tagpdf – A package to experiment with pdf tagging[*]

Ulrike Fischer[†]

Released 2022-01-09

Contents

1 Initialization and test if pdfmanagement is active. 5

2 Package options 5

3 Packages 6

4 Temporary code 6
4.1 a LastPage label 6

5 Variables 7

6 Variants of l3 commands 8

7 Setup label attributes 8

8 Label commands 9

9 Commands to fill seq and prop 9

10 General tagging commands 10

11 Keys for tagpdfsetup 10

12 loading of engine/more dependent code 11

I The tagpdf-checks module
Messages and check code
Part of the tagpdf package 13

1 Commands 13

[*]This file describes v0.93, last revised 2022-01-09.
[†]E-mail: fischer@troubleshooting-tex.de

2 Description of log messages **13**
2.1 `\ShowTagging` command . . . 13
2.2 Messages in checks and commands . . . 13
2.3 Messages from the ptagging code . . . 14
2.4 Warning messages from the lua-code . . . 14
2.5 Info messages from the lua-code . . . 14
2.6 Debug mode messages and code . . . 15
2.7 Messages . . . 15

3 Messages **16**
3.1 Messages related to mc-chunks . . . 16
3.2 Messages related to mc-chunks . . . 17
3.3 Attributes . . . 18
3.4 Roles . . . 18
3.5 Miscellaneous . . . 18

4 Retrieving data **19**

5 User conditionals **19**

6 Internal checks **19**
6.1 checks for active tagging . . . 19
6.2 Checks related to stuctures . . . 20
6.3 Checks related to roles . . . 21
6.4 Check related to mc-chunks . . . 21
6.5 Checks related to the state of MC on a page or in a split stream . . . 24

II The tagpdf-user module
Code related to LaTeX2e user commands and document commands
Part of the tagpdf package **27**

1 Setup commands **27**

2 Commands related to mc-chunks **27**

3 Commands related to structures **28**

4 Debugging **28**

5 Extension commands **28**
5.1 Fake space . . . 29
5.2 Paratagging . . . 29
5.3 Header and footer . . . 29
5.4 Link tagging . . . 30

6 User commands and extensions of document commands **30**

7 Setup and preamble commands **30**

8 Commands for the mc-chunks **30**

9 Commands for the structure **31**

10 Debugging **31**

11 Commands to extend document commands **34**

11.1 Document structure 34
11.2 Structure destinations 35
11.3 Fake space 35
11.4 Paratagging 36
11.5 Header and footer 38
11.6 Links 40

Index **42**

`\ref_value:nnn` `\ref_value:nnn{⟨label⟩}{⟨attribute⟩}{⟨fallback default⟩}`

This is a temporary definition which will have to move to l3ref. It allows to locally set a default value if the label or the attribute doesn't exist. See issue #4 in Accessible-xref.

`\tag_stop_group_begin:`
`\tag_stop_group_end:`

We need a command to stop tagging in some places. This simply switches the two local booleans.

`activate-space␣(setup-key)`

`activate-space` activates the additional parsing needed for interword spaces. It is not documented, the parsing is currently implicitly activated by the known key `interwordspace`, as the code will perhaps move to some other place, now that it is better separated.

`activate-mc␣(setup-key)`
`activate-tree␣(setup-key)`
`activate-struct␣(setup-key)`
`activate-all␣(setup-key)`

Keys to activate the various tagging steps

`no-struct-dest␣(setup-key)`

The key allows to suppress the creation of structure destinations

`log␣(setup-key)`

The `log` takes currently the values `none`, `v`, `vv`, `vvv`, `all`. More details are in `tagpdf-checks`.

`tagunmarked␣(setup-key)`

This key allows to set if (in luamode) unmarked text should be marked up as artifact. The initial value is true.

`tabsorder␣(setup-key)`

This sets the tabsorder on a page. The values are `row`, `column`, `structure` (default) or `none`. Currently this is set more or less globally. More finer control can be added if needed.

`tagstruct`
`tagstructobj`
`tagabspage`
`tagmcabs`
`tagmcid`

These are attributes used by the label/ref system.

1 Initialization and test if pdfmanagement is active.

```
⟨@@=tag⟩
⟨*package⟩
\ProvidesExplPackage {tagpdf} {2022-01-09} {0.93}
  { A package to experiment with pdf tagging }

\bool_if:nF
  {
    \bool_lazy_and_p:nn
      {\cs_if_exist_p:N \pdfmanagement_if_active_p:}
      { \pdfmanagement_if_active_p: }
  }
  {  %error for now, perhaps warning later.
    \PackageError{tagpdf}
     {
       PDF~resource~management~is~no~active!\MessageBreak
       tagpdf~will~no~work.
     }
     {
       Activate~it~with \MessageBreak
       \string\RequirePackage{pdfmanagement-testphase}\MessageBreak
       \string\DeclareDocumentMetadata{<options>}\MessageBreak
       before~\string\documentclass
     }
  }
⟨/package⟩
```

<*debug>

```
\ProvidesExplPackage {tagpdf-debug} {2022-01-09} {0.93}
  { debug code for tagpdf }
\@ifpackageloaded{tagpdf}{}{\PackageWarning{tagpdf-debug}{tagpdf~not~loaded,~quitting}\endinp
    \end{macrocode}
⟨/debug⟩
% We map the internal module name \enquote{tag} to \enquote{tagpdf} in messages.
%    \begin{macrocode}
⟨*package⟩
\prop_gput:Nnn \g_msg_module_name_prop { tag }{ tagpdf }
⟨/package⟩
```

Debug mode has its special mapping:

```
⟨*debug⟩
\prop_gput:Nnn \g_msg_module_type_prop { tag / debug} {}
\prop_gput:Nnn \g_msg_module_name_prop { tag / debug }{tagpdf~DEBUG}
⟨/debug⟩
```

2 Package options

There are only two options to switch for luatex between generic and luamode, TODO try to get rid of them.

```
⟨*package⟩
\bool_new:N\g__tag_mode_lua_bool
\DeclareOption {luamode}    { \sys_if_engine_luatex:T { \bool_gset_true:N \g__tag_mode_lua_bo
\DeclareOption {genericmode}{ \bool_gset_false:N\g__tag_mode_lua_bool }
\ExecuteOptions{luamode}
```

```
\ProcessOptions
```

3 Packages

We need the temporary version of l3ref until this is in the kernel.

```
\RequirePackage{l3ref-tmp}
```

4 Temporary code

This is code which will be removed when proper support exists in LaTeX

4.1 a LastPage label

See also issue #2 in Accessible-xref

__tag_lastpagelabel:

```
\cs_new_protected:Npn \__tag_lastpagelabel:
  {
    \legacy_if:nT { @filesw }
      {
        \exp_args:NNnx \exp_args:NNx\iow_now:Nn \@auxout
           {
             \token_to_str:N \newlabeldata
               {__tag_LastPage}
               {
                 {abspage} { \int_use:N \g_shipout_readonly_int}
                 {tagmcabs}{ \int_use:N \c@g__tag_MCID_abs_int }
               }
           }
      }
  }

\AddToHook{enddocument/afterlastpage}
 {\__tag_lastpagelabel:}
```

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

\ref_value:nnn This allows to locally set a default value if the label or the attribute doesn't exist.

```
\cs_if_exist:NF \ref_value:nnn
  {
    \cs_new:Npn \ref_value:nnn #1#2#3
      {
        \exp_args:Nee
          \__ref_value:nnn
           { \tl_to_str:n {#1} } { \tl_to_str:n {#2} } {#3}
      }
    \cs_new:Npn \__ref_value:nnn #1#2#3
      {
        \tl_if_exist:cTF { g__ref_label_ #1 _ #2 _tl }
          { \tl_use:c { g__ref_label_ #1 _ #2 _tl } }
          {
            #3
          }
        }
    }
```

(*End definition for* `\ref_value:nnn`. *This function is documented on page 4.*)

5 Variables

\l__tag_tmpa_tl
\l__tag_tmpa_str
\l__tag_tmpa_prop
\l__tag_tmpa_seq
\l__tag_tmpb_seq
\l__tag_tmpa_clist
\l__tag_tmpa_int
\l__tag_tmpa_box
\l__tag_tmpb_box

A few temporary variables

```
\tl_new:N    \l__tag_tmpa_tl
\str_new:N   \l__tag_tmpa_str
\prop_new:N  \l__tag_tmpa_prop
\seq_new:N   \l__tag_tmpa_seq
\seq_new:N   \l__tag_tmpb_seq
\clist_new:N \l__tag_tmpa_clist
\int_new:N   \l__tag_tmpa_int
\box_new:N   \l__tag_tmpa_box
\box_new:N   \l__tag_tmpb_box
```

(*End definition for* `\l__tag_tmpa_tl` *and others.*)

Attribute lists for the label command. We have a list for mc-related labels, and one for structures.

\c__tag_refmc_clist
\c__tag_refstruct_clist

```
\clist_const:Nn \c__tag_refmc_clist     {tagabspage,tagmcabs,tagmcid}
\clist_const:Nn \c__tag_refstruct_clist {tagstruct,tagstructobj}
```

(*End definition for* `\c__tag_refmc_clist` *and* `\c__tag_refstruct_clist`.)

\l__tag_loglevel_int

This integer hold the log-level and so allows to control the messages. TODO: a list which log-level shows what is needed. The current behaviour is quite ad-hoc.

```
\int_new:N  \l__tag_loglevel_int
```

(*End definition for* `\l__tag_loglevel_int`.)

\g__tag_active_space_bool
\g__tag_active_mc_bool
\g__tag_active_tree_bool
\g__tag_active_struct_bool
\g__tag_active_struct_dest_bool

These booleans should help to control the global behaviour of tagpdf. Ideally it should more or less do nothing if all are false. The space-boolean controles the interword space code, the mc-boolean activates `\tag_mc_begin:n`, the tree-boolean activates writing the finish code and the pdfmanagement related commands, the struct-boolean activates the storing of the structure data. In a normal document all should be active, the split is only there for debugging purpose. Structure destination will be activated automatically if pdf version 2.0 is detected, but with the boolean struct-dest-boolean one can suppress them. Also we assume currently that they are set only at begin document. But if some control passing over groups are needed they could be perhaps used in a document too. TODO: check if they are used everywhere as needed and as wanted.

```
\bool_new:N \g__tag_active_space_bool
\bool_new:N \g__tag_active_mc_bool
\bool_new:N \g__tag_active_tree_bool
\bool_new:N \g__tag_active_struct_bool
\bool_new:N \g__tag_active_struct_dest_bool
\bool_gset_true:N \g__tag_active_struct_dest_bool
```

(*End definition for* `\g__tag_active_space_bool` *and others.*)

\l__tag_active_mc_bool
\l__tag_active_struct_bool

These booleans should help to control the *local* behaviour of tagpdf. In some cases it could e.g. be necessary to stop tagging completely. As local booleans they respect groups. TODO: check if they are used everywhere as needed and as wanted.

```
\bool_new:N \l__tag_active_mc_bool
\bool_set_true:N \l__tag_active_mc_bool
\bool_new:N \l__tag_active_struct_bool
\bool_set_true:N \l__tag_active_struct_bool
```

(*End definition for* `\l__tag_active_mc_bool` *and* `\l__tag_active_struct_bool`*.*)

\g__tag_tagunmarked_bool

This boolean controls if the code should try to automatically tag parts not in mc-chunk. It is currently only used in luamode. It would be possible to used it in generic mode, but this would create quite a lot empty artifact mc-chunks.

```
\bool_new:N \g__tag_tagunmarked_bool
```

(*End definition for* `\g__tag_tagunmarked_bool`*.*)

6 Variants of l3 commands

```
\prg_generate_conditional_variant:Nnn \pdf_object_if_exist:n {e}{T,F}
\cs_generate_variant:Nn \pdf_object_ref:n {e}
\cs_generate_variant:Nn \pdfannot_dict_put:nnn {nnx}
\cs_generate_variant:Nn \pdffile_embed_stream:nnn {nxx,oxx}
\cs_generate_variant:Nn \prop_gput:Nnn {Nxx}
\cs_generate_variant:Nn \prop_put:Nnn  {Nxx}
\cs_generate_variant:Nn \ref_label:nn { nv }
\cs_generate_variant:Nn \seq_set_split:Nnn{Nne}
\cs_generate_variant:Nn \str_set_convert:Nnnn {Nonn, Noon, Nnon }
```

7 Setup label attributes

tagstruct
tagstructobj
tagabspage
tagmcabs
tagmcid

This are attributes used by the label/ref system. With structures we store the structure number `tagstruct` and the object reference `tagstructobj`. The second is needed to be able to reference a structure which hasn't been created yet. The alternative would be to create the object in such cases, but then we would have to check the object existence all the time.

