

The **tagpdf-roles** module

Tags, roles and namespace code

part of the tagpdf package

Ulrike Fischer*

Version 0.82, released 2021-06-14

```
1 <@@=tag>
2 <*roles>
3 \ProvidesExplPackage {tagpdf-roles-code} {2021-06-14} {0.82}
4 {part of tagpdf - code related to roles and structure names}
```

1 Code related to roles and structure names

1.1 Variables

Tags have both a name (a string) and a number (for the lua attribute). Testing a name is easier with a prop, while accessing with a number is better done with a seq. So both are used and must be kept in sync if a new tag is added. The number is only relevant for the MC type, tags with the same name from different names spaces can have the same number.

```
\g__tag_role_tags_seq
\g__tag_role_tags_prop      5 \__tag_seq_new:N \g__tag_role_tags_seq %to get names (type/NS) from numbers
                             6 \__tag_prop_new:N \g__tag_role_tags_prop %to get numbers from names (type/NS)
```

(End definition for \g__tag_role_tags_seq and \g__tag_role_tags_prop.)

`\g__tag_role_tags_NS_prop` in pdf 2.0 tags belong to a name space. For every tag we store a default name space. The keys are the tags, the value shorthands like pdf2, or mathml. There is no need to access this from lua, so we use the standard prop commands.

```
7 \prop_new:N \g__tag_role_tags_NS_prop %to namespace info
```

(End definition for \g__tag_role_tags_NS_prop.)

`\g__tag_role_NS_prop` The standard names spaces are the following. The keys are the name tagpdf will use, the urls are the identifier in the namespace object.

mathml <http://www.w3.org/1998/Math/MathML>

pdf2 <http://iso.org/pdf2/ssn>

pdf <http://iso.org/pdf/ssn> (default)

*E-mail: fischer@troubleshooting-tex.de

`user \c__tag_role_userNS_id_str` (random id, for user tags)

More namespaces are possible and their objects references and the ones of the namespaces must be collected so that an array can be written to the StructTreeRoot at the end (see tagpdf-tree). We use a prop to store also the object reference as it will be needed rather often.

```
8 \prop_new:N \g__tag_role_NS_prop % collect namespaces
```

(End definition for `\g__tag_role_NS_prop`.)

We need also a bunch of temporary variables:

```
\l__tag_role_tag_tmpa_tl
\l__tag_role_tag_namespace_tmpa_tl
\l__tag_role_role_tmpa_tl
\l__tag_role_role_namespace_tmpa_tl
9 \tl_new:N \l__tag_role_tag_tmpa_tl
10 \tl_new:N \l__tag_role_tag_namespace_tmpa_tl
11 \tl_new:N \l__tag_role_role_tmpa_tl
12 \tl_new:N \l__tag_role_role_namespace_tmpa_tl
```

(End definition for `\l__tag_role_tag_tmpa_tl` and others.)

1.2 Namespaces

The following commands setups a names space. Namespace dictionaries can contain an optional /Schema and /RoleMapNS entry. We only reserve the objects but delay the writing to the finish code, where we can test if the keys and the name spaces are actually needed This commands setups objects for the name space and its rolemap. It also initialize a prop to collect the rolemaps if needed.

```
\__tag_role_NS_new:nnn \__tag_role_NS_new:nnn{(shorthand)}{(URI-ID)}Schema
```

```
\__tag_role_NS_new:nnn
13 \cs_new_protected:Npn \__tag_role_NS_new:nnn #1 #2 #3
14 {
15   \msg_redirect_name:nnn { pdfdict } { empty-value } { none }
16   \pdf_object_new:nn {tag/NS/#1}{dict}
17   \pdfdict_new:n {g__tag_role/Namespase_#1_dict}
18   \pdf_object_new:nn {\__tag/RoleMapNS/#1}{dict}
19   \pdfdict_new:n {g__tag_role/RoleMapNS_#1_dict}
20   \pdfdict_gput:nnn
21     {g__tag_role/Namespase_#1_dict}
22     {Type}
23     {/Namespase}
24   \pdf_string_from_unicode:nnN{utf8/string}{#2}\l_tmpa_str
25   \pdfdict_gput:nnx
26     {g__tag_role/Namespase_#1_dict}
27     {NS}
28     {\l_tmpa_str}
29   %RoleMapNS is added in tree
30   \pdfdict_gput:nnx{g__tag_role/Namespase_#1_dict}
31     {Schema}{#3}
32   \prop_gput:Nnx \g__tag_role_NS_prop {#1}{\pdf_object_ref:n{tag/NS/#1}~}
33   \msg_redirect_name:nnn { pdfdict } { empty-value } { warning }
34 }
```

(End definition for `_tag_role_NS_new:nnn`.)

