

The l3pdfdict package – tools for PDF dictionaries

The L^AT_EX3 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 a 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

<code>\pdfdict_new:n</code>	<code>\pdfdict_new:n {<dictionary name>}</code>
-----------------------------	---

Updated: 2020-12-03

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`.

<code>\pdfdict_set_eq:nn</code>	<code>\pdfdict_set_eq:nn {<local dictionary name₁>} {<dictionary name₂>}</code>
<code>\pdfdict_gset_eq:nn</code>	<code>\pdfdict_gset_eq:nn {<global dictionary name₁>} {<dictionary name₂>}</code>

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

This functions copy `{<dictionary name2>}` into `{<local/global dictionary name1>}` locally or globally. If the dictionary `{<local/global dictionary name1>}` doesn't exist yet, it will be created. If `{<dictionary name2>}` doesn't exist yet, an error will be raised.

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

<hr/> <code>\pdfdict_name:n *</code> <hr/>	<code>\pdfdict_name:n {<dictionary name>}</code>
<hr/> Updated: 2020-12-03 <hr/>	These command expand to the internal name. With these commands it is possible to use standard commands of the <code>prop</code> module to deal with the dictionaries. For example <code>\prop_clear:c { \pdfdict_name:n { name } }</code>
<hr/> <code>\pdfdict_put:nnn</code> <code>\pdfdict_gput:nnn</code> <hr/>	<code>\pdfdict_put:nnn {<local dictionary>} {<name>} {<value>}</code> <code>\pdfdict_gput:nnn {<global dictionary>} {<name>} {<value>}</code>
<hr/> New: 2020-04-06 <hr/>	This function puts <code>{<name>}</code> <code>{<value>}</code> locally or globally in the <code>{<dictionary>}</code> created with <code>\pdfdict_new:n</code> . <code>{<name>}</code> should be a PDF Name without the starting slash. It will be stored with <code>\str_convert_pdfname</code> , so will be automatically correctly escaped in case it contains slashes, spaces or other chars not allowed in a PDF name. <code>{<value>}</code> should be a valid PDF value for this name in the target dictionary. The value is <i>neither</i> converted <i>nor</i> 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 <code>{<value>}</code> to be expanded nor not expanded by the backend commands.
<hr/> <code>\pdfdict_item:nn *</code> <code>pdfdict_item:ne *</code> <hr/>	<code>\pdfdict_item:nn {<key>} {<value>}</code> A simple command to output key-value as <code>/key value</code> . 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.
<hr/> New: 2020-12-04 <hr/>	
<hr/> <code>\pdfdict_use:n *</code> <hr/>	<code>\pdfdict_use:n {<dictionary>}</code>
<hr/> Updated: 2020-12-03 <hr/>	This outputs the property list of the dictionary as a list of <code>/key value</code> pairs. This can be used e.g. when writing a dictionary object with <code>\pdf_object_write:nx</code>
<hr/> <code>\pdfdict_show:n</code> <hr/>	<code>\pdfdict_show:n {<dictionary>}</code>
<hr/> Updated: 2020-12-03 <hr/>	This shows the content of <code>{<dictionary>}</code> in the log and on the terminal.
<hr/> <code>\pdfdict_if_exist_p:n *</code> <code>\pdfdict_if_exist:nTF *</code> <hr/>	<code>\pdfdict_if_exist:n {<dictionary>}</code> This tests if the dictionary exists.
<hr/> Updated: 2020-12-03 <hr/>	
<hr/> <code>\pdfdict_if_empty_p:n *</code> <code>\pdfdict_if_empty:nTF *</code> <hr/>	<code>\pdfdict_if_empty:n {<dictionary>}</code> This tests if the dictionary is empty. The result is false if the dictionary doesn't exist.
<hr/> Updated: 2020-12-03 <hr/>	
<hr/> <code>\pdfdict_get:nnN</code> <hr/>	<code>\pdfdict_get:nnN {<dictionary>} {<name>} <tl var></code>
<hr/> New: 2020-07-06 <hr/>	Recovers the <code><value></code> stored by <code>\pdfdict_put:nnn</code> or <code>\pdfdict_gput:nnn</code> for <code><name></code> and places this in the <code><token list variable></code> . If <code><name></code> is not found then the <code><token list variable></code> is set to the special marker <code>\q_no_value</code> . <code><name></code> is first converted with <code>\str_convert_pdfname</code> . The <code><token list variable></code> is set within the current \TeX group.

```

\pdfdict_remove:nn \pdfdict_remove:nn {\local dictionary} {\name}
\pdfdict_gremove:nn \pdfdict_gremove:nn {\global dictionary} {\name}

```

Updated: 2020-12-03

Removes $\langle name \rangle$ and its associated $\langle value \rangle$ from the $\{\langle dictionary \rangle\}$. The removal is local from local dictionaries and global from global dictionaries. If $\langle name \rangle$ 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. $\{\langle name \rangle\}$ is first converted with `\str_convert_pdfname`.

