict_names_seq *and* \g__pdfdict_gnames_seq*.*)

__pdfdict_name:n
__kernel_pdfdict_name:n
__pdfdict_new:n
\pdfdict_new:n

This are the commands to create new dictionaries and to access their internal name. All internal names start with `g__pdfdict_/` or `l__pdfdict_/`.

For the other modules we also need a kernel command to access the internal name to speed up the code and allow the use standard commands of the `prop` module to deal with the dictionaries. For example
`\prop_clear:c { \__kernel_pdfdict_name:n { name }}`

```
⟨*package⟩
\cs_new:Npn \__pdfdict_name:n #1  % #1 dictionary name
  {
    \str_head:n{#1}__pdfdict_/#1_prop
  }
\cs_set_eq:NN \__kernel_pdfdict_name:n \__pdfdict_name:n

\cs_new_protected:Npn \__pdfdict_new:n #1
  {
    \__pdfdict_if_exist:nTF { #1 }
      {
        \msg_error:nnxx
          { pdfdict }
          { dict-already-defined }
          { \tl_to_str:n {#1} }
          { \__pdfdict_get_type:n{#1} }
       }
       {
         \str_case_e:nnF { \str_head:n{#1} }
           {
             {g}
             {
                 \prop_new:c  { \__pdfdict_name:n { #1 } }
                 \seq_gput_right:cn {g__pdfdict_gnames_seq} { #1 }
             }
             {l}
             {
                 \prop_new:c  { \__pdfdict_name:n { #1 } }
                 \seq_gput_right:cn {g__pdfdict_lnames_seq} { #1 }
             }
           }
           {
             \msg_error:nnx{pdfdict}{invalid-name}{\tl_to_str:n{#1}}
           }
       }
  }

\cs_set_eq:NN \pdfdict_new:n \__pdfdict_new:n

⟨/package⟩
```

(*End definition for* `\__pdfdict_name:n` *and others. This function is documented on page 1.*)

`\__pdfdict_set_eq:nn`
`\pdfdict_set_eq:nn`
`\__pdfdict_gset_eq:nn`
`\pdfdict_gset_eq:nn`

```
⟨*package⟩
\cs_new_protected:Npn \__pdfdict_set_eq:nn #1 #2
  {
    \__pdfdict_if_exist:nTF  { #2 }
      {
        \__pdfdict_if_exist:nF { #1 }
```

```
          {
            \__pdfdict_new:n { #1 }
          }
        \prop_set_eq:cc { \__pdfdict_name:n {#1} }{ \__pdfdict_name:n {#2} }
      }
      {
         \msg_error:nnn { pdfdict } { unknown-dict } { #1 }
      }
  }

\cs_set_eq:NN \pdfdict_set_eq:nn \__pdfdict_set_eq:nn

\cs_new_protected:Npn \__pdfdict_gset_eq:nn #1 #2
  {
    \__pdfdict_if_exist:nTF { #2 }
      {
        \__pdfdict_if_exist:nF { #1 }
          {
            \__pdfdict_new:n { #1 }
          }
        \prop_gset_eq:cc { \__pdfdict_name:n {#1} }{ \__pdfdict_name:n {#2} }
      }
      {
         \msg_error:nnn { pdfdict } { unknown-dict } { #1 }
      }
  }

\cs_set_eq:NN \pdfdict_gset_eq:nn \__pdfdict_gset_eq:nn
⟨/package⟩
```

(*End definition for* `\__pdfdict_set_eq:nn` *and others. These functions are documented on page 1.*)

`\__pdfdict_if_exist:n`
`\pdfdict_if_exist:n`

Existence tests.

```
⟨*package⟩
%local
\prg_new_conditional:Npnn \__pdfdict_if_exist:n #1 { p , T , F , TF }
  {
    \prop_if_exist:cTF
      { \__pdfdict_name:n { #1 }  }
      { \prg_return_true:  }
      { \prg_return_false: }
  }
\prg_set_eq_conditional:NNn
  \pdfdict_if_exist:n \__pdfdict_if_exist:n { p , T , F , TF }

⟨/package⟩
```

(*End definition for* `\__pdfdict_if_exist:n` *and* `\pdfdict_if_exist:n`*. This function is documented on page* **??**.)

