The tagpdf-checks module
Messages and check code
part of the tagpdf package

Ulrike Fischer*

Version 0.82, released 2021-06-14

```
⟨@@=tag⟩
⟨*checks⟩
\ProvidesExplPackage {tagpdf-checks-code} {2021-06-14} {0.82}
 {part of tagpdf - code related to checks and messages}
```

1 Messages and checks

1.1 Messages

1.1.1 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.

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

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

`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* ??.)

`mc-label-unknown` If the label of a mc that is used in another place is not known (yet)

```
\msg_new:nnn { tag } {mc-label-unknown} { label~#1~unknown~-~rerun }
```

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

`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* ??.)

`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 }
```

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

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

`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* ??.)

1.1.2 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* ??.)

`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* ??.)

`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* ??.)

`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* ??.)

`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* ??.)

`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* ??.)

1.1.3 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* ??.)

1.2 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* ??.)

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* ??.)

1.2.1 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* ??.)

obj-write-num

An info message, useful for reporting.

```
\msg_new:nnn { tag } {obj-write-num} {write~obj~#1~to~pdf}
```

(*End definition for* `obj-write-num`. *This function is documented on page* ??.)

sys-no-interwordspace

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

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

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

1.3 Checks

These are checks used in various places in the code.

1.3.1 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.

```
\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.

```
\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
  {
    \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`.)

1.3.2 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
          {
            \msg_info:nnnn { tag } {role-tag} {#1} {#2}
          }
          {
            \msg_warning:nnn { tag } {role-unknown} {#2}
          }
      }
  }
```

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

1.3.3 Check related to mc-chunks

`\__tag_check_mc_if_nested:`
`\__tag_check_mc_if_open:`

Two tests if a mc is currently open.

```
\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_warning: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_seq` This variable holds the list of used mc. It will hopefully be rather short, so checking the seq is not to slow.

```
\seq_new:N \g__tag_check_mc_used_seq
```

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

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

```
\cs_new_protected:Npn \__tag_check_mc_used:n #1
  {
    \seq_if_in:NnTF \g__tag_check_mc_used_seq {#1}
      {
        \msg_warning:nnn { tag } {mc-used-twice} {#1}
      }
      {
        \seq_gput_right:Nx \g__tag_check_mc_used_seq {#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:`.)

1.3.4 Miscellaneous

`\__tag_check_record_pdfobj_num:n` This writes the object numbers. Not sure if needed, currently unused.

```
\cs_new_protected:Npn \__tag_check_record_pdfobj_num:n #1
  {
    \int_compare:nT { \l__tag_loglevel_int >= 3 }
      {
        \msg_info:nnx { tag } {obj-write-num} {#1}
      }
  }
⟨/checks⟩
```

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

Index

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