The latex-lab-sec package
Changes related to the tagging of sectioning commands

LaTeX Project*

v0.84l 2026-01-19

Abstract

The following code implements a first draft for the tagging of sectioning commands.

1 New implementation with templates

Various parts of this module are obsolete as the commands have been reimplemented in the module `latex-lab-sec-template` with templates. The tagging sockets are still relevant.

In a later step both files will be merged.

2 Limitations

Sectioning commands are in general not defined by the format but by the classes. Their implementation vary: some are defined with the help of `\@startsection`, some are like `\chapter` handcrafted, some build with the help of extension packages or as in the KOMA classes with class code that extends the `\@startsection` functionality.

The following code can therefore currently be used *only* with the standard classes or with classes which do not overwrite the changed definitions.

3 Introduction

Tagging of sectioning commands consist of two parts:

- The heading/title text of the section should be surrounded by a heading tag, typically `Hn` with some value of `n`. In theory, one could put the number of the section command in an `Lbl`. However, current AT doesn't handle this well, so we use a tag `section-number` that is mapped to `Span`. The number of the `Hn` tag should reflect the "natural" level. So in an article `\section` will use `H1`, in a book `\chapter` will use `H1` and `\section` `H2`. Titles of `\part` are a bit out of this system as they are normally not part of the hierarchy: often only some chapters are grouped under a part. Their title is therefore tagged as `Title`.

*Initial implementation done by Ulrike Fischer

- The whole section should normally be surrounded by a `Sect` tag. Parts should be surrounded by `Part`. It is a bit unclear if the headings should be inside or outside of these structures—the best practice guide puts them outside—but on the whole it sounds more logical to group the heading with the text inside the `Sect`. For the part this is actually required, as there can be only one `Title` in a structure, so the part title can't be at the same level as the document `Title`.

  Starting such an enclosing `Sect` structure is rather easy, but closing it requires code in various place, for example the commands `\mainmatter`, `\backmatter`, `\frontmatter` and `\appendix` should typically close everything. Following sectioning commands should close all previous structures with a level equal or higher than their own level.

4 Technical details and problems

The implementation has to take care of various details.

- As sections in LaTeX are not environments, the `<Sect>` structures can be wrongly nested with other structures. For example if a document puts a sectioning command into a list or a trivlist or a minipage then it can no longer close previous `<Sect>` structures correctly. The problem can be detected by checking the structure stack and a warning can be issued, but the author then has to close the structures manually before the list or minipage.

  Thus there have to be user interfaces to handle such cases. It should also be possible not to create all the `<Sect>` structures automatically but to tag only the headings so that the author can handle special cases manually.

- If hyperref is used, targets for links should be inserted, either with `\refstepcounter` or manually with `\MakeLinkTarget`. These targets must be in the correct structure for the structure destinations. They replace some of the current patches in hyperref.

4.1 Functions and keys

`\tag_tool:n` *deprecated*, use tagging sockets instead.
`\tagtool`

4.2 TODO

- A dedicated command to close a sectioning unit should be provided.
- A dedicated command to open a sectioning unit should be provided too.
- It should also be possible to suppress the sectioning unit in sectioning commands to allow e.g. to put an epigraph or similar in front.
- The number in `\part` and `\chapter` is currently not correctly tagged as a `section-number` as this requires to redefine the internal (class dependent) commands too.

1 ⟨*package⟩

5 Implementation

```
\ProvidesExplPackage {latex-lab-testphase-sec} {\ltlabsecdate} {\ltlabsecversion}
  {Code related to the tagging of sectioning commands}
```

.