1.2 Predeclared dictionaries

The module predeclares a number of local dictionaries and fills them with some standard values. Some are not more than templates. But others are used in core commands and can be used to change their settings. For example an link created with `\pdfannot-link:nnn` will use the `annot/Link/URI` dictionary. By adding or changing entries in this dictionary user or packages can for example adapt the border color or add keys needed for ocg layers.

The following tabular summarize the predeclared dictionaries.

name	default keys	notes
<code>l_an-not/Link/Goto</code>	F (Flag)	used in GoTo links (internal references)
<code>l_an-not/Link/URI</code>	F (Flag)	used in URI links (external uri references)
<code>l_an-not/Link/GotoR</code>	F (Flag)	used in GoToR links (file references)
<code>l_an-not/Link/Named</code>	F (Flag)	used in Named actions (menu calls)
<code>l_an-not/Link/Launch</code>	F (Flag)	used in Launch links (application calls)

2 l3pdfdict implementation

```

1 <@@=pdfdict>
2 <*package>
3 \ProvidesExplPackage {l3pdfdict} {2020-12-03} {0.6}
4   {Tools for PDF dictionaries}
5 </package>

```

2.1 messages

```

6 <*package>
7 \cs_new:Npn \__pdfdict_get_type:n #1
8   {
9     \str_case:e:nn { \str_head:n{#1} }
10    {
11      {g}{global}
12      {l}{local}
13    }
14  }
15 \msg_new:nnn { pdfdict } { show-dict }
16   { %1: name of the dictionary
17     %2: expanded content
18     %3: type

```

```

19   The~#3-dictionary~'~#1'~
20   \tl_if_empty:nTF {#2}
21   { is-empty \>~ . }
22   { contains~the~pairs~(without~outer~braces): #2 . }
23 }
24 \msg_new:nnn { pdfdict } { unknown-dict }
25 {
26   The~dictionary~'~#1'~is-unknown.
27 }
28 \msg_new:nnn { pdfdict } { dict-already-defined }
29 {
30   The~#2-dictionary~'~#1'~is-already~defined.
31 }
32 \msg_new:nnn { pdfdict } { empty-value }
33 { The~value~#1~for~#2~is~blank~and~will~be~ignored }
34
35 \msg_new:nnn { pdfdict } { invalid-name }
36 { Name~'~#1'~is~not~valid\\
37   Names~of~dictionaries~should~start~with~'g_ '~or~'l_ ' }
38

```

2.2 Creating dictionaries

`\g__pdfdict_names_seq` Two seq to store the used names for diagnostics.
`\g__pdfdict_gnames_seq`

```

39 \<package>
40 \seq_new:N \g__pdfdict_lnames_seq
41 \seq_new:N \g__pdfdict_gnames_seq
42 \</package>

```

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

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

```

\__pdfdict_new:n
\pdfdict_new:n
43 \<package>
44 \cs_new:Npn \__pdfdict_name:n #1 % #1 dictionary name
45 {
46   \str_head:n{#1}__pdfdict_/#1_prop
47 }
48 \cs_set_eq:NN \pdfdict_name:n \__pdfdict_name:n
49
50 \cs_new_protected:Npn \__pdfdict_new:n #1
51 {
52   \__pdfdict_if_exist:nTF { #1 }
53   {
54     \msg_error:nnxx
55     { pdfdict }
56     { dict-already-defined }
57     { \tl_to_str:n {#1} }
58     { \__pdfdict_get_type:n{#1} }
59   }
60   {
61     \str_case_e:nnF { \str_head:n{#1} }
62     {
63       {g}

```

```

64         {
65             \prop_new:c { \__pdfdict_name:n { #1 } }
66             \seq_gput_right:cn {g__pdfdict_gnames_seq} { #1 }
67         }
68         {1}
69         {
70             \prop_new:c { \__pdfdict_name:n { #1 } }
71             \seq_gput_right:cn {g__pdfdict_lnames_seq} { #1 }
72         }
73     }
74     {
75         \msg_error:nnx{pdfdict}{invalid-name}{\tl_to_str:n{#1}}
76     }
77 }
78 }
79
80 \cs_set_eq:NN \pdfdict_new:n \__pdfdict_new:n
81
82 \</package>