With mc-chunks we store the absolute page number `tagabspage`, the absolute id `tagmcabc`, and the id on the page `tagmcid`.

```
\ref_attribute_gset:nnnn { tagstruct } {0} { now }
  { \int_use:N \c@g__tag_struct_abs_int }
\ref_attribute_gset:nnnn { tagstructobj } {} { now }
  {
    \pdf_object_if_exist:eT {__tag/struct/\int_use:N \c@g__tag_struct_abs_int}
      {
        \pdf_object_ref:e{__tag/struct/\int_use:N \c@g__tag_struct_abs_int}
      }
  }
\ref_attribute_gset:nnnn { tagabspage } {0} { shipout }
  { \int_use:N \g_shipout_readonly_int }
\ref_attribute_gset:nnnn { tagmcabs } {0} { now }
  { \int_use:N \c@g__tag_MCID_abs_int }
\ref_attribute_gset:nnnn {tagmcid }  {0} { now }
  { \int_use:N \g__tag_MCID_tmp_bypage_int }
```

(*End definition for* `tagstruct` *and others. These functions are documented on page 4.*)

8 Label commands

`\__tag_ref_label:nn`

A version of `\ref_label:nn` to set a label which takes a keyword `mc` or `struct` to call the relevant lists. TODO: check if `\@bsphack` and `\@esphack` make sense here.

```
\cs_new_protected:Npn \__tag_ref_label:nn #1 #2 %#1 label, #2 name of list mc or struct
  {
    \@bsphack
    \ref_label:nv {#1}{c__tag_ref#2_clist}
    \@esphack
  }
\cs_generate_variant:Nn \__tag_ref_label:nn {en}
```

(*End definition for* `\__tag_ref_label:nn`.)

`\__tag_ref_value:nnn`

A local version to retrieve the value. It is a direct wrapper, but to keep naming consistent .... It uses the variant defined temporarly above.

```
\cs_new:Npn \__tag_ref_value:nnn #1 #2 #3 %#1 label, #2 attribute, #3 default
  {
    \ref_value:nnn {#1}{#2}{#3}
  }
\cs_generate_variant:Nn \__tag_ref_value:nnn {enn}
```

(*End definition for* `\__tag_ref_value:nnn`.)

`\__tag_ref_value_lastpage:nn`

A command to retrieve the lastpage label, this will be adapted when there is a proper, kernel lastpage label.

```
\cs_new:Npn \__tag_ref_value_lastpage:nn #1 #2
  {
    \ref_value:nnn {__tag_LastPage}{#1}{#2}
  }
```

(*End definition for* `\__tag_ref_value_lastpage:nn`.)

9 Commands to fill seq and prop

With most engines these are simply copies of the expl3 commands, but luatex will overwrite them, to store the data also in lua tables.

`\__tag_prop_new:N`
`\__tag_seq_new:N`
`\__tag_prop_gput:Nnn`
`\__tag_seq_gput_right:Nn`
`\__tag_seq_item:cn`
`\__tag_prop_item:cn`
`\__tag_seq_show:N`
`\__tag_prop_show:N`

```
\cs_set_eq:NN \__tag_prop_new:N        \prop_new:N
\cs_set_eq:NN \__tag_seq_new:N         \seq_new:N
\cs_set_eq:NN \__tag_prop_gput:Nnn     \prop_gput:Nnn
\cs_set_eq:NN \__tag_seq_gput_right:Nn \seq_gput_right:Nn
\cs_set_eq:NN \__tag_seq_item:cn       \seq_item:cn
\cs_set_eq:NN \__tag_prop_item:cn      \prop_item:cn
\cs_set_eq:NN \__tag_seq_show:N        \seq_show:N
\cs_set_eq:NN \__tag_prop_show:N       \prop_show:N

\cs_generate_variant:Nn \__tag_prop_gput:Nnn      { Nxn , Nxx, Nnx , cnn, cxn, cnx, cno}
\cs_generate_variant:Nn \__tag_seq_gput_right:Nn  { Nx  , No, cn, cx }
```

```
\cs_generate_variant:Nn \__tag_prop_new:N   { c }
\cs_generate_variant:Nn \__tag_seq_new:N    { c }
\cs_generate_variant:Nn \__tag_seq_show:N   { c }
\cs_generate_variant:Nn \__tag_prop_show:N  { c }
```

(*End definition for* `\__tag_prop_new:N` *and others.*)

10 General tagging commands

`\tag_stop_group_begin:`
`\tag_stop_group_end:`

We need a command to stop tagging in some places. This simply switches the two local booleans.

```
\cs_new_protected:Npn \tag_stop_group_begin:
  {
    \group_begin:
    \bool_set_false:N \l__tag_active_struct_bool
    \bool_set_false:N \l__tag_active_mc_bool
  }
\cs_set_eq:NN \tag_stop_group_end: \group_end:
```

(*End definition for* `\tag_stop_group_begin:` *and* `\tag_stop_group_end:`*. These functions are documented on page 4.*)

11 Keys for tagpdfsetup

TODO: the log-levels must be sorted

`activate-space␣(setup-key)`
`activate-mc␣(setup-key)`
`activate-tree␣(setup-key)`
`activate-struct␣(setup-key)`
`activate-all␣(setup-key)`
`no-struct-dest␣(setup-key)`

Keys to (globally) activate tagging. `activate-space` activates the additional parsing needed for interword spaces. It is not documented, the parsing is currently implicitly activated by the known key `interwordspace`, as the code will perhaps move to some other place, now that it is better separated. `no-struct-dest` allows to suppress structure destinations.

```
\keys_define:nn { __tag / setup }
  {
    activate-space  .bool_gset:N = \g__tag_active_space_bool,
    activate-mc     .bool_gset:N = \g__tag_active_mc_bool,
    activate-tree   .bool_gset:N = \g__tag_active_tree_bool,
    activate-struct .bool_gset:N = \g__tag_active_struct_bool,
    activate-all    .meta:n =
      {activate-mc={#1},activate-tree={#1},activate-struct={#1}},
    activate-all   .default:n = true,
    no-struct-dest .bool_gset_inverse:N = \g__tag_active_struct_dest_bool,

```

(*End definition for* `activate-space (setup-key)` *and others. These functions are documented on page 4.*)

`log␣(setup-key)`

The `log` takes currently the values `none`, `v`, `vv`, `vvv`, `all`. The description of the log levels is in tagpdf-checks.

```
    log             .choice:,
    log / none      .code:n = {\int_set:Nn \l__tag_loglevel_int { 0 }},
    log / v         .code:n =
      {
        \int_set:Nn \l__tag_loglevel_int { 1 }
        \cs_set_protected:Nn \__tag_check_typeout_v:n { \iow_term:x {##1} }
      },
    log / vv        .code:n = {\int_set:Nn \l__tag_loglevel_int { 2 }},
    log / vvv       .code:n = {\int_set:Nn \l__tag_loglevel_int { 3 }},
    log / all       .code:n = {\int_set:Nn \l__tag_loglevel_int { 10 }},
```

(*End definition for* `log (setup-key)`. *This function is documented on page 4.*)

`tagunmarked␣(setup-key)` This key allows to set if (in luamode) unmarked text should be marked up as artifact. The initial value is true.

```
    tagunmarked     .bool_gset:N = \g__tag_tagunmarked_bool,
    tagunmarked     .initial:n  = true,
```

(*End definition for* `tagunmarked (setup-key)`. *This function is documented on page 4.*)

`tabsorder␣(setup-key)` This sets the tabsorder on a page. The values are `row`, `column`, `structure` (default) or `none`. Currently this is set more or less globally. More finer control can be added if needed.

```
    tabsorder       .choice:,
    tabsorder / row       .code:n =
      \pdfmanagement_add:nnn { Page } {Tabs}{/R},
    tabsorder / column    .code:n =
      \pdfmanagement_add:nnn { Page } {Tabs}{/C},
    tabsorder / structure .code:n =
      \pdfmanagement_add:nnn { Page } {Tabs}{/S},
    tabsorder / none      .code:n =
      \pdfmanagement_remove:nn {Page} {Tabs},
    tabsorder       .initial:n = structure,
    uncompress      .code:n = { \pdf_uncompress:  },
  }
```

(*End definition for* `tabsorder (setup-key)`. *This function is documented on page 4.*)

12 loading of engine/more dependent code

```
\sys_if_engine_luatex:T
  {
    \file_input:n {tagpdf-luatex.def}
  }
⟨/package⟩
⟨*mcloading⟩
\bool_if:NTF \g__tag_mode_lua_bool
  {
   \RequirePackage {tagpdf-mc-code-lua}
  }
  {
   \RequirePackage {tagpdf-mc-code-generic} %
  }
⟨/mcloading⟩
⟨*debug⟩
\bool_if:NTF \g__tag_mode_lua_bool
  {
    \RequirePackage {tagpdf-debug-lua}
   }
   {
    \RequirePackage {tagpdf-debug-generic} %
   }
⟨/debug⟩
```

Part I
The **tagpdf-checks** module Messages and check code Part of the tagpdf package

1 Commands

`\tag_if_active_p:` ⋆
`\tag_if_active:TF` ⋆

This command tests if tagging is active. It only gives true if all tagging has been activated, *and* if tagging hasn't been stopped locally.

`\tag_get:n` ⋆

`\tag_get:n{⟨keyword⟩}`

This is a generic command to retrieve data. Currently the only sensible values for the argument ⟨*keyword*⟩ are `mc_tag` and `struct_tag`.

2 Description of log messages

2.1 `\ShowTagging` command

Argument	type	note
`\ShowTagging`mc-data = num	log+term	lua-only
`\ShowTagging`mc-current	log+term	
`\ShowTagging`struck-stack= [log|show]	log or term+stop	

2.2 Messages in checks and commands

command	message	action
`\@@_check_structure_has_tag:n`	`struct-missing-tag`	error
`\@@_check_structure_tag:N`	`role-unknown-tag`	warning
`\@@_check_info_closing_struct:n`	`struct-show-closing`	info
`\@@_check_no_open_struct:`	struct-faulty-nesting	error
`\@@_check_struct_used:n`	`struct-used-twice`	warning
`\@@_check_add_tag_role:nn`	`role-missing`, `role-tag`, `role-unknown`	warning, info (>0), warning
`\@@_check_mc_if_nested:`,	`mc-nested`	warning
`\@@_check_mc_if_open:`	`mc-not-open`	warning
`\@@_check_mc_pushed_popped:nn`	`mc-pushed`, `mc-popped`	info (2), info+`seq_log` (>2)
`\@@_check_mc_tag:N`	`mc-tag-missing`, `role-unknown-tag`	error (missing), warning (unknown).
`\@@_check_mc_used:n`	`mc-used-twice`	warning
`\@@_check_show_MCID_by_page:`		
`\tag_mc_use:n`	`mc-label-unknown`, `mc-used-twice`	warning
`\role_add_tag:nn`	`new-tag`	info (>0)
	`sys-no-interwordspace`	warning
`\@@_struct_write_obj:n`	`struct-no-objnum`	error
`\tag_struct_begin:n`	`struct-faulty-nesting`	error
`\@@_struct_insert_annot:nn`	`struct-faulty-nesting`	error
`tag_struct_use:n`	`struct-label-unknown`	warning
`attribute-class`, `attribute`	`attr-unknown`	error
`\@@_tree_fill_parenttree:`	`tree-mcid-index-wrong`	warning TODO: should trigger a standard rerun m
in `enddocument/info-hook`	`para-hook-count-wrong`	error (warning?)