We need an id for the user space. For the tests it should be possible to set it to a fix value. So we use random numbers which can be fixed by setting a seed. We fake a sort of GUID but not try to be really exact as it doesn't matter ...

`\c__tag_role_userNS_id_str`

```

35 \str_const:Nx \c__tag_role_userNS_id_str
36 { data:,
37   \int_to_Hex:n{\int_rand:n {65535}}
38   \int_to_Hex:n{\int_rand:n {65535}}
39   -
40   \int_to_Hex:n{\int_rand:n {65535}}
41   -
42   \int_to_Hex:n{\int_rand:n {65535}}
43   -
44   \int_to_Hex:n{\int_rand:n {65535}}
45   -
46   \int_to_Hex:n{\int_rand:n {16777215}}
47   \int_to_Hex:n{\int_rand:n {16777215}}
48 }

```

(End definition for `\c__tag_role_userNS_id_str`.)

Now we setup the standard names spaces. Currently only if we detect pdf2.0 but this will perhaps have to change if the structure code gets to messy.

```

49 \pdf_version_compare:NnT > {1.9}
50 {
51   \_tag_role_NS_new:nnn {pdf}    {http://iso.org/pdf/ssn}{ }
52   \_tag_role_NS_new:nnn {pdf2}  {http://iso.org/pdf2/ssn}{ }
53   \_tag_role_NS_new:nnn {mathml}{http://www.w3.org/1998/Math/MathML}{ }
54   \exp_args:Nnx
55   \_tag_role_NS_new:nnn {user}{\c__tag_role_userNS_id_str}{ }
56 }

```

1.3 Data

In this section we setup the standard data. At first the list of structure types. We split them in three lists, the tags with which are both in the pdf and pdf2 namespace, the one only in pdf and the one with the tags only in pdf2. We also define a rolemap for the pdfII only type to pdf so that they can always be used.

```

\c__tag_role_sttags_pdf_pdfII_clist
\c__tag_role_sttags_only_pdf_clist
\c__tag_role_sttags_only_pdfII_clist
\c__tag_role_sttags_mathml_clist
\c__tag_role_sttags_pdfII_to_pdf_prop
57 %
58 \clist_const:Nn \c__tag_role_sttags_pdf_pdfII_clist
59 {
60   Document,    %A complete document. This is the root element
61                %of any structure tree containing
62                %multiple parts or multiple articles.
63   Part,        %A large-scale division of a document.
64   Sect,        %A container for grouping related content elements.
65   Div,         %A generic block-level element or group of elements
66   Caption,     %A brief portion of text describing a table or figure.
67   Index,
68   NonStruct,   %probably not needed

```

```

69   H,
70   H1,
71   H2,
72   H3,
73   H4,
74   H5,
75   H6,
76   P,
77   L,           %list
78   LI,          %list item (around label and list item body)
79   Lbl,         %list label
80   LBody,       %list item body
81   Table,
82   TR,          %table row
83   TH,          %table header cell
84   TD,          %table data cell
85   THead,       %table header (n rows)
86   TBody,       %table rows
87   TFoot,       %table footer
88   Span,        %generic inline marker
89   Link,        %
90   Annot,
91   Figure,
92   Formula,
93   Form,
94   % ruby warichu etc ..
95   Ruby,
96   RB,
97   RT,
98   Warichu,
99   WT,
100  WP,
101  Artifact % only MC-tag ?...
102  }
103
104  \clist_const:Nn \c__tag_role_sttags_only_pdf_clist
105  {
106    Art,          %A relatively self-contained body of text
107                  %constituting a single narrative or exposition
108    BlockQuote,  %A portion of text consisting of one or more paragraphs
109                  %attributed to someone other than the author of the
110                  %surrounding text.
111    TOC,         %A list made up of table of contents item entries
112                  %(structure tag TOCI; see below) and/or other
113                  %nested table of contents entries
114    TOCI,        %An individual member of a table of contents.
115                  %This entry's children can be any of the following structure tags:
116                  %Lbl,Reference,NonStruct,P,TOC
117    Index,
118    Private,
119    Quote,       %inline quote
120    Note,        %footnote, endnote. Lbl can be child
121    Reference,   %A citation to content elsewhere in the document.
122    BibEntry,    %bibentry

