The latex-lab-table package
Changes related to the tagging of tables

Frank & Ulrike, LaTeX Project*

v0.85i 2024-05-25

Abstract

The following code implements a first draft for the tagging of tables. It still has a large number of limitations and restrictions!

Contents

*Initial implementation done by Frank Mittelbach

1 Documentation

In LaTeX the word `table` is used as the name of the float environment that can contain a data table[1] along with a caption and some additional text. The environments for actual data tables have various names like `tabular`, `tabular*`, `tabularx` and `longtable`—the last should not be used inside a float and supports its own caption command.

In this documentation "table" always means such data tables and not the float environment.

Tagging of tables is on one side doesn't look very difficult: one only has to surround rows and cells by TR and TH or TD structures. But there are difficulties:

- One is that over the years various packages related to tables have been written that all change some of the internals. Inserting the tagging commands and testing all the variants and various nestings is not trivial.
- The other difficulty is that most of the existing environments to create tables do not know the concept of headers as a semantic structures.
- Headers are only produced through visual formatting, e.g., by making them bold or by underlying some color. But accessible tables need headers (and the PDF/UA standards requires them) and this means that additional syntax to declare headers (when they can't be guessed) must be developed. This is still an area for research.

Right now, this module therefore does implement only some basic support for the tagging of tables. A list of the known limitations is shown below.

The module is not loaded automatically (i.e., not yet integrated into any `phase-XX`) and by itself it doesn't activate tagging. For experimenting with table tagging it is therefore best to load it in combination with phase-III in `\DocumentMetadata`, i.e.:

```
\DocumentMetadata{testphase={phase-III,table}}
```

It will then automatically tag all table environments it already supports with the exception of tables in the header and footer of the page (where tagging is disabled). Such tables can be nested.

If a table should not be tagged as table, for example because it is merely used as a layout to ensure that the content is properly aligned or because it is a not yet (fully) supported table structure, the tagging can be disabled with `\tagpdfsetup{table/tagging=false}` or with `\tagpdfsetup{table/tagging=presentation}` [2] The first option disables the table tagging code and the content of the tabular is then treated more or less like running text. This works ok for simple tables using only hmode-cells (l/c/r) with normal text content, but fails if the table uses vmode-cells (p/m/b). In such cases the second option works better: it keeps the tagging code active but changes the tag names to `Div` and the grouping structure `NonStruct`. It also (re)sets the `table/header-rows` key to empty. The key should currently only be used in a group as there is no key (yet) to reset to the default tag names.

Inside cells the automatic tagging of paragraphs is disabled with the exception of p/m/b-type cells.

Rows do not need to contain a full number of &, missing cells are automatically added with an empty TD-structure.

[1] But it does not really have to, you can put other material into such environments.

[2] The key has been renamed. The old name 'table-tagging' still works but is deprecated. The value `presentation` refers to the ARIA role "presentation".

There is some basic support[3] for headers. With[4]

```
\tagpdfsetup{table/header-rows={⟨list of row numbers⟩}}
```

you can declare which (absolute) row numbers should be tagged as header rows. It applies to all tables until it is changed to a different list of row numbers or undone by setting the key to ⟨*empty*⟩. A row number can be negative, then the counting starts from the end of the table. There is no support for header columns yet. In a `longtable` the code will currently use the `\endhead` or `\endfirsthead` rows as header if one of these commands has been used and in that case the code ignores a `table/header-rows` setting.

You should not insert meaningful text with `!{...}` or `@{...}` or `\noalign` between the rows or columns of the table. With pdflatex such text will be unmarked, with lualatex it will be marked as artifact. Either case means that it will be currently ignored in the structure.[5]

As mentioned below the colortbl doesn't yet work properly with the tagging, but once it does, then colors inside the table will probably be simply ignored (at least initially). If such a color has a semantic meaning (like "important value") this meaning will be lost.

Feedback and problems with the code can be reported at https://github.com/latex3/tagging-project either in form of explicit issues or as a "discussion topic", whatever works best.