2.3 Messages from the ptagging code

A few messages are issued in generic mode from the code which reinserts missing TMB/TME. This is currently done if log-level is larger than zero. TODO: reconsider log-level and messages when this code settles down.

2.4 Warning messages from the lua-code

The messages are triggered if the log-level is at least equal to the number.

message	log-level	remark
WARN TAG-NOT-TAGGED:	1	
WARN TAG-OPEN-MC:	1	
WARN SHIPOUT-MC-OPEN:	1	
WARN SHIPOUT-UPS:	0	shouldn't happen
WARN TEX-MC-INSERT-MISSING:	0	shouldn't happen
WARN TEX-MC-INSERT-NO-KIDS:	2	e.g. from empty hbox

2.5 Info messages from the lua-code

The messages are triggered if the log-level is at least equal to the number. `TAG` messages are from the traversing function, `TEX` from code used in the tagpdf-mc module. `PARENTREE` is the code building the parenttree.

message	log-level	remark
INFO SHIPOUT-INSERT-LAST-EMC	3	finish of shipout code
INFO SPACE-FUNCTION-FONT	3	interwordspace code
INFO TAG-ABSPAGE	3	
INFO TAG-ARGS	4	
INFO TAG-ENDHEAD	4	
INFO TAG-ENDHEAD	4	
INFO TAG-HEAD	3	
INFO TAG-INSERT-ARTIFACT	3	
INFO TAG-INSERT-BDC	3	
INFO TAG-INSERT-EMC	3	
INFO TAG-INSERT-TAG	3	
INFO TAG-KERN-SUBTYPE	4	
INFO TAG-MATH-SUBTYPE	4	
INFO TAG-MC-COMPARE	4	
INFO TAG-MC-INTO-PAGE	3	
INFO TAG-NEW-MC-NODE	4	
INFO TAG-NODE	3	
INFO TAG-NO-HEAD	3	
INFO TAG-NOT-TAGGED	2	replaced by artifact
INFO TAG-QUITTING-BOX	4	
INFO TAG-STORE-MC-KID	4	
INFO TAG-TRAVERSING-BOX 3		
INFO TAG-USE-ACTUALTEXT	3	
INFO TAG-USE-ALT	3	
INFO TAG-USE-RAW	3	
INFO TEX-MC-INSERT-KID	3	

message	log-level	remark
INFO TEX-MC-INSERT-KID-TEST	4	
INFO TEX-MC-INTO-STRUCT	3	
INFO TEX-STORE-MC-DATA	3	
INFO TEX-STORE-MC-KID	3	
INFO PARENTTREE-CHUNKS	3	
INFO PARENTTREE-NO-DATA	3	
INFO PARENTTREE-NUM	3	
INFO PARENTTREE-NUMENTRY	3	
INFO PARENTTREE-STRUCT-OBJREF	4	

2.6 Debug mode messages and code

If the package tagpdf-debug is loaded a number of commands are redefined and enhanced with additional commands which can be used to output debug messages or collect statistics. The commands are present but do nothing if the log-level is zero.

command	name	action	remark
`\tag_mc_begin:n`	mc-begin-insert	msg	
	mc-begin-ignore	msg	if inactive

2.7 Messages

`mc-nested`
`mc-tag-missing`
`mc-label-unknown`
`mc-used-twice`
`mc-not-open`
`mc-pushed`
`mc-popped`
`mc-current`

Various messages related to mc-chunks. TODO document their meaning.

`struct-no-objnum`
`struct-faulty-nesting`
`struct-missing-tag`
`struct-used-twice`
`struct-label-unknown`
`struct-show-closing`

Various messages related to structure. TODO document their meaning.

`attr-unknown`

Message if an attribute i sunknown.

`role-missing`
`role-unknown`
`role-unknown-tag`
`role-tag`
`new-tag`

Messages related to role mapping.

`tree-mcid-index-wrong` Used in the tree code, typically indicates the document must be rerun.

`sys-no-interwordspace` Message if an engine doesn't support inter word spaces

`para-hook-count-wrong` Message if the number of begin paragraph and end paragraph differ. This normally means faulty structure.

```
⟨@@=tag⟩
⟨*header⟩
\ProvidesExplPackage {tagpdf-checks-code} {2022-01-09} {0.93}
 {part of tagpdf - code related to checks, conditionals, debugging and messages}
⟨/header⟩
```

3 Messages

3.1 Messages related to mc-chunks

`mc-nested` This message is issue is a mc is opened before the previous has been closed. This is not relevant for luamode, as the attributes don't care about this. It is used in the `\@@_check_mc_if_nested:` test.

```
⟨*package⟩
\msg_new:nnn { tag } {mc-nested} { nested~marked~content~found~-~mcid~#1 }
```

(*End definition for* `mc-nested`. *This function is documented on page 15.*)

`mc-tag-missing` If the tag is missing

```
\msg_new:nnn { tag } {mc-tag-missing} { required~tag~missing~-~mcid~#1 }
```

(*End definition for* `mc-tag-missing`. *This function is documented on page 15.*)

`mc-label-unknown` If the label of a mc that is used in another place is not known (yet) or has been undefined as the mc was already used.

```
\msg_new:nnn { tag } {mc-label-unknown}
  { label~#1~unknown~or~has~been~already~used.\\
    Either~rerun~or~remove~one~of~the~uses. }
```

(*End definition for* `mc-label-unknown`. *This function is documented on page 15.*)

`mc-used-twice` An mc-chunk can be inserted only in one structure. This indicates wrong coding and so should at least give a warning.

```
\msg_new:nnn { tag } {mc-used-twice} { mc~#1~has~been~already~used }
```

(*End definition for* `mc-used-twice`. *This function is documented on page 15.*)

`mc-not-open` This is issued if a `\tag_mc_end:` is issued wrongly, wrong coding.

```
\msg_new:nnn { tag } {mc-not-open} { there~is~no~mc~to~end~at~#1 }
```

(*End definition for* `mc-not-open`. *This function is documented on page 15.*)

mc-pushed
mc-popped

Informational messages about mc-pushing.

```
\msg_new:nnn { tag } {mc-pushed} { #1~has~been~pushed~to~the~mc~stack}
\msg_new:nnn { tag } {mc-popped} { #1~has~been~removed~from~the~mc~stack }
```

(*End definition for* `mc-pushed` *and* `mc-popped`*. These functions are documented on page 15.*)

mc-current

Informational messages about current mc state.

```
\msg_new:nnn { tag } {mc-current}
  { current~MC:~
    \bool_if:NTF\g__tag_in_mc_bool
      {abscnt=\__tag_get_mc_abs_cnt:,~tag=\g__tag_mc_key_tag_tl}
      {no~MC~open,~current~abscnt=\__tag_get_mc_abs_cnt:"}
  }
```

(*End definition for* `mc-current`*. This function is documented on page 15.*)

3.2 Messages related to mc-chunks

struct-no-objnum

Should not happen . . .

```
\msg_new:nnn { tag } {struct-no-objnum} { objnum~missing~for~structure~#1 }
```

(*End definition for* `struct-no-objnum`*. This function is documented on page 15.*)

struct-faulty-nesting

This indicates that there is somewhere one `\tag_struct_end:` too much. This should be normally an error.

```
\msg_new:nnn { tag }
  {struct-faulty-nesting}
  { there~is~no~open~structure~on~the~stack }
```

(*End definition for* `struct-faulty-nesting`*. This function is documented on page 15.*)

struct-missing-tag

A structure must have a tag.

```
\msg_new:nnn { tag } {struct-missing-tag} { a~structure~must~have~a~tag! }
```

(*End definition for* `struct-missing-tag`*. This function is documented on page 15.*)

struct-used-twice

```
\msg_new:nnn { tag } {struct-used-twice}
  { structure~with~label~#1~has~already~been~used}
```

(*End definition for* `struct-used-twice`*. This function is documented on page 15.*)

struct-label-unknown

label is unknown, typically needs a rerun.

```
\msg_new:nnn { tag } {struct-label-unknown}
  { structure~with~label~#1~is~unknown~rerun}
```

(*End definition for* `struct-label-unknown`*. This function is documented on page 15.*)

struct-show-closing

Informational message shown if log-mode is high enough

```
\msg_new:nnn { tag } {struct-show-closing}
  { closing~structure~#1~tagged~\prop_item:cn{g__tag_struct_#1_prop}{S} }
```

(*End definition for* `struct-show-closing`*. This function is documented on page 15.*)

3.3 Attributes

Not much yet, as attributes aren't used so much.

attr-unknown

```
\msg_new:nnn { tag } {attr-unknown}  { attribute~#1~is~unknown}
```

(*End definition for* `attr-unknown`. *This function is documented on page 15.*)

3.4 Roles

role-missing
role-unknown
role-unknown-tag

Warning message if either the tag or the role is missing

```
\msg_new:nnn { tag } {role-missing}     { tag~#1~has~no~role~assigned  }
\msg_new:nnn { tag } {role-unknown}     { role~#1~is~not~known  }
\msg_new:nnn { tag } {role-unknown-tag} { tag~#1~is~not~known  }
```

(*End definition for* `role-missing`, `role-unknown`, *and* `role-unknown-tag`. *These functions are documented on page 15.*)

role-tag
new-tag

Info messages.

```
\msg_new:nnn { tag } {role-tag}         { mapping~tag~#1~to~role~#2  }
\msg_new:nnn { tag } {new-tag}          { adding~new~tag~#1 }
```

(*End definition for* `role-tag` *and* `new-tag`. *These functions are documented on page 15.*)

3.5 Miscellaneous

tree-mcid-index-wrong

Used in the tree code, typically indicates the document must be rerun.

```
\msg_new:nnn { tag } {tree-mcid-index-wrong}
  {something~is~wrong~with~the~mcid--rerun}
```

(*End definition for* `tree-mcid-index-wrong`. *This function is documented on page 16.*)

sys-no-interwordspace

Currently only pdflatex and lualatex have some support for real spaces.

```
\msg_new:nnn { tag } {sys-no-interwordspace}
  {engine/output~mode~#1~doesn't~support~the~interword~spaces}
```

(*End definition for* `sys-no-interwordspace`. *This function is documented on page 16.*)

__tag_check_typeout_v:n

A simple logging function. By default is gobbles its argument, but the log-keys sets it to typeout.

```
\cs_set_eq:NN \__tag_check_typeout_v:n \use_none:n
```

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

para-hook-count-wrong

At the end of the document we check if the count of para-begin and para-end is identical. If not we issue a warning: this is normally a coding error and and breaks the structure.

```
\msg_new:nnnn { tag } {para-hook-count-wrong}
  {The~number~of~automatic~begin~(#1)~and~end~(#2)~para~hooks~differ!}
  {This~quite~probably~a~coding~error~and~the~structure~will~be~wrong!}
```

(*End definition for* `para-hook-count-wrong`. *This function is documented on page 16.*)

4 Retrieving data

\tag_get:n This retrieves some data. This is a generic command to retrieve data. Currently the only sensible values for the argument are `mc_tag` and `struct_tag`.

```
\cs_new:Npn \tag_get:n #1   { \use:c {__tag_get_data_#1: } }
```

(*End definition for* `\tag_get:n`*. This function is documented on page 13.*)

5 User conditionals

\tag_if_active_p:
\tag_if_active:TF This is a test it tagging is active. This allows packages to add conditional code. The test is true if all booleans, the global and the two local one are true.

```
\prg_new_conditional:Npnn \tag_if_active: { p , T , TF, F }
  {
     \bool_lazy_all:nTF
       {
         {\g__tag_active_struct_bool}
         {\g__tag_active_mc_bool}
         {\g__tag_active_tree_bool}
         {\l__tag_active_struct_bool}
         {\l__tag_active_mc_bool}
       }
       {
         \prg_return_true:
       }
       {
         \prg_return_false:
       }
  }
```

(*End definition for* `\tag_if_active:TF`*. This function is documented on page 13.*)

6 Internal checks

These are checks used in various places in the code.