```

```

123     Code
124 }
125
126 \clist_const:Nn \c__tag_role_sttags_only_pdfII_clist
127 {
128     DocumentFragment
129     ,Aside
130     ,H7
131     ,H8
132     ,H9
133     ,H10
134     ,Title
135     ,FENote
136     ,Sub
137     ,Em
138     ,Strong
139     ,Artifact
140 }
141
142 \clist_const:Nn \c__tag_role_sttags_mathml_clist
143 {
144     abs
145     ,and
146     ,annotation
147     ,apply
148     ,approx
149     ,arccos
150     ,arccosh
151     ,arccot
152     ,arccoth
153     ,arccsc
154     ,arccsch
155     ,arcsec
156     ,arcsech
157     ,arcsin
158     ,arcsinh
159     ,arctan
160     ,arctanh
161     ,arg
162     ,bind
163     ,bvar
164     ,card
165     ,cartesianproduct
166     ,cbytes
167     ,ceiling
168     ,error
169     ,ci
170     ,cn
171     ,codomain
172     ,complexes
173     ,compose
174     ,condition
175     ,conjugate
176     ,cos

```

```

177 ,cosh
178 ,cot
179 ,coth
180 ,cs
181 ,csc
182 ,csch
183 ,csymbol
184 ,curl
185 ,declare
186 ,degree
187 ,determinant
188 ,diff
189 ,divergence
190 ,divide
191 ,domain
192 ,domainofapplication
193 ,emptyset
194 ,eq
195 ,equivalent
196 ,eulergamma
197 ,exists
198 ,exp
199 ,exponentiale
200 ,factorial
201 ,factorof
202 ,false
203 ,floor
204 ,fn
205 ,forall
206 ,gcd
207 ,geq
208 ,grad
209 ,gt
210 ,ident
211 ,image
212 ,imaginary
213 ,imaginaryi
214 ,implies
215 ,in
216 ,infinity
217 ,int
218 ,integers
219 ,intersect
220 ,interval
221 ,inverse
222 ,lambda
223 ,laplacian
224 ,lcm
225 ,leq
226 ,limit
227 ,ln
228 ,log
229 ,logbase
230 ,lowlimit

```

231 ,lt
 232 ,maction
 233 ,maligngroup
 234 ,malignmark
 235 ,math
 236 ,matrix
 237 ,matrixrow
 238 ,max
 239 ,mean
 240 ,median
 241 ,menclose
 242 ,merror
 243 ,mfenced
 244 ,mfrac
 245 ,mglyph
 246 ,mi
 247 ,min
 248 ,minus
 249 ,mlabeledtr
 250 ,mlongdiv
 251 ,mmultiscripts
 252 ,mn
 253 ,mo
 254 ,mode
 255 ,moment
 256 ,momentabout
 257 ,mover
 258 ,mpadded
 259 ,mphantom
 260 ,mprescripts
 261 ,mroot
 262 ,mrow
 263 ,ms
 264 ,mscarries
 265 ,mscarry
 266 ,msgroup
 267 ,msline
 268 ,mspace
 269 ,msqrt
 270 ,msrow
 271 ,mstack
 272 ,mstyle
 273 ,msub
 274 ,msubsup
 275 ,msup
 276 ,mtable
 277 ,mtd
 278 ,mtext
 279 ,mtr
 280 ,munder
 281 ,munderover
 282 ,naturalnumbers
 283 ,neq
 284 ,none

```

285 ,not
286 ,notanumber
287 ,notin
288 ,notprsubset
289 ,notsubset
290 ,or
291 ,otherwise
292 ,outerproduct
293 ,partialdiff
294 ,pi
295 ,piece
296 ,piecewise
297 ,plus
298 ,power
299 ,primes
300 ,product
301 ,prsubset
302 ,quotient
303 ,rationals
304 ,real
305 ,reals
306 ,reln
307 ,rem
308 ,root
309 ,scalarproduct
310 ,sdev
311 ,sec
312 ,sech
313 ,selector
314 ,semantics
315 ,sep
316 ,set
317 ,setdiff
318 ,share
319 ,sin
320 ,sinh
321 ,subset
322 ,sum
323 ,tan
324 ,tanh
325 ,tendsto
326 ,times
327 ,transpose
328 ,true
329 ,union
330 ,uplimit
331 ,variance
332 ,vector
333 ,vectorproduct
334 ,xor
335 }
336
337 \prop_const_from_keyval:Nn \c__tag_role_sttags_pdfII_to_pdf_prop
338 {