5.1 Temporary fix

Until tagpdf correctly sets the symbolic name (2025-10-06)

```
\tl_set:Nn  \l__tag_para_tag_default_tl { \UseStructureName {para/textblock} }
\tl_set:Nn  \l__tag_para_main_tag_tl    { \UseStructureName {para/semantic} }
```

5.2 Surrounding by `Sect` structures

We use a stack to record the levels of the open `Sect`. The first item has level -100. A sectioning command will take a record from the stack. If its level is greater or equal it closes this structure and takes the next record from the stack. If the record has a smaller level then it puts it back and stops. The stack is compared with the main structure stack, if they don't match it means we can't safely close the `Sect` and so we issue a warning and do nothing.

```
⟨/package⟩
```

5.2.1 Glyphtounicode improvements

As lualatex runs with legacy encodings in the test files, we enable and load glyphtounicode. For the math we load additional definitions.

```
⟨*kernelchange⟩
\ifdefined\directlua
 \ifnum\outputmode > 0
   \pdfvariable gentounicode =1
   \protected\def\pdfglyphtounicode {\pdfextension glyphtounicode }
   \protected\edef\pdfgentounicode  {\pdfvariable gentounicode}
   \input{glyphtounicode}
 \fi
\fi
\ifdefined\pdfglyphtounicode
 \input{glyphtounicode-cmex}
\fi
⟨/kernelchange⟩
```

```
⟨*package⟩
⟨@@=tag⟩
```

5.2.2 Tagging commands

`\g__tag_sec_stack_seq` The stack holds the tag, the level and the structure number.

```
\seq_new:N   \g__tag_sec_stack_seq
\seq_gpush:Nn\g__tag_sec_stack_seq {{Document}{-100}{2}}
```

__tag_get_data_current_Sect: This allows to retrieve the number of the current Sect structure (or Document if we are outside any Sect) with `\tag_get:n{current_Sect}`

```
\cs_new:Npn \__tag_get_data_current_Sect:
 {
   \exp_last_unbraced:Ne\use_iii:nnn{\seq_item:Nn\g__tag_sec_stack_seq{1}}
 }
```

(*End of definition for* `\__tag_get_data_current_Sect:`.)

\l__tag_sec_Sect_bool This boolean controls if a Sect structure is opened.

```
\bool_new:N     \l__tag_sec_Sect_bool
\bool_set_true:N\l__tag_sec_Sect_bool
```

\l__tag_sec_tmpa_tl a temp variable

```
\tl_new:N     \l__tag_sec_tmpa_tl
```

__tag_sec_begin:nn This starts a sectioning structure (the „Sect environment“). Currently the default tag is either Sect or Part, depending on the level, but this can be changed by adapting the symbolic structure names which are built from the level. The second argument allows to add more options but is currently unused.

```
\cs_new_protected:Npn\__tag_sec_begin:nn #1 #2 %#1 level #2 keyval
  {
    \tag_struct_begin:n
      {
         tag= \UseStructureName{sec/#1}
        ,#2
      }
    \seq_gpush:Ne \g__tag_sec_stack_seq
      {{\g__tag_struct_tag_tl}{\int_eval:n{#1}}{\g__tag_struct_stack_current_tl}}
  }
\cs_generate_variant:Nn \__tag_sec_begin:nn {en}
```

(*End of definition for* `\__tag_sec_begin:nn`.)

__tag_sec_end:n

```
\msg_new:nnn { tag } {wrong-sect-nesting}
  {
    The~structure~#1~can~not~be~closed.\\
    It~is~not~equal~to~the~current~structure~#2~on~the~main~stack
  }

\cs_new_protected:Npn\__tag_sec_end:n #1 % #1 level
  {
    \seq_get:NN \g__tag_sec_stack_seq \l__tag_tmpa_tl
    \int_compare:nNnT {#1}<{\exp_last_unbraced:NV\use_ii:nnn\l__tag_tmpa_tl+1}
      {
        \seq_get:NN\g__tag_struct_tag_stack_seq \l__tag_tmpb_tl
        \exp_args:Nee
          \tl_if_eq:nnTF
            {\exp_last_unbraced:NV\use_i:nnn\l__tag_tmpa_tl}
```

```
            {\exp_last_unbraced:NV\use_i:nn\l__tag_tmpb_tl}
            {
              \seq_gpop:NN \g__tag_sec_stack_seq \l__tag_tmpa_tl
              \tag_struct_end:
              \__tag_sec_end:n {#1}
            }
            {
              \msg_warning:nnee {tag}{wrong-sect-nesting}
               { \exp_last_unbraced:NV\use_i:nnn \l__tag_tmpa_tl }
               { \exp_last_unbraced:NV\use_i:nn \l__tag_tmpb_tl }
            }
        }
   }
```

(*End of definition for* `\__tag_sec_end:n`.)