```

(End definition for __pdfdict_name:n and others. These functions are documented on page 2.)

```

\__pdfdict_set_eq:nn
\pdfdict_set_eq:nn
\__pdfdict_gset_eq:nn
\pdfdict_gset_eq:nn
83 \< *package>
84 \cs_new_protected:Npn \__pdfdict_set_eq:nn #1 #2
85 {
86     \__pdfdict_if_exist:nTF { #2 }
87     {
88         \__pdfdict_if_exist:nF { #1 }
89         {
90             \__pdfdict_new:n { #1 }
91         }
92         \prop_set_eq:cc { \__pdfdict_name:n {#1} } { \__pdfdict_name:n {#2} }
93     }
94     {
95         \msg_error:nnn { pdfdict } { unknown-dict } { #1 }
96     }
97 }
98
99 \cs_set_eq:NN \pdfdict_set_eq:nn \__pdfdict_set_eq:nn
100
101 \cs_new_protected:Npn \__pdfdict_gset_eq:nn #1 #2
102 {
103     \__pdfdict_if_exist:nTF { #2 }
104     {
105         \__pdfdict_if_exist:nF { #1 }
106         {
107             \__pdfdict_new:n { #1 }
108         }
109         \prop_gset_eq:cc { \__pdfdict_name:n {#1} } { \__pdfdict_name:n {#2} }
110     }
111     {
112         \msg_error:nnn { pdfdict } { unknown-dict } { #1 }
113     }

```

```

114 }
115
116 \cs_set_eq:NN \pdfdict_gset_eq:nn \__pdfdict_gset_eq:nn
117 \endpackage

```

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

```

\__pdfdict_if_exist:n Existence tests.
\pdfdict_if_exist:n
118 \begin{package}
119 %local
120 \prg_new_conditional:Npnn \__pdfdict_if_exist:n #1 { p , T , F , TF }
121 {
122   \prop_if_exist:cTF
123   { \__pdfdict_name:n { #1 } }
124   { \prg_return_true: }
125   { \prg_return_false: }
126 }
127 \prg_set_eq_conditional:NNn
128 \pdfdict_if_exist:n \__pdfdict_if_exist:n { p , T , F , TF }
129
130 \endpackage

```

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

```

\__pdfdict_if_empty:n Tests for emptiness.
\pdfdict_if_empty:n
131 \begin{package}
132 \prg_new_conditional:Npnn \__pdfdict_if_empty:n #1 { p , T , F , TF }
133 {
134   \prop_if_empty:cTF
135   { \__pdfdict_name:n { #1 } }
136   { \prg_return_true: }
137   { \prg_return_false: }
138 }
139
140 \prg_set_eq_conditional:NNn
141 \pdfdict_if_empty:n \__pdfdict_if_empty:n { p , T , F , TF }
142
143 \endpackage

```

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

```

\__pdfdict_put:nnn These are the commands to store values into the dictionaries. The main difference to
\pdfdict_put:nnn adding values to a normal property list is, that the keys are converted with \str_
\__pdfdict_gput:nnn convert_pdfname:n and that empty values are ignored.
\pdfdict_gput:nnn
144 \begin{package}
145 \cs_new_protected:Npn \__pdfdict_put:nnn #1 #2 #3 %#1 (local) dict, #2 name, #3 value
146 {
147   \tl_if_blank:nTF { #3 }
148   {
149     \msg_warning:nnnn { pdfdict }{ empty-value }{ #2 } { #1 }
150   }
151   {
152     \__pdfdict_if_exist:nTF { #1 }

```

```

153     {
154         \exp_args:Nnx \prop_put:cnn
155         { \_pdfdict_name:n { #1 } } { \str_convert_pdfname:n { #2 } } { #3 }
156     }
157     {
158         \msg_error:nnn { pdfdict } { unknown-dict } { #1 }
159     }
160 }
161 }
162
163
164 \cs_set_eq:NN \pdfdict_put:nnn \_pdfdict_put:nnn
165 \cs_generate_variant:Nn \pdfdict_put:nnn {nnx, nno}
166
167 \cs_new_protected:Npn \_pdfdict_gput:nnn #1 #2 #3 % #1 global dict, #2 name, #3 value
168 {
169     \tl_if_empty:nTF { #3 }
170     {
171         \msg_warning:nnnn { pdfdict } { empty-value } { #2 } { #1 }
172     }
173     {
174         \_pdfdict_if_exist:nTF { #1 }
175         {
176             \exp_args:Nnx \prop_gput:cnn
177             { \_pdfdict_name:n { #1 } } { \str_convert_pdfname:n { #2 } } { #3 }
178         }
179         {
180             \msg_error:nnn { pdfdict } { unknown-dict } { #1 }
181         }
182     }
183 }
184
185 \cs_set_eq:NN \pdfdict_gput:nnn \_pdfdict_gput:nnn
186 \cs_generate_variant:Nn \pdfdict_gput:nnn {nnx, nno}
187 \end{package}

```

(End definition for _pdfdict_put:nnn and others. These functions are documented on page 2.)