`\__pdfdict_if_empty:n`
`\pdfdict_if_empty:n`

Tests for emptiness.

```
⟨*package⟩
\prg_new_conditional:Npnn \__pdfdict_if_empty:n #1 { p , T , F , TF }
  {
    \prop_if_empty:cTF
```

```
      { \__pdfdict_name:n { #1 } }
      { \prg_return_true:  }
      { \prg_return_false: }
  }

\prg_set_eq_conditional:NNn
  \pdfdict_if_empty:n \__pdfdict_if_empty:n { p , T , F , TF }

⟨/package⟩
```

(*End definition for* `\__pdfdict_if_empty:n` *and* `\pdfdict_if_empty:n`*. This function is documented on page* **??**.)

`\__pdfdict_put:nnn`
`\pdfdict_put:nnn`
`\__pdfdict_gput:nnn`
`\pdfdict_gput:nnn`

These are the commands to store values into the dictionaries. The main difference to adding values to a normal property list is, that the keys are converted with `\str_convert_pdfname:n` and that empty values are ignored.

```
⟨*package⟩
\cs_new_protected:Npn \__pdfdict_put:nnn #1 #2 #3  %#1 (local) dict, #2 name, #3 value
  {
    \tl_if_blank:nTF { #3 }
      {
        \msg_warning:nnnn { pdfdict }{ empty-value }{ #2 } { #1 }
      }
      {
        \__pdfdict_if_exist:nTF  { #1 }
          {
            \exp_args:Nnx \prop_put:cnn
              { \__pdfdict_name:n { #1 } }{ \str_convert_pdfname:n { #2 } } { #3 }
          }
          {
            \msg_error:nnn { pdfdict } { unknown-dict } { #1 }
          }
      }
  }


\cs_set_eq:NN \pdfdict_put:nnn \__pdfdict_put:nnn
\cs_generate_variant:Nn \pdfdict_put:nnn {nnx,nno}

\cs_new_protected:Npn \__pdfdict_gput:nnn #1 #2 #3  %#1 global dict, #2 name, #3 value
  {
    \tl_if_empty:nTF { #3 }
      {
        \msg_warning:nnnn { pdfdict }{ empty-value }{ #2 } { #1 }
      }
      {
        \__pdfdict_if_exist:nTF { #1 }
          {
            \exp_args:Nnx \prop_gput:cnn
              { \__pdfdict_name:n { #1 } }{ \str_convert_pdfname:n { #2 } } { #3 }
          }
          {
            \msg_error:nnn { pdfdict } { unknown-dict } { #1 }
          }
      }
  }

\cs_set_eq:NN \pdfdict_gput:nnn \__pdfdict_gput:nnn
\cs_generate_variant:Nn \pdfdict_gput:nnn {nnx,nno}
⟨/package⟩
```

(*End definition for* `\__pdfdict_put:nnn` *and others. These functions are documented on page 2.*)

`\__pdfdict_get:nnN`
`\pdfdict_get:nnN`

Recover the values. The name must be first escaped to match the stored name.

```
⟨*package⟩
\cs_new_protected:Npn \__pdfdict_get:nnN  #1 #2 #3 %dict,key,macro
  {
    \__pdfdict_if_exist:nTF { #1 }
      {
        \exp_args:Nnx \prop_get:cnN
          { \__pdfdict_name:n { #1 } }
          { \str_convert_pdfname:n { #2 } } #3
      }
      {
        \msg_error:nnn { pdfdict } { unknown-dict } { #1 }
      }
  }

\cs_set_eq:NN \pdfdict_get:nnN \__pdfdict_get:nnN

⟨/package⟩
```

(*End definition for* `\__pdfdict_get:nnN` *and* `\pdfdict_get:nnN`*. This function is documented on page 2.*)