`\__tag_sec_title_split:`
`\__tag_sec_restore_para:`

Runin-sectioning command must separate the heading from the following text. The code is in an `\everypar` which is perhaps executed in a group (e.g. when a list follows), we have to ensure that the restoring of the para can escape.

```
\cs_new_protected:Npn\__tag_sec_restore_para:
 {
   \UseTaggingSocket {para/restore}
   \if_int_compare:w \tex_currentgrouptype:D =14        % semi-simple group
     \group_insert_after:N \__tag_sec_restore_para:
   \else:
       \if_int_compare:w \tex_currentgrouptype:D =\c_one_int  % simple group
         \group_insert_after:N \__tag_sec_restore_para:
       \fi:
   \fi:
 }
\cs_new_protected:Npn \__tag_sec_title_split:
  {
```

This ends the title structure. As the begin is from the automatic (flattened) para-tagging we have to increase the counter.

```
    \tag_mc_end:
    \tag_struct_end:
    \__tag_gincr_para_end_int:
```

In case something (e.g. a list) did reset the boolean we need to close also a semantic paragraph.

```
    \bool_if:NF\l__tag_para_flattened_bool
      {\UseTaggingSocket{para/semantic/end}{}}
```

Now restore the para-tagging and start a normal paragraph:

```
    \__tag_sec_restore_para:
    \UseTaggingSocket{para/begin}
  }
```

(*End of definition for* `\__tag_sec_title_split:` *and* `\__tag_sec_restore_para:`.)

`\__tag_sec_title_begin:nn`

This command is used in the socket at the begin of display sectioning commands like part and chapter. It takes two arguments: the level and the title.

```
\cs_new_protected:Npn \__tag_sec_title_begin:nn #1 #2 %level, title
  {
    \protected@edef\l__tag_sec_tmpa_tl{#2}
```

```
    \tag_struct_begin:n{tag=\UseStructureName{sec/#1/title},title-o={\l__tag_sec_tmpa_tl}}
    \bool_set_true:N\l__tag_para_flattened_bool
    \tl_set:Nn\l__tag_para_tag_tl {\UseStructureName{sec/#1/titleline}}
  }
```

(*End of definition for* `\__tag_sec_title_begin:nn`.)

`\__tag_sec_title_end:`

```
\cs_new_protected:Npn \__tag_sec_title_end:
  {
    \tag_struct_end: %P = Hn
    \UseTaggingSocket{para/restore}
  }
```

(*End of definition for* `\__tag_sec_title_end:`.)

`\__tag_set_title_hang:nnn` To be able to correctly tag the number and insert the link target in the title of a sectioning command created with `\@startsection` we need a special`\@hangfrom` variant. This is a bit tricky: The argument contains the link target and for a correct structure destination it should be typeset *after* the structure has been opened. But to measure the hangindent it must be typeset *before* the paragraph is started. This means that we have to open the title structure manually and then have to suppress the para-tagging. Additionally there is an engine difference: with pdftex the literals for the mc are inserted with the box after the paragraph has started but luatex sets the attributes before and we have to reset them. Hiding all this in a tagging socket is non-trivial. The code assumes that we are in vmode! Attention: The code opens a structure that it doesn't close (it is closed by the `\par`). It therefore does not handle the full tagging of the title. In a new implementation of the sectioning command this will perhaps have to change.

```
\cs_new_protected:Npn \__tag_set_title_hang:nNnn #1 #2 #3 #4
  %#1 level,
  %#2 boolean: nonumber? (will be later \l__head_nonumber_bool)
  %#3 formated number /hang space
  %#4 title
```