2 Limitations

- The code loads the array package and so does not work without it (that is not really a limitation, but can affect existing tables).
- It supports only a restricted number of tables types. Currently `tabular`, `tabular*`, `tabularx`, and `longtable`.
- the `array` environment is assumed to be part of math and tagging as a table is disabled for it.
- Row spans are not yet supported (and the multirow package is untested).
- The colortbl package breaks tagging if there are nested tables. It also breaks the filling up of incomplete rows.
- The tabularray package use a completed different method to create tables and will not be supported by this code.
- The nicematrix package is currently incompatible.
- Most other packages related to tables in LaTeX are not yet tested, that includes packages that change rules like booktabs, hhline, arydshln, hvdashln.
- `longtable` currently only works with lualatex. With other engines it breaks as its output routine clashes with the code which closes open MC-chunks at pagebreaks and if this is resolved there will probably be problems with the head and foot boxes (but this can't be tested currently).

[3] This is not meant to be the final interface, though.

[4] The old key name `table-header-rows` still works but is deprecated.

[5] While it is theoretically possible to collect such text and move it into a structure it would require manual markup from the author to clarify where this text belongs too.

- Not every table should be tagged as a Table structure, often they are only used as layout help, e.g. to align authors in a title pages. In such uses the tagging of the table must be deactivated with `\tagpdfsetup{table/tagging=false}` or `\tagpdfsetup{table/tagging=presentation}`.
- Only simple header rows are currently supported. Columns and complex headers with subheaders will be handled later as that needs some syntax changes. Tables with more than one header row are probably not pdf/UA as the headers array in the cells is missing.
- A longtable `\caption` is currently simply formatted as a multicolumn and not tagged as a `Caption` structure.
- The caption package will quite probably break the longtable caption.
- The setup for longtable requires lots of patches to internal longtable commands and so can easily break if other packages try to patch longtable too.
- The `longtable` environment supports footnotes in p-type columns, but it hasn't been tested yet if this works also with the tagging code.
- Vertical boxes (`\parbox`, `minipage`, ...) inside cells can be problematic.
- The code is quite noisy and fills the log with lots of messages.[6]

3 Introduction

4 Technical details and problems

The implementation has to take care of various details.

4.1 TODOs

- Test `array-006-longtable.lvt` and `array-007-longtable.lvt` have errors with pdftex (para tagging)
- Instead of before/after hooks we should add sockets directly into the code.
- Debugging code and messages must be improved.
- Cells need an `Headers` array.
- Row spans should be supported (but perhaps need syntax support)
- Longtable captions should be properly supported.
- Handle p-cells better. para-tagging should probably be enabled, but Part can't be a child of TD, so this should probably be changed to Div here. Also there is a stray MC somewhere.
- More packages must be tested.

[6] Helpful for us at this stage.

5 Implementation

```
⟨@@=tbl⟩
⟨*package⟩
\ProvidesExplPackage {latex-lab-testphase-table} {\ltlabtbldate} {\ltlabtblversion}
  {Code related to the tagging of tables}
```

This builds on array so we load it by default:

```
\RequirePackage{array}
```

5.1 Variables

\l__tbl_celltag_tl, \l__tbl_pcelltag_tl, \l__tbl_rowtag_tl, \l__tbl_table_tl, \l__tbl_cellattribute_tl, \l__tbl_rowattribute_tl, \l__tbl_tmpa_clist, \l__tbl_tmpa_seq, \l__tbl_tmpa_tl

This is for the celltag, e.g. TD or TH:

```
\tl_new:N  \l__tbl_celltag_tl
\tl_set:Nn \l__tbl_celltag_tl {TD}
\tl_new:N  \l__tbl_pcelltag_tl
\tl_set:Nn \l__tbl_pcelltag_tl {TD}
```

For the rowtag, probably always TR:

```
\tl_new:N  \l__tbl_rowtag_tl
\tl_set:Nn \l__tbl_rowtag_tl {TR}
```

For the tabletag, probably always Table:

```
\tl_new:N  \l__tbl_tabletag_tl
\tl_set:Nn \l__tbl_tabletag_tl {Table}
```