6.1 checks for active tagging

__tag_check_if_active_mc:TF
__tag_check_if_active_struct:TF Structures must have a tag, so we check if the S entry is in the property. It is an error if this is missing. The argument is a number.

```
\prg_new_conditional:Npnn \__tag_check_if_active_mc: {T,F,TF}
  {
    \bool_lazy_and:nnTF { \g__tag_active_mc_bool } { \l__tag_active_mc_bool }
      {
         \prg_return_true:
      }
      {
         \prg_return_false:
      }
  }
\prg_new_conditional:Npnn \__tag_check_if_active_struct: {T,F,TF}
```

```
  {
    \bool_lazy_and:nnTF { \g__tag_active_struct_bool } { \l__tag_active_struct_bool }
      {
         \prg_return_true:
      }
      {
         \prg_return_false:
      }
  }
```

(*End definition for* `\__tag_check_if_active_mc:TF` *and* `\__tag_check_if_active_struct:TF`.)

6.2 Checks related to stuctures

`\__tag_check_structure_has_tag:n`

Structures must have a tag, so we check if the S entry is in the property. It is an error if this is missing. The argument is a number. The tests for existence and type is split in structures, as the tags are stored differently to the mc case.

```
\cs_new_protected:Npn \__tag_check_structure_has_tag:n #1 %#1 struct num
  {
    \prop_if_in:cnF { g__tag_struct_#1_prop }
      {S}
      {
        \msg_error:nn { tag } {struct-missing-tag}
      }
  }
```

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

`\__tag_check_structure_tag:N`

This checks if the name of the tag is known, either because it is a standard type or has been rolemapped.

```
\cs_new_protected:Npn \__tag_check_structure_tag:N #1
  {
    \prop_if_in:NoF \g__tag_role_tags_prop {#1}
      {
       \msg_warning:nnx { tag } {role-unknown-tag} {#1}
      }
  }
```

(*End definition for* `\__tag_check_structure_tag:N`.)

`\__tag_check_info_closing_struct:n`

This info message is issued at a closing structure, the use should be guarded by log-level.

```
\cs_new_protected:Npn \__tag_check_info_closing_struct:n #1 %#1 struct num
  {
    \int_compare:nNnT {\l__tag_loglevel_int} > { 0 }
      {
       \msg_info:nnn { tag } {struct-show-closing} {#1}
      }
  }

\cs_generate_variant:Nn \__tag_check_info_closing_struct:n {o,x}
```

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

__tag_check_no_open_struct: This checks if there is an open structure. It should be used when trying to close a structure. It errors if false.

```
\cs_new_protected:Npn \__tag_check_no_open_struct:
  {
    \msg_error:nn { tag } {struct-faulty-nesting}
  }
```

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

__tag_check_struct_used:n This checks if a stashed structure has already been used.

```
\cs_new_protected:Npn \__tag_check_struct_used:n #1 %#1 label
  {
    \prop_get:cnNT
      {g__tag_struct_\__tag_ref_value:enn{tagpdfstruct-#1}{tagstruct}{unknown}_prop}
      {P}
      \l_tmpa_tl
      {
        \msg_warning:nnn { tag } {struct-used-twice} {#1}
      }
  }
```

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

6.3 Checks related to roles

__tag_check_add_tag_role:nn This check is used when defining a new role mapping.

```
\cs_new_protected:Npn \__tag_check_add_tag_role:nn #1 #2 %#1 tag, #2 role
  {
    \tl_if_empty:nTF {#2}
      {
        \msg_warning:nnn { tag } {role-missing} {#1}
      }
      {
        \prop_get:NnNTF \g__tag_role_tags_prop {#2} \l_tmpa_tl
          {
            \int_compare:nNnT {\l__tag_loglevel_int} > { 0 }
              {
                \msg_info:nnnn { tag } {role-tag} {#1} {#2}
              }
          }
          {
            \msg_warning:nnn { tag } {role-unknown} {#2}
          }
      }
  }
```

(*End definition for* `\__tag_check_add_tag_role:nn`*.*)

6.4 Check related to mc-chunks

__tag_check_mc_if_nested:
__tag_check_mc_if_open: Two tests if a mc is currently open. One for the true (for begin code), one for the false part (for end code).

```
\cs_new_protected:Npn \__tag_check_mc_if_nested:
  {
    \__tag_mc_if_in:T
      {
        \msg_warning:nnx { tag } {mc-nested} { \__tag_get_mc_abs_cnt: }
      }
  }

\cs_new_protected:Npn \__tag_check_mc_if_open:
  {
    \__tag_mc_if_in:F
      {
        \msg_warning:nnx { tag } {mc-not-open} { \__tag_get_mc_abs_cnt: }
      }
  }
```

(*End definition for* `\__tag_check_mc_if_nested:` *and* `\__tag_check_mc_if_open:`*.*)

`\__tag_check_mc_pushed_popped:nn`

This creates an information message if mc's are pushed or popped. The first argument is a word (pushed or popped), the second the tag name. With larger log-level the stack is shown too.

```
\cs_new_protected:Npn \__tag_check_mc_pushed_popped:nn #1 #2
  {
    \int_compare:nNnT
      { \l__tag_loglevel_int } ={ 2 }
      { \msg_info:nnx {tag}{mc-#1}{#2} }
    \int_compare:nNnT
      { \l__tag_loglevel_int } > { 2 }
      {
        \msg_info:nnx {tag}{mc-#1}{#2}
        \seq_log:N \g__tag_mc_stack_seq
      }
  }
```

(*End definition for* `\__tag_check_mc_pushed_popped:nn`*.*)

`\__tag_check_mc_tag:N`

This checks if the mc has a (known) tag.

```
\cs_new_protected:Npn \__tag_check_mc_tag:N #1  %#1 is var with a tag name in it
  {
    \tl_if_empty:NT #1
      {
        \msg_error:nnx { tag } {mc-tag-missing} { \__tag_get_mc_abs_cnt: }
      }
   \prop_if_in:NoF \g__tag_role_tags_NS_prop {#1}
     {
       \msg_warning:nnx { tag } {role-unknown-tag} {#1}
     }
  }
```

(*End definition for* `\__tag_check_mc_tag:N`*.*)

`\g__tag_check_mc_used_intarray`
`\__tag_check_init_mc_used:`

This variable holds the list of used mc numbers. Everytime we store a mc-number we will add one the relevant array index If everything is right at the end there should be only 1 until the max count of the mcid. 2 indicates that one mcid was used twice, 0 that we lost one. In engines other than luatex the total number of all intarray entries are restricted so we use only a rather small value of 65536, and we initialize the array only at first used, guarded by the log-level. This check is probably only needed for debugging. TODO does this really make sense to check? When can it happen??

```
\cs_new_protected:Npn \__tag_check_init_mc_used:
  {
    \intarray_new:Nn \g__tag_check_mc_used_intarray { 65536 }
    \cs_gset_eq:NN \__tag_check_init_mc_used: \prg_do_nothing:
  }
```

(*End definition for* `\g__tag_check_mc_used_intarray` *and* `\__tag_check_init_mc_used:`.)

`\__tag_check_mc_used:n` This checks if a mc is used twice.

```
\cs_new_protected:Npn \__tag_check_mc_used:n #1 %#1 mcid abscnt
  {
    \int_compare:nNnT {\l__tag_loglevel_int} > { 2 }
      {
        \__tag_check_init_mc_used:
        \intarray_gset:Nnn \g__tag_check_mc_used_intarray
          {#1}
          { \intarray_item:Nn \g__tag_check_mc_used_intarray {#1} + 1 }
        \int_compare:nNnT
          {
            \intarray_item:Nn \g__tag_check_mc_used_intarray {#1}
          }
          >
          { 1 }
          {
            \msg_warning:nnn { tag } {mc-used-twice} {#1}
          }
      }
  }
```

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

`\__tag_check_show_MCID_by_page:` This allows to show the mc on a page. Currently unused.

```
\cs_new_protected:Npn \__tag_check_show_MCID_by_page:
  {
    \tl_set:Nx \l__tag_tmpa_tl
      {
        \__tag_ref_value_lastpage:nn
          {abspage}
          {-1}
      }
    \int_step_inline:nnnn {1}{1}
      {
        \l__tag_tmpa_tl
      }
      {
        \seq_clear:N \l_tmpa_seq
        \int_step_inline:nnnn
          {1}
          {1}
          {
            \__tag_ref_value_lastpage:nn
              {tagmcabs}
```

```
                  {-1}
              }
              {
                \int_compare:nT
                  {
                    \__tag_ref_value:enn
                      {mcid-####1}
                      {tagabspage}
                      {-1}
                    =
                    ##1
                  }
                  {
                    \seq_gput_right:Nx \l_tmpa_seq
                      {
                        Page##1-####1-
                        \__tag_ref_value:enn
                          {mcid-####1}
                          {tagmcid}
                          {-1}
                      }
                  }
              }
              \seq_show:N \l_tmpa_seq
          }
      }
```

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

6.5 Checks related to the state of MC on a page or in a split stream

The following checks are currently only usable in generic mode as they rely on the marks defined in the mc-generic module. They are used to detect if a mc-chunk has been split by a page break or similar and additional end/begin commands are needed.

`\__tag_check_mc_in_galley_p:`
`\__tag_check_mc_in_galley:TF`

At first we need a test to decide if `\tag_mc_begin:n` (tmb) and `\tag_mc_end:` (tme) has been used at all on the current galley. As each command issues two slightly different marks we can do it by comparing firstmarks and botmarks. The test assumes that the marks have been already mapped into the sequence with `\@@_mc_get_marks:`. As `\seq_if_eq:NNTF` doesn't exist we use the tl-test.

```
\prg_new_conditional:Npnn \__tag_check_if_mc_in_galley: { T,F,TF }
 {
   \tl_if_eq:NNTF \l__tag_mc_firstmarks_seq \l__tag_mc_botmarks_seq
    { \prg_return_false: }
    { \prg_return_true: }
 }
```

(*End definition for* `\__tag_check_mc_in_galley:TF`.)

`\__tag_check_if_mc_tmb_missing_p:`
`\__tag_check_if_mc_tmb_missing:TF`

This checks if a extra top mark ("extra-tmb") is needed. According to the analysis this the case if the firstmarks start with `e-` or `b+`. Like above we assume that the marks content is already in the seq's.

```
\prg_new_conditional:Npnn \__tag_check_if_mc_tmb_missing: { T,F,TF }
 {
  \bool_if:nTF
    {
      \str_if_eq_p:ee {\seq_item:Nn \l__tag_mc_firstmarks_seq {1}}{e-}
      ||
      \str_if_eq_p:ee {\seq_item:Nn \l__tag_mc_firstmarks_seq {1}}{b+}
    }
    { \prg_return_true: }
    { \prg_return_false: }
 }
```

(*End definition for* `\__tag_check_if_mc_tmb_missing:TF`.)