The handling of the title is not perfect. It would be better to pass it through something like `\GetTitleString`. TODO.

```
  {
    \protected@edef\l__tag_sec_tmpa_tl {#4}
    \tagstructbegin{tag=\UseStructureName{sec/#1/title},title-o={\l__tag_sec_tmpa_tl}}
    \cs_if_exist_use:N \__tag_gincr_para_begin_int:
    \bool_if:NF #2
     { \tagstructbegin{tag=\UseStructureName{sec/#1/number}} }
    \setbox\@tempboxa\hbox{{#3}}
```

We stop paratagging now, to avoid that the `\noindent` creates a structure.

```
    \bool_set_false:N \l__tag_para_bool
    \hangindent \wd\@tempboxa\noindent
```

Restart paratagging and insert the box. If the box has a real content (if there is a number) we have to add mc-chunks and reset the attribute of the box.

```
    \bool_set_true:N \l__tag_para_bool
    \bool_if:NTF #2
     {
       \box\@tempboxa
     }
```

```
      {
        \tagmcbegin{}
```

In lua mode we have to reset the attributes inside the box!

```
        \tag_mc_reset_box:N\@tempboxa
        \box\@tempboxa
        \tagmcend
        \tagstructend
      }
    \tagmcbegin{}
  }
```

(*End of definition for* `\__tag_set_title_hang:nnn`.)

`\__tag_sec_title_runin_number:nn`

```
\cs_new_protected:Npn \__tag_sec_title_runin_number:nNn #1 #2 #3 %#1 level, #2 boolean no num
  {
    \bool_if:NTF #2
    { #3 }
    {
      \tag_mc_end_push:
      \tag_struct_begin:n{tag=\UseStructureName{sec/#1/number}}
      \tag_mc_begin:n{}
      #3
      \tag_mc_end:
      \tag_struct_end:
      \tag_mc_begin_pop:n{}
    }
  }
```

(*End of definition for* `\__tag_sec_title_runin_number:nn`.)

Open sec structures should be closed at the end of the document. This should be done before tagpdf closes the Document structure.

```
\hook_gput_code:nnn
  {tagpdf/finish/before}
  {tagpdf/sec}
  {\AssignTaggingSocketPlug{sec/end}{kernel}\UseTaggingSocket{sec/end}{-10}}
\hook_gset_rule:nnnn {tagpdf/finish/before}{tagpdf/sec}{before}{tagpdf}
```

The commands `\mainmatter`, `\backmatter`, `\frontmatter` and `\appendix` close all `Sect` and `Part` structures.

```
\AddToHook{cmd/frontmatter/before}{\par\UseTaggingSocket{sec/end}{-10}}
\AddToHook{cmd/mainmatter/before} {\par\UseTaggingSocket{sec/end}{-10}}
\AddToHook{cmd/backmatter/before} {\par\UseTaggingSocket{sec/end}{-10}}
\AddToHook{cmd/appendix/before}   {\par\UseTaggingSocket{sec/end}{-10}}
```

5.3 Tagging Sockets

First the sockets that handle the Sect structures.

The argument of the begin socket consists of two brace groups, in the first brace is the level, in the second the keys for the structure.

```
\NewTaggingSocketPlug{sec/begin}{kernel}
  {
     \__tag_sec_begin:en #1
```

```
  }
\AssignTaggingSocketPlug{sec/begin}{kernel}
```

The end socket takes as argument only the level to close.

```
\NewTaggingSocketPlug{sec/end}{kernel}
  {
     \__tag_sec_end:n {#1}
  }
\AssignTaggingSocketPlug{sec/end}{kernel}
```

These two sockets handle the tagging of headings with special formatting like part and chapter. The argument of the begin socket is two brace groups containing the level and the title.

```
\NewTaggingSocketPlug{sec/title/begin}{kernel}
 {
   \__tag_sec_title_begin:nn #1
 }
\AssignTaggingSocketPlug{sec/title/begin}{kernel}
```