And here cell and row attributes:

```
\tl_new:N  \l__tbl_cellattribute_tl
\tl_set:Nn \l__tbl_cellattribute_tl {}
\tl_new:N  \l__tbl_rowattribute_tl
\tl_set:Nn \l__tbl_rowattribute_tl {}
```

Temp variables

```
\clist_new:N \l__tbl_tmpa_clist
\tl_new:N \l__tbl_tmpa_tl
```

(*End of definition for* `\l__tbl_celltag_tl` `\l__tbl_pcelltag_tl` *and others.*)

5.2 Tagging support sockets

This are the standard plugs for tagging of cells and rows.

TD (*plug*)

```
\NewSocketPlug{tagsupport/tbl/cell/begin}{TD}
  {
```

Next line was previously outside of the plug, so if we want to execute it always even if the noop plug is in force this needs a different solution.

```
    \__tbl_show_curr_cell_data:
    \tag_struct_begin:n
      {
        tag             =\l__tbl_celltag_tl,
        attribute-class =\l__tbl_cellattribute_tl
      }
    \seq_gput_right:Ne \g__tbl_struct_cur_seq { \tag_get:n {struct_num} }
```

we store the cells of multicolumns as negative number. This allow to skip them or to use them as needed.

```
    \int_step_inline:nn { \g__tbl_span_tl - 1 }
      {
        \seq_gput_right:Ne \g__tbl_struct_cur_seq { -\tag_get:n {struct_num} }
      }
    \tag_mc_begin:n{}
  }
```

TD (*plug*)

```
\NewSocketPlug{tagsupport/tbl/cell/end}{TD}
  {
    \tag_mc_end:
    \tag_struct_end:
  }
```

In p-columns we need a slightly different plug which reactivates the paragraph tagging.

TDpbox (*plug*)

```
\NewSocketPlug{tagsupport/tbl/pcell/begin}{TDpbox}
  {
    \__tbl_show_curr_cell_data:
    \tag_struct_begin:n
      {
        tag             =\l__tbl_pcelltag_tl,
        attribute-class =\l__tbl_cellattribute_tl
      }
    \seq_gput_right:Ne \g__tbl_struct_cur_seq { \tag_get:n {struct_num} }
    \int_step_inline:nn { \g__tbl_span_tl - 1 }
      {
        \seq_gput_right:Ne \g__tbl_struct_cur_seq { -\tag_get:n {struct_num} }
      }
    \tagpdfparaOn
    \tl_set:Nn \l__tag_para_main_tag_tl {Div}
  }
```

TDpbox (*plug*)

```
\NewSocketPlug{tagsupport/tbl/pcell/end}{TDpbox}
  {
    \tag_struct_end:
  }
```

TR (*plug*)

```
\NewSocketPlug{tagsupport/tbl/row/begin}{TR}
  {
    \seq_gclear:N \g__tbl_struct_cur_seq
    \tag_struct_begin:n
     {
       tag            =\l__tbl_rowtag_tl,
       attribute-class=\l__tbl_rowattribute_tl
     }
   \seq_gput_right:Ne \g__tbl_struct_rows_seq { \tag_get:n {struct_num} }
  }
```

TR (*plug*)

```
\NewSocketPlug{tagsupport/tbl/row/end}{TR}
  {
    \tag_if_active:T
      {
    \__tbl_add_missing_cells:
    \seq_gput_right:Ne \g__tbl_struct_cells_seq
      {
        \seq_use:Nn \g__tbl_struct_cur_seq {,}
      }
    \int_compare:nNnTF { \g__tbl_row_int } =
                       { \seq_count:N \g__tbl_struct_cells_seq }
      {
        \__tbl_trace:n
          {==>~
            stucture~stored~for~row~\int_use:N\g__tbl_row_int :~
            \seq_use:Nn \g__tbl_struct_cur_seq {,}
          }
      }
      { \ERROR tbl/row } % should not happen ...
    \tag_struct_end:
  }
  }
```

And the plugs for the table as whole. The code can be different for normal tables which can also be used inline and nested and "vmode" tables like longtable.