__tag_check_if_mc_tme_missing_p:
__tag_check_if_mc_tme_missing:TF

This checks if a extra bottom mark ("extra-tme") is needed. According to the analysis this the case if the botmarks starts with `b+`. Like above we assume that the marks content is already in the seq's.

```
\prg_new_conditional:Npnn \__tag_check_if_mc_tme_missing: { T,F,TF }
 {
   \str_if_eq:eeTF {\seq_item:Nn \l__tag_mc_botmarks_seq {1}}{b+}
    { \prg_return_true: }
    { \prg_return_false: }
 }
```

(*End definition for* `\__tag_check_if_mc_tme_missing:TF`.)

```
⟨/package⟩
⟨*debug⟩
```

Code for tagpdf-debug. This will probably change over time. At first something for the mc commands.

```
\msg_new:nnn { tag / debug } {mc-begin} { MC~begin~#1~with~options:~\tl_to_str:n{#2}~[\msg_li
\msg_new:nnn { tag / debug } {mc-end}   { MC~end~#1~[\msg_line_context:] }

\cs_new_protected:Npn \__tag_debug_mc_begin_insert:n #1
 {
   \int_compare:nNnT { \l__tag_loglevel_int } > {0}
     {
        \msg_note:nnnn { tag / debug } {mc-begin} {inserted} { #1 }
     }
 }
\cs_new_protected:Npn \__tag_debug_mc_begin_ignore:n #1
 {
   \int_compare:nNnT { \l__tag_loglevel_int } > {0}
     {
        \msg_note:nnnn { tag / debug } {mc-begin } {ignored} { #1 }
     }
 }
\cs_new_protected:Npn \__tag_debug_mc_end_insert:
 {
   \int_compare:nNnT { \l__tag_loglevel_int } > {0}
     {
        \msg_note:nnn { tag / debug } {mc-end} {inserted}
     }
 }
```

```
\cs_new_protected:Npn \__tag_debug_mc_end_ignore:
 {
   \int_compare:nNnT { \l__tag_loglevel_int } > {0}
     {
        \msg_note:nnn { tag / debug } {mc-end } {ignored}
     }
 }
```

And now something for the structures

```
\msg_new:nnn { tag / debug } {struct-begin}
  {
    Struct~begin~#1~with~options:~\tl_to_str:n{#2}~[\msg_line_context:]
  }
\msg_new:nnn { tag / debug } {struct-end}
  {
    Struct~end~#1~[\msg_line_context:]
  }

\cs_new_protected:Npn \__tag_debug_struct_begin_insert:n #1
 {
   \int_compare:nNnT { \l__tag_loglevel_int } > {0}
     {
        \msg_note:nnnn { tag / debug } {struct-begin} {inserted} { #1 }
        \seq_log:N \g__tag_struct_tag_stack_seq
     }
 }
\cs_new_protected:Npn \__tag_debug_struct_begin_ignore:n #1
 {
   \int_compare:nNnT { \l__tag_loglevel_int } > {0}
     {
        \msg_note:nnnn { tag / debug } {struct-begin } {ignored} { #1 }
     }
 }
\cs_new_protected:Npn \__tag_debug_struct_end_insert:
 {
   \int_compare:nNnT { \l__tag_loglevel_int } > {0}
     {
        \msg_note:nnn { tag / debug } {struct-end} {inserted}
        \seq_log:N \g__tag_struct_tag_stack_seq
     }
 }
\cs_new_protected:Npn \__tag_debug_struct_end_ignore:
 {
   \int_compare:nNnT { \l__tag_loglevel_int } > {0}
     {
        \msg_note:nnn { tag / debug } {struct-end } {ignored}
     }
 }
⟨/debug⟩
```

Part II

The **tagpdf-user** module
Code related to LaTeX2e user commands and document commands
Part of the tagpdf package

1 Setup commands

`\tagpdfsetup` `\tagpdfsetup{⟨key val list⟩}`

This is the main setup command to adapt the behaviour of tagpdf. It can be used in the preamble and in the document (but not all keys make sense there).

`activate␣(setup-key)` And additional setup key which combine the other activate keys `activate-mc`, `activate-tree`, `activate-struct` and additionally add a document structure.

`\tagpdfifluatexTF`
`\tagpdfifluatexT`
`\tagpdfifpdftexT`

small wrappers around engine tests. This functions should not be used and will be removed in one of the next versions.

2 Commands related to mc-chunks

`\tagmcbegin` `\tagmcbegin {⟨key-val⟩}`
`\tagmcend` `\tagmcend`
`\tagmcuse` `\tagmcuse{⟨label⟩}`

These are wrappers around `\tag_mc_begin:n`, `\tag_mc_end:` and `\tag_mc_use:n`. The commands and their argument are documentated in the tagpdf-mc module. In difference to the expl3 commands, `\tagmcbegin` issues also an `\ignorespaces`, and `\tagmcend` will issue in horizontal mode an `\unskip`.

`\tagmcifinTF` `\tagmcifin {⟨true code⟩}{⟨false code⟩}`

This is a wrapper around `\tag_mc_if_in:TF`. and tests if an mc is open or not. It is mostly of importance for pdflatex as lualatex doesn't mind much if a mc tag is not correctly closed. Unlike the expl3 command it is not expandable.

The command is probably not of much use and will perhaps disappear in future versions. It normally makes more sense to push/pop an mc-chunk.

3 Commands related to structures

`\tagstructbegin`
`\tagstructend`
`\tagstructuse`

```
\tagstructbegin {⟨key-val⟩}
\tagstructend
\tagstructuse{⟨label⟩}
```

These are direct wrappers around `\tag_struct_begin:n`, `\tag_struct_end:` and `\tag_struct_use:n`. The commands and their argument are documentated in the tagpdf-struct module.

4 Debugging

`\ShowTagging`

```
\ShowTagging {⟨key-val⟩}
```

This is a generic function to output various debugging helps. It not necessary stops the compilation. The keys and their function are described below.

`mc-data␣(show-key)`

```
mc-data = ⟨number⟩
```

This key is (currently?) relevant for lua mode only. It shows the data of all mc-chunks created so far. It is accurate only after shipout (and perhaps a second compilation), so typically should be issued after a newpage. The value is a positive integer and sets the first mc-shown. If no value is given, 1 is used and so all mc-chunks created so far are shown.

`mc-current␣(show-key)`

```
mc-current
```

This key shows the number and the tag of the currently open mc-chunk. If no chunk is open it shows only the state of the abs count. It works in all mode, but the output in luamode looks different.

`mc-marks␣(show-key)`

```
mc-marks = show|use
```

This key helps to debug the page marks. It should only be used at shipout in header or footer.

`struct-stack␣(show-key)`

```
struct-stack = log|show
```

This key shows the current structure stack. With `log` the info is only written to the log-file, `show` stops the compilation and shows on the terminal. If no value is used, then the default is `show`.

5 Extension commands

The following commands and code parts are not core command of tagpdf. They either provide work-arounds for missing functionality elsewhere, or do a first step to apply tagpdf commands to document commands.

The commands and keys should be view as experimental!

This part will be regularly revisited to check if the code should go to a better place or can be improved and so can change easily.

5.1 Fake space

`\pdffakespace` (lua-only) This provides a lua-version of the `\pdffakespace` primitive of pdftex.

5.2 Paratagging

This is a first try to make use of the new paragraph hooks in a current LaTeX to automate the tagging of paragraph. It requires sane paragraph nesting, faulty code, e.g. a missing `\par` at the end of a low-level vbox can highly confuse the tagging. The tags should be carefully checked if this is used.

```
paratagging␣(setup-key)       paratagging = true|false
paratagging-show␣(setup-key)  paratagging-show = true|false
```

This keys can be used in `\tagpdfsetup` and enable/disable paratagging. `paratagging-show` puts small red numbers at the begin and end of a paragraph. This is meant as a debugging help. The number are boxes and have a (tiny) height, so they can affect typesetting.

`\tagpdfparaOn`
`\tagpdfparaOff` These commands allow to enable/disable para tagging too and are a bit faster then `\tagpdfsetup`. But I'm not sure if the names are good.

`\tagpdfsuppressmarks` This command allows to suppress the creation of the marks. It takes an argument which should normally be one of the mc-commands, puts a group around it and suppress the marks creation in this group. This command should be used if the begin and end command are at different boxing levels. E.g.

```
\@hangfrom
 {
  \tagstructbegin{tag=H1}%
  \tagmcbegin    {tag=H1}%
  #2
 }
 {#3\tagpdfsuppressmarks{\tagmcend}\tagstructend}%
```

5.3 Header and footer

Header and footer are automatically excluded from tagging. This can be disabled with the following key. If some real content is in the header and footer, tagging must be restarted there explicitly. The key accepts the values `true` which surrounds the header with an artifact mc-chunk, `false` which disables the automatic tagging, and `pagination` which additionally adds an artifact structure with an pagination attribute.

```
exclude-header-footer␣(setup-key)  exclude-header-footer = true|false|pagination
```

5.4 Link tagging

Links need a special structure and cross reference system. This is added through hooks of the l3pdfannot module and will work automatically if tagging is activated.

Links should (probably) have an alternative text in the Contents key. It is unclear which text this should be and how to get it. Currently the code simply adds the fix texts `url` and `ref`. Another text can be added by changing the dictionary value:

```
\pdfannot_dict_put:nnn
{ link/GoTo }
{ Contents }
{ (ref) }
```

6 User commands and extensions of document commands

```
⟨@@=tag⟩
⟨*header⟩
\ProvidesExplPackage {tagpdf-user} {2022-01-09} {0.93}
  {tagpdf - user commands}
⟨/header⟩
```

7 Setup and preamble commands

\tagpdfsetup

```
⟨*package⟩
\NewDocumentCommand \tagpdfsetup { m }
  {
    \keys_set:nn { __tag / setup } { #1 }
  }
```

(*End definition for* `\tagpdfsetup`. *This function is documented on page 27.*)

8 Commands for the mc-chunks

\tagmcbegin
\tagmcend
\tagmcuse

```
\NewDocumentCommand \tagmcbegin { m }
  {
    \tag_mc_begin:n {#1}%\ignorespaces
  }


\NewDocumentCommand \tagmcend {  }
  {
    %\if_mode_horizontal: \unskip \fi: %
    \tag_mc_end:
  }

\NewDocumentCommand \tagmcuse { m }
  {
    \tag_mc_use:n {#1}
  }

```

(*End definition for* `\tagmcbegin`*,* `\tagmcend`*, and* `\tagmcuse`*. These functions are documented on page 27.*)

`\tagmcifinTF` This is a wrapper around `\tag_mc_if_in:` and tests if an mc is open or not. It is mostly of importance for pdflatex as lualatex doesn't mind much if a mc tag is not correctly closed. Unlike the expl3 command it is not expandable.

```
\NewDocumentCommand \tagmcifinTF { m m }
  {
    \tag_mc_if_in:TF { #1 } { #2 }
  }
```

(*End definition for* `\tagmcifinTF`*. This function is documented on page 27.*)

9 Commands for the structure

`\tagstructbegin` `\tagstructend` `\tagstructuse` These are structure related user commands. There are direct wrapper around the expl3 variants.

```
\NewDocumentCommand \tagstructbegin { m }
  {
    \tag_struct_begin:n {#1}
  }

\NewDocumentCommand \tagstructend {  }
  {
   \tag_struct_end:
  }

\NewDocumentCommand \tagstructuse { m }
  {
    \tag_struct_use:n {#1}
  }
```

(*End definition for* `\tagstructbegin`*,* `\tagstructend`*, and* `\tagstructuse`*. These functions are documented on page 28.*)

`\tagpdfifluatexTF` `\tagpdfifluatexT` `\tagpdfifpdftexT` I should deprecate them ...

```
\cs_set_eq:NN\tagpdfifluatexTF \sys_if_engine_luatex:TF
\cs_set_eq:NN\tagpdfifluatexT  \sys_if_engine_luatex:T
\cs_set_eq:NN\tagpdfifpdftexT  \sys_if_engine_pdftex:T
```

(*End definition for* `\tagpdfifluatexTF`*,* `\tagpdfifluatexT`*, and* `\tagpdfifpdftexT`*. These functions are documented on page 27.*)

10 Debugging

`\ShowTagging` This is a generic command for various show commands. It takes a keyval list, the various keys are implemented below.

```
\NewDocumentCommand\ShowTagging { m }
  {
    \keys_set:nn { __tag / show }{ #1}

}
```

(*End definition for* `\ShowTagging`*. This function is documented on page 28.*)

`mc-data␣(show-key)` This key is (currently?) relevant for lua mode only. It shows the data of all mc-chunks created so far. It is accurate only after shipout, so typically should be issued after a newpage. With the optional argument the minimal number can be set.

```
\keys_define:nn { __tag / show }
  {
    mc-data .code:n =
      {
        \sys_if_engine_luatex:T
          {
            \lua_now:e{ltx.__tag.trace.show_all_mc_data(#1,\__tag_get_mc_abs_cnt:,0)}
          }
      }
    ,mc-data .default:n = 1
  }

```

(*End definition for* `mc-data (show-key)`*. This function is documented on page 28.*)