The end socket does not take an argument. It only closes the structures and restores the para settings.

```
\NewTaggingSocketPlug{sec/title/end}{kernel}
 {
   \__tag_sec_title_end:
 }
\AssignTaggingSocketPlug{sec/title/end}{kernel}
```

The `sec/title/hang` socket is used to typeset the heading of a title using `\@hangfrom`. It is the most tricky one. It takes two argument. The second argument of this socket will pass the normal `\@hangfrom` command if tagging is not active. It is not used with tagging. The first argument passes four brace groups: the level, a boolean for "nonumber", the actual content of the number and the title.

Attention: The socket opens a structure that it doesn't close (it is closed by the `\par`). It therefore does not handle the full tagging of the title. In a new implementation of the sectioning command this will perhaps have to change.

```
\NewTaggingSocketPlug{sec/title/hang}{kernel}
  {
    \__tag_set_title_hang:nNnn #1
  }
\AssignTaggingSocketPlug{sec/title/hang}{kernel}
```

The `sec/title/init` socket is used to do some initialization for sectioning commands that are setup with `\@startsection`. It sets the tag name and flattens the para-tagging. It is mostly needed for run-in headings which set the heading inside `\everypar`. It takes 1 argument, the level.

```
\NewTaggingSocketPlug{sec/title/init}{kernel}
 {
   \tl_set:Ne\l__tag_para_tag_tl{\UseStructureName{sec/#1/title}}
   \bool_set_true:N \l__tag_para_flattened_bool
 }
\AssignTaggingSocketPlug{sec/title/init}{kernel}
```

This socket handles the tagging between a run-in heading and the following text. It takes no argument.

```
\NewTaggingSocketPlug{sec/title/split}{kernel}
```

```
{
  \__tag_sec_title_split:
}
\AssignTaggingSocketPlug{sec/title/split}{kernel}
```

The following tagging socket command is used to handle the tagging of the number in the title of run-in headings. Similar to the hang variant it does not handle the full structure but relies in part on the paragraph tagging. This again can change if sectioning commands are reimplemented. It takes as first argument the level and a boolean for the numbering. The second argument contains the formatted number (if numbered) and the destination.

```
\NewTaggingSocketPlug{sec/title/runin/number}{kernel}
{
  \__tag_sec_title_runin_number:nNn #1 {#2}
}
\AssignTaggingSocketPlug{sec/title/runin/number}{kernel}
```

The following tagging socket command simply tags a number. It is not used here, as the legacy code needs a more complicated setup.

It takes the level as first argument. The second argument contains the formatted number.

```
\NewTaggingSocketPlug{sec/title/number}{kernel}
{
  \tag_mc_end_push:
  \tag_struct_begin:n{tag=\UseStructureName{sec/#1/number}}
  \tag_mc_begin:n{}
    #2
  \tag_mc_end:
  \tag_struct_end:
  \tag_mc_begin_pop:n{}
}
\AssignTaggingSocketPlug{sec/title/number}{kernel}
```

6 Sectioning commands

6.1 `\part` and `\chapter`

`\part` and `\chapter` are defined by the classes. To tag them we redefine the user commands. This will probably break with various classes and with titlesec. The tagging inside relies on the para tagging. We do not yet use keyval in the optional argument, as this requires latex-dev and the naming of the keys and their key family is unclear.

6.2 Sectioning commands based on `\@startsection`

The tagging relies again on the para tagging: we simply exchange the tag name by the one given as #1. This assumes that a tag with the name of the sectioning type is defined. We don't try to pass the title, this will be done together with the new keyval handling in the user command.

6.2.1 Hyperref code

hyperref has to insert anchors. If the sectioning is numbered this is done by `\refstepcounter` (and so in vmode). For unnumbered section hyperref injects the anchor in hmode before the text, it also inserts a kern to compensate the indent.