Table (*plug*) Inside a table we currently only disable paratagging. We assume that these sockets are in an environment group, so there is no need to reenable paratagging.

```
\NewSocketPlug{tagsupport/tbl/init}{Table}
  {
    \tag_if_active:T
      {
        \bool_set_false:N \l__tag_para_bool
```

We also initialize the structure data variables a this point.

```
        \__tbl_init_struct_data:
      }
  }
```

Table (*plug*) This plug will fine tune the structure.

```
\NewSocketPlug{tagsupport/tbl/finalize}{Table}
  {
    \__tbl_set_header_rows:
```

Similarly, we restore the outer values of the structure data when we leave the table.

```
    \__tbl_restore_struct_data:
  }
```

Table (*plug*) This plug will initialize the structure in longtable.

```
\NewSocketPlug{tagsupport/tbl/longtable/init}{Table}
 {
   \seq_gclear:N\g__tbl_struct_rows_seq
   \seq_gclear:N\g__tbl_struct_cells_seq
```

```
  \seq_gclear:N\g__tbl_struct_cur_seq
  \seq_gclear:N\g__tbl_LT@firsthead_rows_seq
  \seq_gclear:N\g__tbl_LT@head_rows_seq
  \seq_gclear:N\g__tbl_LT@lastfoot_rows_seq
  \seq_gclear:N\g__tbl_LT@foot_rows_seq
}
```

`Table` (*plug*) This plug will fine tune the structure in longtable.

```
\NewSocketPlug{tagsupport/tbl/longtable/finalize}{Table}
 {
```

If neither `\endhead` nor `\endfirsthead` has been used we use the standard header command:

```
    \bool_lazy_and:nnTF
      { \seq_if_empty_p:N \g__tbl_LT@head_rows_seq  }
      { \seq_if_empty_p:N \g__tbl_LT@firsthead_rows_seq }
      { \__tbl_set_header_rows: }
```

Otherwise, if firsthead has not been used we use head. For this we simple retrieve the row numbers and then call the header command.

```
      {
        \seq_if_empty:NTF \g__tbl_LT@firsthead_rows_seq
          {
            \clist_set:Ne \l__tbl_header_rows_clist
              {\seq_use:Nn \g__tbl_LT@head_rows_seq {,}}
            \__tbl_set_header_rows:
          }
```

In the other case we use firsthead.

```
          {
            \clist_set:Ne \l__tbl_header_rows_clist
              { \seq_use:Nn \g__tbl_LT@firsthead_rows_seq {,} }
            \__tbl_set_header_rows:
```

Additionally we have to remove the head to avoid duplication. The one option here is to remove the rows from the kid sequence of the table (which will lead to orphaned structure elements), the other to make them artifact. For now we use the first option for pdf 1.7 and the second for pdf 2.0.

```
            \pdf_version_compare:NnTF < {2.0}
             {
               \seq_map_inline:Nn \g__tbl_LT@head_rows_seq
                 {
                   \seq_gset_item:cnn
                     {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
                     { ##1 }
                     {}
```

Not sure if needed, but if needed we can remove also the P tag. This is currently disabled as it produce warnings. TODO: This needs also a tagpdf command which takes care of debug code.

```
                   \tl_set:Ne \l__tbl_tmpa_tl
                     { \seq_item:Nn\g__tbl_struct_rows_seq {##1} }
                   \prop_if_exist:cT
                     { g__tag_struct_ \l__tbl_tmpa_tl _prop }
                     {
```

```
                    %\prop_gremove:cn {g__tag_struct_ \l__tbl_tmpa_tl _prop} {P}
                  }
                }
              }
              {
                \seq_map_inline:Nn \g__tbl_LT@head_rows_seq
                  {
                    \tl_set:Ne \l__tbl_tmpa_tl
                      { \seq_item:Nn\g__tbl_struct_rows_seq {##1} }
                    \prop_if_exist:cT
                      { g__tag_struct_ \l__tbl_tmpa_tl _prop }
                      {
                        \exp_args:No \__tag_struct_prop_gput:nnn {\l__tbl_tmpa_tl} {S}{/Artif
                      }
                  }
              }
           }
      }
```