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

```

\pdfdict_get:nnN
188 \begin{package}
189 \cs_new_protected:Npn \_pdfdict_get:nnN #1 #2 #3 %dict, key, macro
190 {
191     \_pdfdict_if_exist:nTF { #1 }
192     {
193         \exp_args:Nnx \prop_get:cnN
194         { \_pdfdict_name:n { #1 } }
195         { \str_convert_pdfname:n { #2 } } { #3 }
196     }
197     {
198         \msg_error:nnn { pdfdict } { unknown-dict } { #1 }
199     }
200 }
201
202 \cs_set_eq:NN \pdfdict_get:nnN \_pdfdict_get:nnN

```

```

203
204 </package>

```

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

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

```

\pdfdict_remove:nn
\_pdfdict_gremove:nn
\pdfdict_gremove:nn
205 <*package>
206
207 \cs_new_protected:Npn \_pdfdict_remove:nn #1 #2 %dict,name
208 {
209   \_pdfdict_if_exist:nTF { #1 }
210   {
211     \exp_args:Nnx \prop_remove:cn
212     { \_pdfdict_name:n { #1 } } { \str_convert_pdfname:n { #2 } }
213   }
214   {
215     \msg_error:nnn { pdfdict } { unknown-dict } { #1 }
216   }
217 }
218 \cs_set_eq:NN \pdfdict_remove:nn \_pdfdict_remove:nn
219
220 \cs_new_protected:Npn \_pdfdict_gremove:nn #1 #2 %dict,name
221 {
222   \_pdfdict_if_exist:nTF { #1 }
223   {
224     \exp_args:Nnx \prop_gremove:cn
225     { \_pdfdict_name:n { #1 } } { \str_convert_pdfname:n { #2 } }
226   }
227   {
228     \msg_error:nnn { pdfdict } { unknown-dict } { #1 }
229   }
230 }
231
232 \cs_set_eq:NN \pdfdict_gremove:nn \_pdfdict_gremove:nn
233 </package>

```

(End definition for `_pdfdict_remove:nn` and others. These functions are documented on page 3.)

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

```

\pdfdict_show:n
234 <*package>
235 \cs_new_protected:Npn \_pdfdict_show:Nn #1#2 %#1 message command, #2 dict
236 {
237   \prop_if_exist:cTF { \_pdfdict_name:n { #2 } }
238   {
239     #1
240     { pdfdict }
241     { show-dict }
242     { \tl_to_str:n {#2} }
243     { \prop_map_function:cn { \_pdfdict_name:n { #2 } } \msg_show_item:nn }
244     { \_pdfdict_get_type:n{#2} }
245     { }

```



```

246     }
247     {
248     #1 { pdfdict } { unknown-dict } { #2 } {}{}{}
249     }
250 }
251 \cs_new_protected:Npn \pdfdict_show:n #1
252 {
253   \__pdfdict_show:Nn \msg_show:nnxxxx {#1}
254 }
255 \end{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
256 \begin{package}
257 \cs_new:Npn \__pdfdict_item:nn #1 #2 %#1 name, #2 value
258 {
259   \tl_if_blank:nF {#2} { /#1~#2~ }
260 }
261 \cs_generate_variant:Nn \__pdfdict_item:nn {ne}
262 \cs_set_eq:NN \pdfdict_item:nn \__pdfdict_item:nn
263 \cs_generate_variant:Nn \pdfdict_item:nn {ne}
264 \end{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.