This means that the target of numbered and unnumbered sectioning commands differ, both regarding the location and in relation to the tagging structure: The anchor from the `\refstepcounter` is outside of the structure created by the heading title if the para tags are used, while the other anchors are inside and so the structure destinations are different.

We unify this by suppressing the anchor from the refstepcounter. Also we only go back if the indent is positive.

At first suppress all hyperref patches related to sectioning:

```
\def\hyper@nopatch@sectioning{}
```

`\l__kernel_sec_nonumber_bool`

A boolean to keep track if a sectioning command should be numbered or not.

```
\bool_new:N\l__kernel_sec_nonumber_bool
```

`\@hyp@section@target@nnn` A simple internal command. There is no need for something public, as packages defining their own version of `\@startsection` will probably need something slightly different based on `\MakeLinkTarget`.

```
\cs_new_protected:Npn \@hyp@section@target@nnn #1 #2 #3 %#1 optarg #2 name/counter, #3 indent
  {
    \makebox[0pt][l]
     {
       \skip_set:Nn \@tempskipa {#3}
       \dim_compare:nNnF {\@tempskipa}<{0pt}{\kern-\@tempskipa}
       \MakeLinkTarget#1{#2}
     }
  }
```

(*End of definition for* `\@hyp@section@target@nnn`.)

`\__tag_makecurrentHref:n` In run-in heading we have to set the name of the anchor before it is actually created in a everypar. If hyperref is loaded we could use `\Hy@MakeCurrentHrefAuto` but without it we need a similar command.

```
\cs_new_protected:Npn \@kernel@makecurrentHref #1 %#1 prefix
  {
    \int_gincr:N\g__kernel_target_int
    \tl_gset:Ne \@currentHref {#1.\int_use:N\g__kernel_target_int}
  }
```

hyperref uses a different counter so we need to use a different command, TODO: merge that.

```
\AddToHook{package/hyperref/after}
 {
  \cs_set_eq:NN \@kernel@makecurrentHref \Hy@MakeCurrentHrefAuto
 }
```

(*End of definition for* `\__tag_makecurrentHref:n`.)

6.3 Adaption of the heading commands

We add to `\@startsection` the commands to open the `Sect` structure and to change the para tag.

We save the current level so that unnumbered sections can use it too.

`\@currentseclevel`

```
\newcommand\@currentseclevel{-2}
```

(*End of definition for* `\@currentseclevel`*. This function is documented on page* **??**.)

```
\def\@startsection#1#2#3#4#5#6{%
  \def\@currentseclevel{#2}
  \if@noskipsec \leavevmode \fi
  \par
  \@tempskipa #4\relax
  \@afterindenttrue
  \ifdim \@tempskipa <\z@
    \@tempskipa -\@tempskipa \@afterindentfalse
  \fi
  \if@nobreak
    \everypar{}%
  \else
    \addpenalty\@secpenalty\addvspace\@tempskipa
  \fi
  \UseTaggingSocket{sec/end}
      {#2}
  \UseTaggingSocket{sec/begin}
      {
        {#2}
        {tag=\cs_if_exist_use:cF{g_tag_role_#1_tl}{Sect}}
      }
```

This tagging sockets changes the para-tagging: it is flattened and the tag is changed. This is actually used only by runin headings, but nevertheless this looks like the best place to use it as we can not make the changes inside some `\everypar` command.

```
  \UseTaggingSocket{sec/title/init}{#2}
  \@ifstar
    {\@ssect{#3}{#4}{#5}{#6}}%
    {\@dblarg{\@sect{#1}{#2}{#3}{#4}{#5}{#6}}}}
```