The foot is handled similar, the difference is that we have to move it to the end one of them is not empty, but do nothing if they aren't there.

```
    \bool_lazy_and:nnF
      { \seq_if_empty_p:N \g__tbl_LT@foot_rows_seq  }
      { \seq_if_empty_p:N \g__tbl_LT@lastfoot_rows_seq }
      {
```

If lastfoot is empty move foot to the end.

```
        \seq_if_empty:NTF \g__tbl_LT@lastfoot_rows_seq
          {
            \seq_map_inline:Nn \g__tbl_LT@foot_rows_seq
              {
                \tl_set:Ne \l__tbl_tmpa_tl
                  {
                    \seq_item:cn
                      {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
                      {##1}
                  }
                \seq_gset_item:cnn
                  {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
                  { ##1 }
                  {}
                \seq_gput_right:cV
                  {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
                  \l__tbl_tmpa_tl
              }
          }
```

If lastfoot is not empty we move that.

```
          {
            \seq_map_inline:Nn \g__tbl_LT@lastfoot_rows_seq
              {
                \tl_set:Ne \l__tbl_tmpa_tl
                  {
                    \seq_item:cn
                      {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
```

```
                    {##1}
                }
              \seq_gset_item:cnn
                {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
                { ##1 }
                {}
              \seq_gput_right:cV
                {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
                \l__tbl_tmpa_tl
            }
```

and we hide foot

```
        \pdf_version_compare:NnTF < {2.0}
          {
            \seq_map_inline:Nn \g__tbl_LT@foot_rows_seq
              {
                \seq_gset_item:cnn
                  {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
                  { ##1 }
                  {}
```

Not sure if needed, but if needed we can remove also the P tag. This is currently disabled as it produce warnings. TODO: This needs also a tagpdf command which takes care of debug code.

```
                \tl_set:Ne \l__tbl_tmpa_tl
                  { \seq_item:Nn\g__tbl_struct_rows_seq {##1} }
                \prop_if_exist:cT
                  { g__tag_struct_ \l__tbl_tmpa_tl _prop }
                  {
                    %\prop_gremove:cn {g__tag_struct_ \l__tbl_tmpa_tl _prop} {P}
                  }
              }
          }
          {
            \seq_map_inline:Nn \g__tbl_LT@foot_rows_seq
              {
                \tl_set:Ne \l__tbl_tmpa_tl
                  { \seq_item:Nn\g__tbl_struct_rows_seq {##1} }
                \prop_if_exist:cT
                  { g__tag_struct_ \l__tbl_tmpa_tl _prop }
                  {
                    \exp_args:No\__tag_struct_prop_gput:nnn {\l__tbl_tmpa_tl} {S}{/Artifa
                  }
              }
          }
      }
  }
```

`Table` (*plug*) We must avoid that the reuse of the header foot box leads to duplicated content, thus reset attribute of the box:

```
\NewSocketPlug{tagsupport/tbl/longtable/head}{Table}
 {
  \tagmcbegin{artifact}
```

```
    \tag_mc_reset_box:N\LT@head
   \tagmcend
  }
```

Table (*plug*)

```
\NewSocketPlug{tagsupport/tbl/longtable/foot}{Table}
 {
   \tagmcbegin{artifact}
   \tag_mc_reset_box:N \LT@foot
   \tagmcend
 }
```

Table (*plug*)

```
\NewSocketPlug{tagsupport/tbl/hmode/begin}{Table}
  {
    \tag_mc_end_push:
```

Close the P-chunk. This assumes that para-tagging is active. For nested tables that is not necessarily true, so we test for it.

```
     \bool_lazy_and:nnT
      { \bool_if_exist_p:N \l__tag_para_bool } { \l__tag_para_bool }
      { \tag_struct_end:n { text } }
    \tag_struct_begin:n {tag=\l__tbl_tabletag_tl}
    \tl_gset:Ne \g__tbl_struct_table_tl { \tag_get:n {struct_num} }
  }
```