```

265 \begin{package}
266 \cs_new:Npn \__pdfdict_use:n #1 %#1 dict
267 {
268   \prop_map_function:cN { \__pdfdict_name:n { #1 } } \__pdfdict_item:ne
269 }
270
271 \cs_set_eq:NN \pdfdict_use:n \__pdfdict_use:n
272 \end{package}

```

(End definition for __pdfdict_use:n.)

2.3 Predeclared dictionaries

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols

C

cs commands:

\\ 21, 36 \cs_generate_variant:Nn

..... 165, 186, 261, 263
 \cs_new:Npn 7, 44, 257, 266
 \cs_new_protected:Npn ... 50, 84,
 101, 145, 167, 189, 207, 220, 235, 251
 \cs_set_eq:NN 48, 80, 99,
 116, 164, 185, 202, 218, 232, 262, 271

E

exp commands:
 \exp_args:Nnx . 154, 176, 193, 211, 224

M

msg commands:
 \msg_error:nnn
 .. 75, 95, 112, 158, 180, 198, 215, 228
 \msg_error:nnnn 54
 \msg_new:nnn 15, 24, 28, 32, 35
 \msg_show:nnnnnn 253
 \msg_show_item:nn 243
 \msg_warning:nnnn 149, 171

P

pdf commands:
 \pdf_object_write:nn 2

pdfannot commands:
 \pdfannot_link:nnn 3

pdfdict commands:
 \pdfdict_get:nnN 2, 188, 202
 \pdfdict_gput:nnn . 2, 2, 144, 185, 186
 \pdfdict_gremove:nn 3, 205, 232
 \pdfdict_gset_eq:nn 1, 83, 116
 \pdfdict_if_empty:n 2, 131, 141
 \pdfdict_if_empty:nTF 2
 \pdfdict_if_empty_p:n 2
 \pdfdict_if_exist:n 2, 118, 128
 \pdfdict_if_exist:nTF 2
 \pdfdict_if_exist_p:n 2
 \pdfdict_item:nn 2, 256, 262, 263
 pdfdict_item:nn 2
 \pdfdict_name:n 2, 43, 48
 \pdfdict_new:n 1, 2, 43, 80
 \pdfdict_put:nnn .. 2, 2, 144, 164, 165
 \pdfdict_remove:nn 3, 205, 218
 \pdfdict_set_eq:nn 1, 83, 99
 \pdfdict_show:n 2, 234, 251
 \pdfdict_use:n 2, 271

pdfdict internal commands:
 __pdfdict_get:nnN 188, 189, 202
 __pdfdict_get_type:n 7, 58, 244
 \g__pdfdict_gnames_seq 39
 __pdfdict_gput:nnn ... 144, 167, 185
 __pdfdict_gremove:nn . 205, 220, 232
 __pdfdict_gset_eq:nn .. 83, 101, 116
 __pdfdict_if_empty:n . 131, 132, 141
 __pdfdict_if_exist:n . 118, 120, 128

__pdfdict_if_exist:nTF 52,
 86, 88, 103, 105, 152, 174, 191, 209, 222
 __pdfdict_item:nn
 256, 257, 261, 262, 268
 \g__pdfdict_lnames_seq 40
 __pdfdict_name:n 43,
 44, 48, 65, 70, 92, 109, 123, 135,
 155, 177, 194, 212, 225, 237, 243, 268
 \g__pdfdict_names_seq 39
 __pdfdict_new:n .. 43, 50, 80, 90, 107
 __pdfdict_put:nnn 144, 145, 164
 __pdfdict_remove:nn .. 205, 207, 218
 __pdfdict_set_eq:nn 83, 84, 99
 __pdfdict_show:Nn 234, 235, 253
 __pdfdict_use:n 9, 265, 266, 271

prg commands:
 \prg_new_conditional:Npnn .. 120, 132
 \prg_return_false: 125, 137
 \prg_return_true: 124, 136
 \prg_set_eq_conditional:NNn 127, 140

prop commands:
 \prop_get:NnN 193
 \prop_gput:Nnn 176
 \prop_gremove:Nn 224
 \prop_gset_eq:NN 109
 \prop_if_empty:NTF 134
 \prop_if_exist:NTF 122, 237
 \prop_map_function:NN 243, 268
 \prop_new:N 65, 70
 \prop_put:Nnn 154
 \prop_remove:Nn 211
 \prop_set_eq:NN 92
 \ProvidesExplPackage 3

Q

quark commands:
 \q_no_value 2

S

seq commands:
 \seq_gput_right:Nn 66, 71
 \seq_new:N 40, 41

str commands:
 \str_case_e:nn 9
 \str_case_e:nnTF 61
 \str_convert_pdfname 1, 2, 2, 3
 \str_convert_pdfname:n
 6, 155, 177, 195, 212, 225
 \str_head:n 9, 46, 61

T

tl commands:
 \tl_if_blank:nTF 147, 259
 \tl_if_empty:nTF 20, 169
 \tl_to_str:n 57, 75, 242