`\@sect` is only changed to replace the hyperref patches and to use the new `\@kernel@tag@hangfrom`.

```
⟨@@=⟩
\def\@sect#1#2#3#4#5#6[#7]#8{%
% #1= name, #2= level, #3= indent #4 unused #5 after vspace #6 formatting #7=short title, #8=
  \ifnum #2>\c@secnumdepth
    \bool_set_true:N\l__kernel_sec_nonumber_bool
    \@kernel@makecurrentHref {#1*}
    \def\@svsec{\NoCaseChange{\@hyp@section@target@nnn{*}{\@currentHref}{#3}}}
  \else
    \bool_set_false:N\l__kernel_sec_nonumber_bool
    \LinkTargetOff
    \refstepcounter{#1}%
    \tl_gset:Ne\@currentHref{#1.\use:c{theH#1}}
    \LinkTargetOn
```

```
    \protected@edef\@svsec{\NoCaseChange{\@hyp@section@target@nnn{}{#1}{#3}}\@seccntformat{#1
  \fi
  \@tempskipa #5\relax
  \ifdim \@tempskipa>\z@
    \begingroup
      #6{%
```

The formatting can contain a `\MakeUppercase` so we must protect the name of the socket:

```
        \NoCaseChange
         {\UseTaggingSocket{sec/title/hang}
            {{#2}\l__kernel_sec_nonumber_bool{\hskip #3\relax\@svsec}{#7}}
            {\@hangfrom {\hskip #3\relax\@svsec}}}
         \interlinepenalty \@M #8\@@par}%
    \endgroup
    \csname #1mark\endcsname{#7}%
    \addcontentsline{toc}{#1}{%
      \ifnum #2>\c@secnumdepth \else
        \protect\numberline{\csname the#1\endcsname}%
      \fi
      #7}%
  \else
    \def\@svsechd{%
      #6{\hskip #3\relax
         \NoCaseChange{
           \UseTaggingSocket{sec/title/runin/number}{{#2}\l__kernel_sec_nonumber_bool}{\@svse
        #8}%
      \csname #1mark\endcsname{#7}%
      \addcontentsline{toc}{#1}{%
        \ifnum #2>\c@secnumdepth \else
          \protect\numberline{\csname the#1\endcsname}%
        \fi
        #7}}%
  \fi
  \@xsect{#5}}
```

similar for `\@ssect`

```
\def\@ssect#1#2#3#4#5{%
  \@tempskipa #3\relax
  \ifdim \@tempskipa>\z@
    \begingroup
      #4{
         \NoCaseChange
          {
            \UseTaggingSocket{sec/title/hang}
             {
               {\@currentseclevel}
               \c_true_bool
               {\hskip #1\relax\NoCaseChange{\@hyp@section@target@nnn{[section]}{}{#1}}}
               {#5}
             }
             {\@hangfrom{\hskip #1\relax\NoCaseChange{\@hyp@section@target@nnn{[section]}{}{#
          }
        \interlinepenalty \@M #5\@@par}%
    \endgroup
  \else
```

```
    \@kernel@makecurrentHref{section*}
    \def\@svsechd{#4{\hskip #1\relax\NoCaseChange{\@hyp@section@target@nnn{*}{\@currentHref}{
  \fi
  \@xsect{#3}}
```

At last `\@xsect` needs code in two places. For display headings it has to restore the default para code, for run in headings it has to separated the heading from the following text.

```
\def\@xsect#1{%
  \@tempskipa #1\relax
  \ifdim \@tempskipa>\z@
    \par \nobreak
    \vskip \@tempskipa
    \UseTaggingSocket {para/restore}
    \@afterheading
  \else
    \@nobreakfalse
    \global\@noskipsectrue
    \everypar{%
      \if@noskipsec
        \global\@noskipsecfalse
       {\setbox\z@\lastbox}%
        \clubpenalty\@M
        \begingroup \@svsechd \endgroup
        \unskip
        \UseTaggingSocket{sec/title/split}
        \@tempskipa #1\relax
        \hskip -\@tempskipa
      \else
        \clubpenalty \@clubpenalty
        \everypar{}%
      \fi}%
  \fi
  \ignorespaces}
```