`mc-current␣(show-key)` This shows some info about the current mc-chunk. It works in generic and lua-mode.

```
\keys_define:nn { __tag / show }
  { mc-current .code:n =
     {
       \bool_if:NTF \g__tag_mode_lua_bool
         {
           \sys_if_engine_luatex:T
             {
               \int_compare:nNnTF
                 { -2147483647 }
                  =
                 {
                   \lua_now:e
                     {
                        tex.print
                         (tex.getattribute
                           (luatexbase.attributes.g__tag_mc_cnt_attr))
                     }
                 }
                 {
                   \lua_now:e
                     {
                       ltx.__tag.trace.log
                        (
                          "mc-current:~no~MC~open,~current~abscnt
                           =\__tag_get_mc_abs_cnt:"
                          ,0
                        )
                       texio.write_nl("")
                     }
                 }
                 {
                   \lua_now:e
                     {
                       ltx.__tag.trace.log
                        (
                          "mc-current:~abscnt=\__tag_get_mc_abs_cnt:=="
                           ..
                           tex.getattribute(luatexbase.attributes.g__tag_mc_cnt_attr)
                           ..
                           "~=>tag="
                           ..
                           tostring
                             (ltx.__tag.func.get_tag_from
                               (tex.getattribute
                                 (luatexbase.attributes.g__tag_mc_type_attr)))
                           ..
                           "="
                           ..
                           tex.getattribute
                            (luatexbase.attributes.g__tag_mc_type_attr)
                           ,0
                        )
                       texio.write_nl("")
                     }
                 }
             }
         }
         {
          \msg_note:nn{ tag }{ mc-current }
         }
     }
 }
```

(*End definition for* `mc-current (show-key)`. *This function is documented on page 28.*)

`mc-marks␣(show-key)` It maps the mc-marks into the sequences and then shows them. This allows to inspect the first and last mc-Mark on a page. It should only be used in the shipout (header/footer).

```
\keys_define:nn { __tag / show }
  {
    mc-marks .choice: ,
    mc-marks / show .code:n =
      {
        \__tag_mc_get_marks:
        \__tag_check_if_mc_in_galley:TF
         {
          \iow_term:n {Marks~from~this~page:~}
         }
         {
           \iow_term:n {Marks~from~a~previous~page:~}
         }
        \seq_show:N \l__tag_mc_firstmarks_seq
        \seq_show:N \l__tag_mc_botmarks_seq
        \__tag_check_if_mc_tmb_missing:T
         {
          \iow_term:n {BDC~missing~on~this~page!}
        }
       \__tag_check_if_mc_tme_missing:T
        {
          \iow_term:n {EMC~missing~on~this~page!}
        }
     },
   mc-marks / use .code:n =
     {
       \__tag_mc_get_marks:
       \__tag_check_if_mc_in_galley:TF
        { Marks~from~this~page:~}
        { Marks~from~a~previous~page:~}
       \seq_use:Nn \l__tag_mc_firstmarks_seq {,~}\quad
       \seq_use:Nn \l__tag_mc_botmarks_seq {,~}\quad
       \__tag_check_if_mc_tmb_missing:T
        {
          BDC~missing~
        }
       \__tag_check_if_mc_tme_missing:T
        {
          EMC~missing
        }
     },
  mc-marks .default:n = show
 }
```

(*End definition for* `mc-marks (show-key)`*. This function is documented on page 28.*)

`struct-stack␣(show-key)`

```
\keys_define:nn { __tag / show }
  {
     struct-stack .choice:
    ,struct-stack / log .code:n = \seq_log:N \g__tag_struct_tag_stack_seq
    ,struct-stack / show .code:n = \seq_show:N \g__tag_struct_tag_stack_seq
    ,struct-stack .default:n = show
  }
```

(*End definition for* `struct-stack (show-key)`*. This function is documented on page 28.*)

11 Commands to extend document commands

The following commands and code parts are not core command of tagpdf. The either provide work arounds for missing functionality elsewhere, or do a first step to apply tagpdf commands to document commands. This part should be regularly revisited to check if the code should go to a better place or can be improved.

11.1 Document structure

`\__tag_add_document_structure:n`
`activate␣(setup-key)`

```
\cs_new_protected:Npn \__tag_add_document_structure:n #1
 {
   \hook_gput_code:nnn{begindocument}{tagpdf}{\tagstructbegin{tag=#1}}
   \hook_gput_code:nnn{tagpdf/finish/before}{tagpdf}{\tagstructend}
 }
\keys_define:nn { __tag / setup}
 {
   activate   .code:n =
    {
      \keys_set:nn { __tag / setup }
        { activate-mc,activate-tree,activate-struct }
      \__tag_add_document_structure:n {#1}
    },
  activate .default:n = Document
 }
```

(*End definition for* `\__tag_add_document_structure:n` *and* `activate (setup-key)`*. This function is documented on page 27.*)

11.2 Structure destinations

In TeXlive 2022 pdftex and luatex will offer support for structure destinations. The pdfmanagement has already backend support. We activate them if the prerequisites are there: The pdf version should be 2.0, structures should be activated, the code in the pdfmanagement must be there.

```
\AddToHook{begindocument/before}
  {
    \bool_lazy_all:nT
      {
        { \g__tag_active_struct_dest_bool }
        { \g__tag_active_struct_bool }
        { \cs_if_exist_p:N \pdf_activate_structure_destination: }
        { ! \pdf_version_compare_p:Nn < {2.0} }
      }
      {
        \tl_set:Nn \l_pdf_current_structure_destination_tl { __tag/struct/\g__tag_struct_stac
        \pdf_activate_structure_destination:
      }
   }
```

11.3 Fake space

`\pdffakespace` We need a luatex variant for `\pdffakespace`. This should probably go into the kernel at some time.

```
\sys_if_engine_luatex:T
  {
    \NewDocumentCommand\pdffakespace { }
     {
       \__tag_fakespace:
     }
  }
```

(*End definition for* `\pdffakespace`*. This function is documented on page 29.*)

11.4 Paratagging

The following are some simple commands to enable/disable paratagging. Probably one should add some checks if we are already in a paragraph.

\l__tag_para_bool
\l__tag_para_show_bool
\g__tag_para_int

At first some variables.

```
\bool_new:N \l__tag_para_bool
\bool_new:N \l__tag_para_show_bool
\int_new:N  \g__tag_para_begin_int
\int_new:N  \g__tag_para_end_int
```

(*End definition for* `\l__tag_para_bool`*,* `\l__tag_para_show_bool`*, and* `\g__tag_para_int`*.*)

paratagging␣(setup-key)
paratagging-show␣(setup-key)

These keys enable/disable locally paratagging, and the debug modus. It can affect the typesetting if `paratagging-show` is used. The small numbers are boxes and they have a (small) height.

```
\keys_define:nn { __tag / setup }
  {
    paratagging      .bool_set:N = \l__tag_para_bool,
    paratagging-show .bool_set:N = \l__tag_para_show_bool,
  }

```

(*End definition for* `paratagging (setup-key)` *and* `paratagging-show (setup-key)`*. These functions are documented on page 29.*)

This fills the para hooks with the needed code.

```
\AddToHook{para/begin}
  {
   \bool_if:NT \l__tag_para_bool
     {
       \int_gincr:N \g__tag_para_begin_int
       \tag_struct_begin:n {tag=P}
       \bool_if:NT \l__tag_para_show_bool
        { \tag_mc_begin:n{artifact}
          \llap{\color_select:n{red}\tiny\int_use:N\g__tag_para_begin_int\ }
          \tag_mc_end:
        }
       \tag_mc_begin:n {tag=P}
     }
  }
\AddToHook{para/end}
  {
    \bool_if:NT \l__tag_para_bool
      {
        \int_gincr:N \g__tag_para_end_int
        \tag_mc_end:
        \bool_if:NT \l__tag_para_show_bool
          { \tag_mc_begin:n{artifact}
            \rlap{\color_select:n{red}\tiny\ \int_use:N\g__tag_para_end_int}
            \tag_mc_end:
          }
        \tag_struct_end:
      }
  }
```

```
\AddToHook{enddocument/info}
  {
    \int_compare:nNnF {\g__tag_para_begin_int}={\g__tag_para_end_int}
      {
        \msg_error:nnxx
          {tag}
          {para-hook-count-wrong}
          {\int_use:N\g__tag_para_begin_int}
          {\int_use:N\g__tag_para_end_int}
      }
  }
```

In generic mode we need the additional code from the ptagging tests.

```
\AddToHook{begindocument/before}
 {
   \bool_if:NF \g__tag_mode_lua_bool
     {
        \cs_if_exist:NT \@kernel@before@footins
         {
           \tl_put_right:Nn \@kernel@before@footins
             { \__tag_add_missing_mcs_to_stream:Nn \footins {footnote} }
           \tl_put_right:Nn \@kernel@before@cclv
             {
               \__tag_check_typeout_v:n {====>~In~\token_to_str:N \@makecol\c_space_tl\the\c@
               \__tag_add_missing_mcs_to_stream:Nn \@cclv {main}
             }
           \tl_put_right:Nn \@mult@ptagging@hook
             {
               \__tag_check_typeout_v:n {====>~In~\string\page@sofar}
               \process@cols\mult@firstbox
                {
                  \__tag_add_missing_mcs_to_stream:Nn \count@ {multicol}
                }
               \__tag_add_missing_mcs_to_stream:Nn \mult@rightbox {multicol}
             }
         }
     }
  }
```

`\tagpdfparaOn` `\tagpdfparaOff` This two command switch para mode on and off. `\tagpdfsetup` could be used too but is longer.

```
\newcommand\tagpdfparaOn {\bool_set_true:N \l__tag_para_bool}
\newcommand\tagpdfparaOff{\bool_set_false:N \l__tag_para_bool}
```

(*End definition for* `\tagpdfparaOn` *and* `\tagpdfparaOff`*. These functions are documented on page 29.*)

`\tagpdfsuppressmarks` This command allows to suppress the creation of the marks. It takes an argument which should normally be one of the mc-commands, puts a group around it and suppress the marks creation in this group. This command should be used if the begin and end command are at different boxing levels. E.g.

```
\@hangfrom
 {
  \tagstructbegin{tag=H1}%
  \tagmcbegin    {tag=H1}%
  #2
 }
 {#3\tagpdfsuppressmarks{\tagmcend}\tagstructend}%
```

```
\NewDocumentCommand\tagpdfsuppressmarks{m}
  {{\use:c{__tag_mc_disable_marks:} #1}}
```

(*End definition for* `\tagpdfsuppressmarks`. *This function is documented on page 29.*)

11.5 Header and footer

Header and footer should normally be tagged as artifacts. The following code requires the new hooks. For now we allow to disable this function, but probably the code should always there at the end. TODO check if Pagination should be changeable.

```
\cs_new_protected:Npn\__tag_hook_kernel_before_head:{}
\cs_new_protected:Npn\__tag_hook_kernel_after_head:{}
\cs_new_protected:Npn\__tag_hook_kernel_before_foot:{}
\cs_new_protected:Npn\__tag_hook_kernel_after_foot:{}

\AddToHook{begindocument}
 {
  \cs_if_exist:NT \@kernel@before@head
   {
     \tl_put_right:Nn \@kernel@before@head {\__tag_hook_kernel_before_head:}
     \tl_put_left:Nn  \@kernel@after@head  {\__tag_hook_kernel_after_head:}
     \tl_put_right:Nn \@kernel@before@foot {\__tag_hook_kernel_before_foot:}
     \tl_put_left:Nn  \@kernel@after@foot  {\__tag_hook_kernel_after_foot:}
   }
 }

\bool_new:N \g__tag_saved_in_mc_bool
\cs_new_protected:Npn \__tag_exclude_headfoot_begin:
 {
    \bool_set_false:N  \l__tag_para_bool
    \bool_if:NTF \g__tag_mode_lua_bool
     {
      \tag_mc_end_push:
     }
     {
       \bool_gset_eq:NN   \g__tag_saved_in_mc_bool \g__tag_in_mc_bool
       \bool_gset_false:N \g__tag_in_mc_bool
     }
    \tag_mc_begin:n {artifact}
 }
\cs_new_protected:Npn \__tag_exclude_headfoot_end:
 {
    \tag_mc_end:
    \bool_if:NTF \g__tag_mode_lua_bool
     {
      \tag_mc_begin_pop:n{}
     }
     {
       \bool_gset_eq:NN \g__tag_in_mc_bool\g__tag_saved_in_mc_bool
    }
}
```