Table (*plug*)

```
\NewSocketPlug{tagsupport/tbl/hmode/end}{Table}
  {
    \tag_struct_end:
```

reopen the P-chunk. This assumes that para-tagging is active. For nested tables that is not necessarily true, so we test for it.

```
    \bool_lazy_and:nnT
      { \bool_if_exist_p:N \l__tag_para_bool } { \l__tag_para_bool }
      { \tag_struct_begin:n { tag=\l__tag_para_tag_tl } }
    \tag_mc_begin_pop:n{}
  }
```

Table (*plug*)

```
\NewSocketPlug{tagsupport/tbl/vmode/begin}{Table}
  {
    \tag_struct_begin:n {tag=\l__tbl_tabletag_tl}
    \tl_gset:Ne \g__tbl_struct_table_tl { \tag_get:n {struct_num} }
  }
```

Table (*plug*)

```
\NewSocketPlug{tagsupport/tbl/vmode/end}{Table}
  {
    \tag_struct_end:
    \par
  }
```

`code` (*plug*) This socket takes a number, checks if is larger than one, checks if the colspan attribute already exists (we can't predefine an arbitrary number), and updates `\l__tbl_cellattribute_tl`.

```
\NewSocketPlug{tagsupport/tbl/colspan}{code}
  {
    \int_compare:nNnT {#1}>{1}
     {
       \prop_get:NeNF \g__tag_attr_entries_prop
          {colspan-\int_eval:n{#1}}
          \l__tbl_tmpa_tl
          {
            \__tag_attr_new_entry:ee
              {colspan-\int_eval:n{#1}}
              {/O /Table /ColSpan~\int_eval:n{#1}}
          }
       \tl_set:Ne \l__tbl_cellattribute_tl
          {colspan-\int_eval:n{#1}}
     }
  }
```

TODO: move to tagpdf

```
\tag_if_active:T
  { \cs_generate_variant:Nn \__tag_attr_new_entry:nn {ee} }

```

5.3 Environments

Currently we support only tabular, tabular*, tabularx and longtable (and possibly environments build directly on top of them).

The `array` environment is math. So we disable table tagging for now. We use the command hook to catch also cases where `\array` is used directly in other environments like matrix environments. Perhaps table tagging should be disable for math generally, but then we have to handle text insertions.

```
\AddToHook{cmd/array/before}{\__tag_tbl_disable:}
```

5.4 Interfaces to tagging

5.4.1 Tagging helper commands

5.4.2 Disabling/enabling

For now we have only the option true/false but this will probably be extended to allow different setups like first row header etc.

`\__tag_tbl_disable:`

```
\cs_new_protected:Npn \__tag_tbl_disable:
 {
   \AssignSocketPlug{tagsupport/tbl/cell/begin}{noop}
   \AssignSocketPlug{tagsupport/tbl/cell/end}{noop}
   \AssignSocketPlug{tagsupport/tbl/pcell/begin}{noop}
   \AssignSocketPlug{tagsupport/tbl/pcell/end}{noop}
   \AssignSocketPlug{tagsupport/tbl/row/begin}{noop}
   \AssignSocketPlug{tagsupport/tbl/row/end}{noop}
```

```
    \AssignSocketPlug{tagsupport/tbl/init}{noop}
    \AssignSocketPlug{tagsupport/tbl/finalize}{noop}
    \AssignSocketPlug{tagsupport/tbl/longtable/init}{noop}
    \AssignSocketPlug{tagsupport/tbl/longtable/head}{noop}
    \AssignSocketPlug{tagsupport/tbl/longtable/foot}{noop}
    \AssignSocketPlug{tagsupport/tbl/longtable/finalize}{noop}
    \AssignSocketPlug{tagsupport/tbl/hmode/begin}{noop}
    \AssignSocketPlug{tagsupport/tbl/hmode/end}{noop}
    \AssignSocketPlug{tagsupport/tbl/vmode/begin}{noop}
    \AssignSocketPlug{tagsupport/tbl/vmode/end}{noop}
    \AssignSocketPlug{tagsupport/tbl/colspan}{noop}
  }
```