6.4 Keys for \tagpdfsetup

We need to provide user and package level commands

```
\keys_define:nn{__tag / setup}
 {
   ,sec/end .code:n =
      {
        \par
        \UseTaggingSocket{sec/end}{\int_eval:n{\cs_if_exist_use:c{toclevel@#1}+0}}
      }
   ,sec/end .value_required:n = true
   ,sec/grouping .choice:,
   ,sec/grouping / true .code:n =
     {
       \AssignTaggingSocketPlug{sec/begin}{kernel}
       \AssignTaggingSocketPlug{sec/end}{kernel}
     }
   ,sec/grouping / false .code:n =
     {
```

```
      \AssignTaggingSocketPlug{sec/begin}{noop}
      \AssignTaggingSocketPlug{sec/end}{noop}
    }
  ,sec/grouping .default:n = true
}
```

6.4.1 Tagging tools (deprecated)

`\tag_tool:n` is deprecated.

```
\cs_if_free:NT \tag_tool:n
 {
   \cs_new_protected:Npn \tag_tool:n #1
    {
      \tag_if_active:T { \keys_set:nn {tag / tool}{#1} }
    }
   \cs_set_eq:NN\tagtool\tag_tool:n
 }
\keys_define:nn { tag / tool}
  {
    ,sec-start-part .code:n =
      {
        \UseTaggingSocket{sec/end}{-1}
        \UseTaggingSocket{sec/begin}{{-1}{tag=\UseStructureName{sec/-1}}}
        \UseTaggingSocket{sec/title/begin}{{-1}{#1}}
```

We remap here the text-unit from the paragraph to NonStruct. It would be better to suppress it completely as with the other sectioning commands, but this would require to redefine `\@spart` and `\@part`, as there is the grouping, and these commands are all slightly different in the standard classes. So this is delayed to the time when sectioning commands are redefined with templates.

```
      }
    ,sec-stop-part .code:n = {\UseTaggingSocket{sec/title/end}}
    ,sec-start-chapter .code:n =
     {
        \UseTaggingSocket{sec/end}{0}
        \UseTaggingSocket{sec/begin}{{0}{tag=\UseStructureName{sec/0}}}
        \UseTaggingSocket{sec/title/begin}{{0}{#1}}
     }
    ,sec-stop-chapter .meta:n = { sec-stop-part}
    ,sec-start .code:n = % #1 is a name like "section"
      {
        \UseTaggingSocket{sec/end}  {\int_eval:n{\cs_if_exist_use:c{toclevel@#1}+0}}
        \UseTaggingSocket{sec/begin}
          {
            {\int_eval:n{\cs_if_exist_use:c{toclevel@#1}+0}}
            {tag=\cs_if_exist_use:cF{g_tag_role_#1_tl}{Sect}}
          }
        \tl_set:Nn\l_@@_para_tag_tl{#1}
      }
    ,sec-start .value_required:n = true
    ,sec-split-para .code:n = {\UseTaggingsocket{sec/title/split}}
    ,restore-para .code:n = {\UseTaggingSocket{para/restore}}%
    ,sec-stop .code:n =
      {
```

```
        \par
        \UseTaggingSocket{sec/end}{\int_eval:n{\cs_if_exist_use:c{toclevel@#1}+0}}
      }
    ,sec-stop .value_required:n = true
    ,sec-add-grouping .choice:,
    ,sec-add-grouping / true .code:n =
     {
       \AssignTaggingSocketPlug{sec/begin}{kernel}
       \AssignTaggingSocketPlug{sec/end}{kernel}
     }
    ,sec-add-grouping / false .code:n =
     {
       \AssignTaggingSocketPlug{sec/begin}{noop}
       \AssignTaggingSocketPlug{sec/end}{noop}
     }
    ,sec-add-grouping .default:n = true
  }
⟨/package⟩
⟨*latex-lab⟩
\ProvidesFile{sec-latex-lab-testphase.ltx}
        [\ltlabsecdate\space v\ltlabsecversion\space latex-lab wrapper sec]
\RequirePackage{latex-lab-testphase-sec}
⟨/latex-lab⟩
```