This version allows to use an Artifact structure

```
\__tag_attr_new_entry:nn {__tag/attr/pagination}{/O/Artifact/Type/Pagination}
\cs_new_protected:Npn \__tag_exclude_struct_headfoot_begin:n #1
 {
   \bool_set_false:N  \l__tag_para_bool
   \bool_if:NTF \g__tag_mode_lua_bool
    {
     \tag_mc_end_push:
    }
    {
      \bool_gset_eq:NN   \g__tag_saved_in_mc_bool \g__tag_in_mc_bool
      \bool_gset_false:N \g__tag_in_mc_bool
    }
   \tag_struct_begin:n{tag=Artifact,attribute-class=__tag/attr/#1}
   \tag_mc_begin:n {artifact=#1}
 }

\cs_new_protected:Npn \__tag_exclude_struct_headfoot_end:
 {
   \tag_mc_end:
   \tag_struct_end:
   \bool_if:NTF \g__tag_mode_lua_bool
    {
     \tag_mc_begin_pop:n{}
    }
    {
      \bool_gset_eq:NN \g__tag_in_mc_bool\g__tag_saved_in_mc_bool
    }
 }
```

And now the keys

exclude-header-footer␣(setup-key)

```
\keys_define:nn { __tag / setup }
  {
   exclude-header-footer .choice:,
   exclude-header-footer / true .code:n =
    {
      \cs_set_eq:NN \__tag_hook_kernel_before_head: \__tag_exclude_headfoot_begin:
      \cs_set_eq:NN \__tag_hook_kernel_before_foot: \__tag_exclude_headfoot_begin:
      \cs_set_eq:NN \__tag_hook_kernel_after_head:  \__tag_exclude_headfoot_end:
      \cs_set_eq:NN \__tag_hook_kernel_after_foot:  \__tag_exclude_headfoot_end:
    },
   exclude-header-footer / pagination .code:n =
    {
      \cs_set:Nn \__tag_hook_kernel_before_head: { \__tag_exclude_struct_headfoot_begin:n {p
      \cs_set:Nn \__tag_hook_kernel_before_foot: { \__tag_exclude_struct_headfoot_begin:n {p
      \cs_set_eq:NN \__tag_hook_kernel_after_head:  \__tag_exclude_struct_headfoot_end:
      \cs_set_eq:NN \__tag_hook_kernel_after_foot:  \__tag_exclude_struct_headfoot_end:
    },
   exclude-header-footer / false .code:n =
    {
      \cs_set_eq:NN \__tag_hook_kernel_before_head: \prg_do_nothing:
      \cs_set_eq:NN \__tag_hook_kernel_before_foot: \prg_do_nothing:
      \cs_set_eq:NN \__tag_hook_kernel_after_head:  \prg_do_nothing:
      \cs_set_eq:NN \__tag_hook_kernel_after_foot:  \prg_do_nothing:
    },
   exclude-header-footer .default:n = true,
   exclude-header-footer .initial:n = true
  }
```

(*End definition for* `exclude-header-footer (setup-key)`. *This function is documented on page 29.*)

11.6 Links

We need to close and reopen mc-chunks around links. Currently we handle URI and GoTo (internal) links. Links should have an alternative text in the Contents key. It is unclear which text this should be and how to get it.

```
\hook_gput_code:nnn
  {pdfannot/link/URI/before}
  {tagpdf}
  {
    \tag_mc_end_push:
    \tag_struct_begin:n { tag=Link }
    \tag_mc_begin:n { tag=Link }
    \pdfannot_dict_put:nnx
      { link/URI }
      { StructParent }
      { \tag_struct_parent_int: }
  }

\hook_gput_code:nnn
  {pdfannot/link/URI/after}
  {tagpdf}
  {
     \tag_struct_insert_annot:xx {\pdfannot_link_ref_last:}{\tag_struct_parent_int:}
     \tag_mc_end:
     \tag_struct_end:
     \tag_mc_begin_pop:n{}
  }

\hook_gput_code:nnn
  {pdfannot/link/GoTo/before}
  {tagpdf}
  {
     \tag_mc_end_push:
     \tag_struct_begin:n{tag=Link}
     \tag_mc_begin:n{tag=Link}
     \pdfannot_dict_put:nnx
       { link/GoTo }
       { StructParent }
       { \tag_struct_parent_int: }
  }

\hook_gput_code:nnn
  {pdfannot/link/GoTo/after}
  {tagpdf}
  {
    \tag_struct_insert_annot:xx {\pdfannot_link_ref_last:}{\tag_struct_parent_int:}
    \tag_mc_end:
    \tag_struct_end:
    \tag_mc_begin_pop:n{}

  }

% "alternative descriptions " for PAX3. How to get better text here??
\pdfannot_dict_put:nnn
 { link/URI }
 { Contents }
 { (url) }

\pdfannot_dict_put:nnn
 { link/GoTo }
 { Contents }
 { (ref) }

```

</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.

Symbols

\\\\ . . . 10
\␣ . . . 232, 246

A

activate␣(setup-key) . . . *27*, 178
activate-all␣(setup-key) . . . *4*, 167
activate-mc␣(setup-key) . . . *4*, 167
activate-space␣(setup-key) . . . *4*, 167
activate-struct␣(setup-key) . . . *4*, 167
activate-tree␣(setup-key) . . . *4*, 167
\AddToHook 63, 193, 224, 238, 252, 263, 297
attr-unknown . . . *15*, 33

B

\begin . . . 32
bool commands:
  \bool_gset_eq:NN . . . 317, 330, 342, 358
  \bool_gset_false:N . . . 43, 318, 343
  \bool_gset_true:N . . . 42, 99
  \bool_if:NTF . 18, 69, 208, 217, 226, 230, 240, 244, 265, 312, 325, 337, 353
  \bool_if:nTF . . . 6, 258
  \bool_lazy_all:nTF . . . 50, 195
  \bool_lazy_and:nnTF . . . 67, 77
  \bool_lazy_and_p:nn . . . 8
  \bool_new:N . . . 41, 94, 95, 96, 97, 98, 100, 102, 104, 214, 215, 308
  \bool_set_false:N . . . 163, 164, 289, 311, 336
  \bool_set_true:N . . . 101, 103, 288
box commands:
  \box_new:N . . . 89, 90

C

c@g internal commands:
  \c@g__tag_MCID_abs_int . . . 57, 126
  \c@g__tag_struct_abs_int 115, 118, 120
clist commands:
  \clist_const:Nn . . . 91, 92
  \clist_new:N . . . 87
color commands:
  \color_select:n . . . 232, 246
cs commands:
  \cs_generate_variant:Nn . 106, 107, 108, 108, 109, 110, 111, 112, 113, 135, 140, 154, 155, 156, 157, 158, 159
  \cs_gset_eq:NN . . . 183
  \cs_if_exist:NTF . . . 65, 267, 299
  \cs_if_exist_p:N . . . 9, 199
  \cs_new:Npn . . . 47, 67, 73, 136, 141
  \cs_new_protected:Npn . . . 47, 85, 93, 100, 109, 113, 123, 129, 142, 150, 157, 160, 169, 178, 180, 185, 204, 278, 285, 292, 292, 293, 294, 295, 299, 309, 315, 322, 323, 330, 334, 338, 349
  \cs_set:Nn . . . 373, 374
  \cs_set_eq:NN . . . 43, 46, 47, 48, 145, 146, 147, 148, 149, 150, 151, 152, 166, 366, 367, 368, 369, 375, 376, 380, 381, 382, 383
  \cs_set_protected:Nn . . . 183

D

\DeclareDocumentMetadata . . . 21
\DeclareOption . . . 42, 43
\documentclass . . . 22

E

\end . . . 29
\endinput . . . 28
\enquote . . . 31
exclude-header-footer␣(setup-key) . . . *29*, 361
\ExecuteOptions . . . 44
exp commands:
  \exp_args:Nee . . . 69
  \exp_args:NNnx . . . 51
  \exp_args:NNx . . . 51

F

fi commands:
  \fi: . . . 19
file commands:
  \file_input:n . . . 204
\footins . . . 270

G

group commands:
  \group_begin: . . . 162
  \group_end: . . . 166

H

hook commands:
  \hook_gput_code:nnn . . . 180, 181, 388, 401, 411, 424

I

if commands:
 \if_mode_horizontal: 19
\ignorespaces *27*, 13
int commands:
 \int_compare:nNnTF 73, 102, 132, 159, 162, 187, 193, 254, 280, 287, 294, 301, 317, 325, 332, 340
 \int_compare:nTF 227
 \int_gincr:N 228, 242
 \int_new:N 88, 93, 216, 217
 \int_set:Nn . . . 179, 182, 185, 186, 187
 \int_step_inline:nnnn 212, 218
 \int_use:N 56, 57, 115, 118, 120, 124, 126, 128, 232, 246, 259, 260
intarray commands:
 \intarray_gset:Nnn 190
 \intarray_item:Nn 192, 195
 \intarray_new:Nn 182
iow commands:
 \iow_now:Nn 51
 \iow_term:n . . . 136, 139, 145, 149, 183

K

keys commands:
 \keys_define:nn 54, 66, 128, 167, 171, 183, 218, 361
 \keys_set:nn 9, 51, 187

L

legacy commands:
 \legacy_if:nTF 49
\llap 232
log␣(setup-key) . . . . *4*, <u>178</u>
lua commands:
 \lua_now:n 60, 77, 85, 97

M

mc-current . . . . *15*, <u>16</u>
mc-current␣(show-key) . . . . *28*, <u>66</u>
mc-data␣(show-key) . . . . *28*, <u>54</u>
mc-label-unknown . . . . *15*, <u>9</u>
mc-marks␣(show-key) . . . . *28*, <u>128</u>
mc-nested . . . . *15*, <u>6</u>
mc-not-open . . . . *15*, <u>13</u>
mc-popped . . . . *15*, <u>14</u>
mc-pushed . . . . *15*, <u>14</u>
mc-tag-missing . . . . *15*, <u>8</u>
mc-used-twice . . . . *15*, <u>12</u>
\MessageBreak 15, 19, 20, 21
msg commands:
 \msg_error:nn 90, 111
 \msg_error:nnn 173
 \msg_error:nnnn 256
 \msg_info:nnn 104, 161, 165
 \msg_info:nnnn 134
 \msg_line_context: . 275, 276, 308, 312
 \g_msg_module_name_prop 34, 38
 \g_msg_module_type_prop 37
 \msg_new:nnn . . . 7, 8, 9, 12, 13, 14, 15, 16, 22, 23, 26, 27, 29, 31, 33, 34, 35, 36, 37, 38, 39, 41, 275, 276, 306, 310
 \msg_new:nnnn 44
 \msg_note:nn 124
 \msg_note:nnn 296, 303, 334, 342
 \msg_note:nnnn 282, 289, 319, 327
 \msg_warning:nnn 97, 120, 127, 138, 146, 154, 177, 200

N

new-tag . . . . *15*, <u>37</u>
\newcommand 288, 289
\NewDocumentCommand 7, 11, 17, 23, 28, 32, 37, 42, 49, 209, 290
\newlabeldata 53
no-struct-dest␣(setup-key) . . . . . *4*, <u>167</u>

P

\PackageError 13
\PackageWarning 28
para-hook-count-wrong . . . . *16*, <u>44</u>
paratagging␣(setup-key) . . . . *29*, <u>218</u>
paratagging-show␣(setup-key) . . . *29*, <u>218</u>
pdf commands:
 \pdf_activate_structure_destination: 199, 204
 \l_pdf_current_structure_-destination_tl 203
 \pdf_object_if_exist:n 105
 \pdf_object_if_exist:nTF 118
 \pdf_object_ref:n 106, 120
 \pdf_uncompress: 200
 \pdf_version_compare_p:Nn 200
pdfannot commands:
 \pdfannot_dict_put:nnn 107, 395, 418, 436, 441
 \pdfannot_link_ref_last: . . . 405, 428
\pdffakespace . . . . *29*, <u>207</u>
pdffile commands:
 \pdffile_embed_stream:nnn 108
pdfmanagement commands:
 \pdfmanagement_add:nnn . 192, 194, 196
 \pdfmanagement_if_active_p: . . 9, 10
 \pdfmanagement_remove:nn 198
prg commands:
 \prg_do_nothing: 183, 380, 381, 382, 383
 \prg_generate_conditional_-variant:Nnn 105