```



```

339 DocumentFragment = Art,
340 Aside = Note,
341 Title = H1,
342 Sub = Span,
343 H7 = H6 ,
344 H8 = H6 ,
345 H9 = H6 ,
346 H10 = H6,
347 FENote= Note,
348 Em = Span,
349 Strong= Span,
350 }

```

(End definition for `\c__tag_role_sttags_pdf_pdfII_clist` and others.)

We fill the structure tags in to the seq. We allow all pdf1.7 and pdf2.0, and role map if needed the 2.0 tags.

```

351 % get tag name from number: \seq_item:Nn \g__tag_role_tags_seq { n }
352 % get tag number from name: \prop_item:Nn \g__tag_role_tags_prop { name }
353
354 \clist_map_inline:Nn \c__tag_role_sttags_pdf_pdfII_clist
355 {
356   \__tag_seq_gput_right:Nn \g__tag_role_tags_seq { #1 }
357   \prop_gput:Nnn \g__tag_role_tags_NS_prop { #1 }{ pdf2 }
358 }
359 \clist_map_inline:Nn \c__tag_role_sttags_only_pdf_clist
360 {
361   \__tag_seq_gput_right:Nn \g__tag_role_tags_seq { #1 }
362   \prop_gput:Nnn \g__tag_role_tags_NS_prop { #1 }{ pdf }
363 }
364 \clist_map_inline:Nn \c__tag_role_sttags_only_pdfII_clist
365 {
366   \__tag_seq_gput_right:Nn \g__tag_role_tags_seq { #1 }
367   \prop_gput:Nnn \g__tag_role_tags_NS_prop { #1 }{ pdf2 }
368 }
369 \pdf_version_compare:NnT > {1.9}
370 {
371   \clist_map_inline:Nn \c__tag_role_sttags_mathml_clist
372   {
373     \__tag_seq_gput_right:Nn \g__tag_role_tags_seq { #1 }
374     \prop_gput:Nnn \g__tag_role_tags_NS_prop { #1 }{ mathml }
375   }
376 }

```

For luatex and the MC we need a name/number relation. The name space is not relevant.

```

377 \int_step_inline:nnnn { 1 }{ 1 }{ \seq_count:N \g__tag_role_tags_seq }
378 {
379   \__tag_prop_gput:Nxn \g__tag_role_tags_prop
380   {
381     \seq_item:Nn \g__tag_role_tags_seq { #1 }
382   }
383   { #1 }
384 }

```

1.4 Adding new tags and rolemapping

1.4.1 pdf 1.7 and earlier

With this versions only RoleMap is filled. At first the dictionary:

`g__tag_role/RoleMap_dict`

```
385 \pdfdict_new:n {g__tag_role/RoleMap_dict}
```

(End definition for g__tag_role/RoleMap_dict.)

`__tag_role_add_tag:nn`

The pdf 1.7 version has only two arguments: new and rolemap name. To make pdf 2.0 types usable we directly define a rolemapping for them.

```
386 \cs_new_protected:Nn \__tag_role_add_tag:nn %(new) name, reference to old
387 {
388   \prop_if_in:NnF \g__tag_role_tags_prop {#1}
389   {
390     \msg_info:nnn { tag }{new-tag}{#1}
391     \__tag_seq_gput_right:Nn \g__tag_role_tags_seq { #1 }
392     \__tag_prop_gput:Nnx \g__tag_role_tags_prop { #1 }
393     {
394       \seq_count:N \g__tag_role_tags_seq
395     }
396     \prop_gput:Nnn \g__tag_role_tags_NS_prop { #1 }{ user }
397   }
398   \__tag_check_add_tag_role:nn {#1}{#2}
399   \tl_if_empty:nF { #2 }
400   {
401     \pdfdict_gput:nnx {g__tag_role/RoleMap_dict}
402     {#1}
403     {\pdf_name_from_unicode_e:n{#2}}
404   }
405 }
406 \cs_generate_variant:Nn \__tag_role_add_tag:nn {VV}
407
408 \pdf_version_compare:NnT < {2.0}
409 {
410   \prop_map_inline:Nn \c__tag_role_sttags_pdfII_to_pdf_prop
411   {
412     \__tag_role_add_tag:nn {#1}{#2}
413   }
414 }
415
```

(End definition for __tag_role_add_tag:nn.)