`\__pdfdict_remove:nn`
`\pdfdict_remove:nn`
`\__pdfdict_gremove:nn`
`\pdfdict_gremove:nn`

This removes a name/value pair from a dictionary. The name has to be passed through the escaping.

```
⟨*package⟩

\cs_new_protected:Npn \__pdfdict_remove:nn #1 #2 %dict,name
  {
    \__pdfdict_if_exist:nTF { #1 }
      {
        \exp_args:Nnx \prop_remove:cn
          { \__pdfdict_name:n  { #1 } }{ \str_convert_pdfname:n { #2 } }
      }
      {
        \msg_error:nnn { pdfdict } { unknown-dict } { #1 }
      }
  }
\cs_set_eq:NN \pdfdict_remove:nn \__pdfdict_remove:nn

\cs_new_protected:Npn \__pdfdict_gremove:nn #1 #2 %dict,name
  {
    \__pdfdict_if_exist:nTF  { #1 }
       {
         \exp_args:Nnx \prop_gremove:cn
           { \__pdfdict_name:n  { #1 } }{ \str_convert_pdfname:n { #2 } }
       }
```

```
        {
          \msg_error:nnn { pdfdict } { unknown-dict } { #1 }
        }
  }

\cs_set_eq:NN \pdfdict_gremove:nn \__pdfdict_gremove:nn
⟨/package⟩
```

(*End definition for* `\__pdfdict_remove:nn` *and others. These functions are documented on page 2.*)

`\__pdfdict_show:Nn`
`\pdfdict_show:n`

This allows to show the content of dictionaries. It also displays if a dictionary is local or global. If both exists both are shown.

```
⟨*package⟩
\cs_new_protected:Npn \__pdfdict_show:Nn #1#2 %#1 message command, #2 dict
  {
    \prop_if_exist:cTF { \__pdfdict_name:n { #2 } }
      {
        #1
            { pdfdict }
            { show-dict }
            { \tl_to_str:n {#2}  }
            { \prop_map_function:cN {\__pdfdict_name:n { #2 }} \msg_show_item:nn }
            { \__pdfdict_get_type:n{#2} }
            { }
      }
      {
        #1 { pdfdict } { unknown-dict } { #2 } {}{}{}
      }
  }
\cs_new_protected:Npn \pdfdict_show:n #1
  {
    \__pdfdict_show:Nn \msg_show:nnxxxx {#1}
  }
⟨/package⟩
```

(*End definition for* `\__pdfdict_show:Nn` *and* `\pdfdict_show:n`*. This function is documented on page 2.*)

`\__pdfdict_item:nn`
`\__pdfdict_item:ne`
`\pdfdict_item:nn`
`\pdfdict_item:ne`

```
⟨*package⟩
\cs_new:Npn \__pdfdict_item:nn #1 #2 %#1 name, #2 value
  {
    \tl_if_blank:nF {#2} { /#1~#2~ }
  }
\cs_generate_variant:Nn \__pdfdict_item:nn {ne}
\cs_set_eq:NN \pdfdict_item:nn \__pdfdict_item:nn
\cs_generate_variant:Nn \pdfdict_item:nn {ne}
⟨/package⟩
```

(*End definition for* `\__pdfdict_item:nn` *and* `\pdfdict_item:nn`*. This function is documented on page 2.*)

`\__pdfdict_use:n`

`\__pdfdict_use:n` outputs a prop as needed in a dictionary: as a list of /⟨*key*⟩ ⟨*value*⟩ pairs.

```
⟨*package⟩
```

```
\cs_new:Npn \__pdfdict_use:n #1  %#1 dict
  {
    \prop_map_function:cN { \__pdfdict_name:n { #1 } } \__pdfdict_item:ne
  }

\cs_set_eq:NN \pdfdict_use:n \__pdfdict_use:n
⟨/package⟩
```

(*End definition for* `\__pdfdict_use:n`.)

# 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.