\prg_new_conditional:Npnn 48, 65, 75, 250, 256, 267
\prg_return_false: 62, 72, 82, 253, 265, 271
\prg_return_true: 59, 69, 79, 254, 264, 270
\ProcessOptions 45
prop commands:
\prop_get:NnNTF 115, 130
\prop_gput:Nnn . . . 34, 37, 38, 109, 147
\prop_if_in:NnTF 87, 95, 175
\prop_item:Nn 32, 150
\prop_new:N 84, 145
\prop_put:Nnn 110
\prop_show:N 152
\ProvidesExplPackage 3, 3, 3, 26

Q

\quad 158, 159

R

ref commands:
\ref_attribute_gset:nnnn 114, 116, 123, 125, 127
\ref_label:nn 111, 132
\ref_value:nnn . *4*, <u>65</u>, 65, 67, 138, 143
ref internal commands:
__ref_value:nnn 70, 73
\RequirePackage 20, 46, 210, 213, 219, 222
\rlap 246
role-missing . . . . *15*, <u>34</u>
role-tag . . . . *15*, <u>37</u>
role-unknown . . . . *15*, <u>34</u>
role-unknown-tag . . . . *15*, <u>34</u>

S

seq commands:
\seq_clear:N 217
\seq_gput_right:Nn 148, 237
\seq_item:Nn 149, 260, 262, 269
\seq_log:N 166, 174, 320, 335
\seq_new:N 85, 86, 146
\seq_set_split:Nnn 112
\seq_show:N . . . 141, 142, 151, 175, 247
\seq_use:Nn 158, 159
\l_tmpa_seq 217, 237, 247
shipout commands:
\g_shipout_readonly_int 56, 124
\ShowTagging . . . . *13*, *28*, <u>49</u>
str commands:
\str_if_eq:nnTF 269
\str_if_eq_p:nn 260, 262
\str_new:N 83
\str_set_convert:Nnnn 113
\string 20, 21, 22, 278
struct-faulty-nesting . . . . *15*, <u>23</u>
struct-label-unknown . . . . *15*, <u>29</u>
struct-missing-tag . . . . *15*, <u>26</u>
struct-no-objnum . . . . *15*, <u>22</u>
struct-show-closing . . . . *15*, <u>31</u>
struct-stack␣(show-key) . . . . *28*, <u>171</u>
struct-used-twice . . . . *15*, <u>27</u>
sys commands:
\sys_if_engine_luatex:TF 42, 46, 47, 58, 71, 202, 207
\sys_if_engine_pdftex:TF 48
sys-no-interwordspace . . . . *16*, <u>41</u>

T

tabsorder␣(setup-key) . . . . *4*, <u>190</u>
tag commands:
\tag_get:n . . . . *13*, <u>47</u>, 47
\tag_if_active: 48
\tag_if_active:TF . . . . *13*, <u>48</u>
\tag_if_active_p: . . . . *13*, <u>48</u>
\tag_mc_begin:n *7*, 13, 231, 235, 245, 320, 346, 394, 417
\tag_mc_begin_pop:n 327, 355, 408, 431
\tag_mc_end: 20, 233, 243, 247, 324, 351, 406, 429
\tag_mc_end_push: . . 314, 339, 392, 415
\tag_mc_if_in:TF 30
\tag_mc_use:n 25
\tag_stop_group_begin: . . . *4*, <u>160</u>, 160
\tag_stop_group_end: . . . . *4*, <u>160</u>, 166
\tag_struct_begin:n 34, 229, 345, 393, 416
\tag_struct_end: 39, 249, 352, 407, 430
\tag_struct_insert_annot:nn 405, 428
\tag_struct_parent_int: 398, 405, 421, 428
\tag_struct_use:n 44
tag internal commands:
\g__tag_active_mc_bool 53, 67, <u>94</u>, 170
\l__tag_active_mc_bool 56, 67, <u>100</u>, 164
\g__tag_active_space_bool . . . <u>94</u>, 169
\g__tag_active_struct_bool . . . . 52, 77, <u>94</u>, 172, 198
\l__tag_active_struct_bool . . . . 55, 77, <u>100</u>, 163
\g__tag_active_struct_dest_bool . . . . <u>94</u>, 176, 197
\g__tag_active_tree_bool . 54, <u>94</u>, 171
__tag_add_document_structure:n . . . . <u>178</u>, 178, 189
__tag_add_missing_mcs_to_-
stream:Nn 270, 274, 281, 283
__tag_attr_new_entry:nn 333

__tag_check_add_tag_role:nn 123, 123
__tag_check_if_active_mc: 65
__tag_check_if_active_mc:TF . . . 65
__tag_check_if_active_struct: . . 75
__tag_check_if_active_struct:TF 65
__tag_check_if_mc_in_galley: . . 250
__tag_check_if_mc_in_galley:TF 134, 155
__tag_check_if_mc_tmb_missing: 256
__tag_check_if_mc_tmb_missing:TF 143, 160, 256
__tag_check_if_mc_tmb_missing_-p: 256
__tag_check_if_mc_tme_missing: 267
__tag_check_if_mc_tme_missing:TF 147, 164, 267
__tag_check_if_mc_tme_missing_-p: 267
__tag_check_info_closing_-struct:n 100, 100, 108
__tag_check_init_mc_used: 180, 180, 183, 189
__tag_check_mc_if_nested: . 142, 142
__tag_check_mc_if_open: . . . 142, 150
__tag_check_mc_in_galley:TF . . 250
__tag_check_mc_in_galley_p: . . 250
__tag_check_mc_pushed_popped:nn 157, 157
__tag_check_mc_tag:N 169, 169
__tag_check_mc_used:n 185, 185
\g__tag_check_mc_used_intarray 180, 190, 192, 195
__tag_check_no_open_struct: 109, 109
__tag_check_show_MCID_by_page: 204, 204
__tag_check_struct_used:n . 113, 113
__tag_check_structure_has_tag:n 85, 85
__tag_check_structure_tag:N 93, 93
__tag_check_typeout_v:n 43, 43, 183, 273, 278
__tag_debug_mc_begin_ignore:n . 285
__tag_debug_mc_begin_insert:n . 278
__tag_debug_mc_end_ignore: . . . 299
__tag_debug_mc_end_insert: . . . 292
__tag_debug_struct_begin_-ignore:n 323
__tag_debug_struct_begin_-insert:n 315
__tag_debug_struct_end_ignore: 338
__tag_debug_struct_end_insert: 330
__tag_exclude_headfoot_begin: 309, 366, 367
__tag_exclude_headfoot_end: 322, 368, 369
__tag_exclude_struct_headfoot_-begin:n 334, 373, 374
__tag_exclude_struct_headfoot_-end: 349, 375, 376
__tag_fakespace: 211
__tag_get_mc_abs_cnt: 19, 20, 60, 90, 101, 146, 154, 173
__tag_hook_kernel_after_foot: 295, 304, 369, 376, 383
__tag_hook_kernel_after_head: 293, 302, 368, 375, 382
__tag_hook_kernel_before_foot: 294, 303, 367, 374, 381
__tag_hook_kernel_before_head: 292, 301, 366, 373, 380
\g__tag_in_mc_bool 18, 317, 318, 330, 342, 343, 358
__tag_lastpagelabel: 47, 47, 64
\l__tag_loglevel_int 93, 102, 132, 160, 163, 179, 182, 185, 186, 187, 187, 280, 287, 294, 301, 317, 325, 332, 340
\l__tag_mc_botmarks_seq 142, 159, 252, 269
\l__tag_mc_firstmarks_seq 141, 158, 252, 260, 262
__tag_mc_get_marks: 133, 154
__tag_mc_if_in:TF 144, 152
\g__tag_mc_key_tag_tl 19
\g__tag_mc_stack_seq 166
\g__tag_MCID_tmp_bypage_int . . . 128
\g__tag_mode_lua_bool . . . 41, 42, 43, 69, 208, 217, 265, 312, 325, 337, 353
\g__tag_para_begin_int 216, 228, 232, 254, 259
\l__tag_para_bool 214, 220, 226, 240, 288, 289, 311, 336
\g__tag_para_end_int 217, 242, 246, 254, 260
\g__tag_para_int 214
\l__tag_para_show_bool 214, 221, 230, 244
__tag_prop_gput:Nnn . . 145, 147, 154
__tag_prop_item:Nn 145, 150
__tag_prop_new:N 145, 145, 156
__tag_prop_show:N 145, 152, 159
__tag_ref_label:nn . . . 129, 129, 135
__tag_ref_value:nnn 116, 136, 136, 140, 229, 240
__tag_ref_value_lastpage:nn 141, 141, 208, 222
\c__tag_refmc_clist 91

\c__tag_refstruct_clist 91
\g__tag_role_tags_NS_prop 175
\g__tag_role_tags_prop 95, 130
\g__tag_saved_in_mc_bool 308, 317, 330, 342, 358
__tag_seq_gput_right:Nn 145, 148, 155
__tag_seq_item:Nn 145, 149
__tag_seq_new:N 145, 146, 157
__tag_seq_show:N 145, 151, 158
\g__tag_struct_stack_current_tl 203
\g__tag_struct_tag_stack_seq 174, 175, 320, 335
\g__tag_tagunmarked_bool . . . 104, 188
\l__tag_tmpa_box 82
\l__tag_tmpa_clist 82
\l__tag_tmpa_int 82
\l__tag_tmpa_prop 82
\l__tag_tmpa_seq 82
\l__tag_tmpa_str 82
\l__tag_tmpa_tl 82, 206, 214
\l__tag_tmpb_box 82
\l__tag_tmpb_seq 82
tagabspage *4*, 114
tagmcabs *4*, 114
\tagmcbegin *27*, 11
\tagmcend *27*, 11
tagmcid *4*, 114
\tagmcifin *27*
\tagmcifinTF *27*, 28
\tagmcuse *27*, 11
\tagpdfifluatexT *27*, 46
\tagpdfifluatexTF *27*, 46
\tagpdfifpdftexT *27*, 46
\tagpdfparaOff *29*, 288
\tagpdfparaOn *29*, 288
\tagpdfsetup *27*, 6
\tagpdfsuppressmarks *29*, 290
tagstruct *4*, 114
\tagstructbegin *28*, 32, 180
\tagstructend *28*, 32, 181
tagstructobj *4*, 114
\tagstructuse *28*, 32
tagunmarked␣(setup-key) *4*, 188

TeX and LaTeX 2ε commands:
\@auxout 51
\@bsphack 131
\@cclv 274
\@esphack 133
\@ifpackageloaded 28
\@kernel@after@foot 304
\@kernel@after@head 302
\@kernel@before@cclv 271
\@kernel@before@foot 303
\@kernel@before@footins 267, 269
\@kernel@before@head 299, 301
\@makecol 273
\@mult@ptagging@hook 276
\c@page 273
\count@ 281
\mult@firstbox 279
\mult@rightbox 283
\page@sofar 278
\process@cols 279
\the 273
\tiny 232, 246
tl commands:
\c_space_tl 273
\tl_if_empty:NTF 171
\tl_if_empty:nTF 125
\tl_if_eq:NNTF 252
\tl_if_exist:NTF 75
\tl_new:N 82
\tl_put_left:Nn 302, 304
\tl_put_right:Nn 269, 271, 276, 301, 303
\tl_set:Nn 203, 206
\tl_to_str:n 71, 275, 308
\tl_use:N 76
\l_tmpa_tl 118, 130
token commands:
\token_to_str:N 53, 273
tree-mcid-index-wrong *16*, 39

U

\unskip *27*, 19
use commands:
\use:N 47, 291
\use_none:n 43