1.4.2 The pdf 2.0 version

`__tag_role_add_tag:nnnn`

The pdf 2.0 version takes four arguments: tag/namespace/role/namespace

```
416 \cs_new_protected:Nn \__tag_role_add_tag:nnnn %tag/namespace/role/namespace
417 {
418   \msg_info:nnn { tag }{new-tag}{#1}
419   \__tag_seq_gput_right:Nn \g__tag_role_tags_seq { #1 }
420   \__tag_prop_gput:Nnx \g__tag_role_tags_prop { #1 }
421   {
```

```

422     \seq_count:N \g__tag_role_tags_seq
423   }
424   \prop_gput:Nnn \g__tag_role_tags_NS_prop { #1 }{ #2 }
425   \__tag_check_add_tag_role:nn {#1}{#3}
426   \pdfdict_gput:nnx {g__tag_role/RoleMapNS_#2_dict}{#1}
427   {
428     [
429       \pdf_name_from_unicode_e:n{#3}
430       \c_space_tl
431       \pdf_object_ref:n {tag/NS/#4}
432     ]
433   }
434 }
435 \cs_generate_variant:Nn \__tag_role_add_tag:nnnn {VVVV}

```

(End definition for __tag_role_add_tag:nnnn.)

1.5 Key-val user interface

The user interface use the key `add-new-tag`, which takes either a keyval list as argument, or a tag/role.

```

tag
tag-namespace
role
role-namespace
add-new-tag
436 \keys_define:nn { __tag / tag-role }
437 {
438   ,tag .tl_set:N = \l__tag_role_tag_tmpa_tl
439   ,tag-namespace .tl_set:N = \l__tag_role_tag_namespace_tmpa_tl
440   ,role .tl_set:N = \l__tag_role_role_tmpa_tl
441   ,role-namespace .tl_set:N = \l__tag_role_role_namespace_tmpa_tl
442 }
443
444 \keys_define:nn { __tag / setup }
445 {
446   add-new-tag .code:n =
447   {
448     \keys_set_known:nnnN
449     {__tag/tag-role}
450     {
451       tag-namespace=user,
452       role-namespace=, %so that we can test for it.
453       #1
454     }{__tag/tag-role}\l_tmpa_tl
455     \tl_if_empty:NF \l_tmpa_tl
456     {
457       \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq { / } {\l_tmpa_tl/}
458       \tl_set:Nx \l__tag_role_tag_tmpa_tl { \seq_item:Nn \l_tmpa_seq {1} }
459       \tl_set:Nx \l__tag_role_role_tmpa_tl { \seq_item:Nn \l_tmpa_seq {2} }
460     }
461     \tl_if_empty:NT \l__tag_role_role_namespace_tmpa_tl
462     {
463       \prop_get:NVNTF
464       \g__tag_role_tags_NS_prop
465       \l__tag_role_role_tmpa_tl
466       \l__tag_role_role_namespace_tmpa_tl

```

```

467         {
468             \prop_if_in:NVF\g__tag_role_NS_prop \l__tag_role_role_namespace_tmpa_tl
469             {
470                 \tl_set:Nn \l__tag_role_role_namespace_tmpa_tl {user}
471             }
472         }
473         {
474             \tl_set:Nn \l__tag_role_role_namespace_tmpa_tl {user}
475         }
476     }
477     \pdf_version_compare:NnTF < {2.0}
478     {
479         %TODO add check for emptyness?
480         \__tag_role_add_tag:VV
481             \l__tag_role_tag_tmpa_tl
482             \l__tag_role_role_tmpa_tl
483     }
484     {
485         \__tag_role_add_tag:VVVV
486             \l__tag_role_tag_tmpa_tl
487             \l__tag_role_tag_namespace_tmpa_tl
488             \l__tag_role_role_tmpa_tl
489             \l__tag_role_role_namespace_tmpa_tl
490     }
491 }
492 }
493 </roles>

```

(End definition for tag and others. These functions are documented on page ??.)

Index

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

A		I	
add-new-tag	436	int commands:	
		\int_rand:n ..	37 , 38 , 40 , 42 , 44 , 46 , 47
		\int_step_inline:nnnn	377
		\int_to_Hex:n	37 , 38 , 40 , 42 , 44 , 46 , 47
C		K	
clist commands:		keys commands:	
\clist_const:Nn	58 , 104 , 126 , 142	\keys_define:nn	436 , 444
\clist_map_inline:Nn	354 , 359 , 364 , 371	\keys_set_known:nnnN	448
cs commands:			
\cs_generate_variant:Nn	406 , 435		
\cs_new_protected:Nn	386 , 416		
\cs_new_protected:Npn	13		
E		M	
exp commands:		msg commands:	
\exp_args:NNno	457	\msg_info:nnn	390 , 418
\exp_args:Nnx	54	\msg_redirect_name:nnn	15 , 33

P

pdf commands:

\pdf_name_from_unicode_e:n . 403, 429
 \pdf_object_new:nn 16, 18
 \pdf_object_ref:n 32, 431
 \pdf_string_from_unicode:nnN . . . 24
 \pdf_version_compare:NnTF
 49, 369, 408, 477

pdfdict commands:

\pdfdict_gput:nnn . 20, 25, 30, 401, 426
 \pdfdict_new:n 17, 19, 385

prop commands:

\prop_const_from_keyval:Nn 337
 \prop_get:NnNTF 463
 \prop_gput:Nnn
 32, 357, 362, 367, 374, 396, 424
 \prop_if_in:NnTF 388, 468
 \prop_item:Nn 352
 \prop_map_inline:Nn 410
 \prop_new:N 7, 8
 \ProvidesExplPackage 3

R

role 436
 role-namespace 436

S

seq commands:

\seq_count:N 377, 394, 422
 \seq_item:Nn 351, 381, 458, 459
 \seq_set_split:Nnn 457
 \l_tmpa_seq 457, 458, 459

str commands:

\str_const:Nn 35
 \l_tmpa_str 24, 28

T

tag 436

tag internal commands:

__tag_check_add_tag_role:nn 398, 425
 __tag_prop_gput:Nnn . . 379, 392, 420
 __tag_prop_new:N 6
 g__tag_role/RoleMap_dict 385
 __tag_role_add_tag:nn
 386, 386, 406, 412, 480

__tag_role_add_tag:nnnn
 416, 416, 435, 485

__tag_role_NS_new:nnn
 2, 13, 13, 51, 52, 53, 55

\g__tag_role_NS_prop 8, 32, 468

\l__tag_role_role_namespace_-
 tmpa_tl 9,
 441, 461, 466, 468, 470, 474, 489

\l__tag_role_role_tmpa_tl
 9, 440, 459, 465, 482, 488

\c__tag_role_sttags_mathml_clist
 57, 371

\c__tag_role_sttags_only_pdf_-
 clist 57, 359

\c__tag_role_sttags_only_pdfII_-
 clist 57, 364

\c__tag_role_sttags_pdf_pdfII_-
 clist 57, 354

\c__tag_role_sttags_pdfII_to_-
 pdf_prop 57, 410

\l__tag_role_tag_namespace_tmpa_-
 tl 9, 439, 487

\l__tag_role_tag_tmpa_tl
 9, 438, 458, 481, 486

\g__tag_role_tags_NS_prop
 . . 7, 357, 362, 367, 374, 396, 424, 464

\g__tag_role_tags_prop
 5, 352, 379, 388, 392, 420

\g__tag_role_tags_seq
 5, 351, 356, 361,
 366, 373, 377, 381, 391, 394, 419, 422

\c__tag_role_userNS_id_str . 2, 35, 55

__tag_seq_gput_right:Nn
 356, 361, 366, 373, 391, 419

__tag_seq_new:N 5

tag-namespace 436

tl commands:

\c_space_tl 430

\tl_if_empty:NnTF 455, 461

\tl_if_empty:nTF 399

\tl_new:N 9, 10, 11, 12

\tl_set:Nn 458, 459, 470, 474

\l_tmpa_tl 454, 455, 457