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

`\__tag_tbl_enable:`

```
\cs_new_protected:Npn \__tag_tbl_enable:
  {
    \AssignSocketPlug{tagsupport/tbl/cell/begin}{TD}
    \AssignSocketPlug{tagsupport/tbl/cell/end}{TD}
    \AssignSocketPlug{tagsupport/tbl/pcell/begin}{TDpbox}
    \AssignSocketPlug{tagsupport/tbl/pcell/end}{TDpbox}
    \AssignSocketPlug{tagsupport/tbl/row/begin}{TR}
    \AssignSocketPlug{tagsupport/tbl/row/end}{TR}
    \AssignSocketPlug{tagsupport/tbl/init}{Table}
    \AssignSocketPlug{tagsupport/tbl/finalize}{Table}
    \AssignSocketPlug{tagsupport/tbl/longtable/head}{Table}
    \AssignSocketPlug{tagsupport/tbl/longtable/foot}{Table}
    \AssignSocketPlug{tagsupport/tbl/longtable/init}{Table}
    \AssignSocketPlug{tagsupport/tbl/longtable/finalize}{Table}
    \AssignSocketPlug{tagsupport/tbl/hmode/begin}{Table}
    \AssignSocketPlug{tagsupport/tbl/hmode/end}{Table}
    \AssignSocketPlug{tagsupport/tbl/vmode/begin}{Table}
    \AssignSocketPlug{tagsupport/tbl/vmode/end}{Table}
    \AssignSocketPlug{tagsupport/tbl/colspan}{code}
  }
```

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

```
% See tagpdfsetup-keys.md in tagpdf/doc for the naming scheme.
\keys_define:nn { __tag / setup }
  {
    table/tagging .choices:nn = { true, on }
      { \__tag_tbl_enable: },
    table/tagging .choices:nn = { false, off }
      { \__tag_tbl_disable: },
    table/tagging .choice:,
    table/tagging / presentation .code:n =
     {
       \__tag_tbl_enable:
       \tl_set:Nn\l__tbl_rowtag_tl   {NonStruct}
       \tl_set:Nn\l__tbl_pcelltag_tl {NonStruct}
       \tl_set:Nn\l__tbl_celltag_tl  {text}
       \tl_set:Nn\l__tbl_tabletag_tl {Div}
       \clist_clear:N \l__tbl_header_rows_clist
```

```
    },
  table/tagging .default:n = true,
  table/tagging .initial:n = true
}
```

This are the old key names kept for now for compability. They will got at some time.

```
\keys_define:nn { __tag / setup }
  {
    table-tagging .meta:n = {table/tagging={#1}}
  }
```

5.4.3 Header support

Accessible table must have header cells declaring the meaning of the data in a row or column. To allow a data cell to find it header cell(s) a number of things must be done:

- every cell meant as a header should use the tag `TH`.
- header cells should have a `Scope` attribute with the value `Column`, `Row` or `Both`. This is not needed in the first row or column of a table.
- For more complex cases both `TD` and `TH` cell can contain a `Headers` attribute, that is an array of `ID`s of `TH` cell.

For now we support only header rows.

At first we define attributes for the three standard cases: We delay to begin document as we can't know if tagpdf is already loaded.

```
\AddToHook{begindocument}
  {
   \tag_if_active:T
     {
       \tagpdfsetup
         {
           role/new-attribute =
            {TH-col}{/O /Table /Scope /Column},
           role/new-attribute =
             {TH-row}{/O /Table /Scope /Row},
           role/new-attribute =
             {TH-both}{/O /Table /Scope /Both},
         }
```

And we put all three into the class map (perhaps the next tagpdf should do that directly with role/new-attribute):

```
         \seq_gput_left:Ne\g__tag_attr_class_used_seq
           {\pdf_name_from_unicode_e:n{TH-col}}
         \seq_gput_left:Ne\g__tag_attr_class_used_seq
           {\pdf_name_from_unicode_e:n{TH-row}}
         \seq_gput_left:Ne\g__tag_attr_class_used_seq
           {\pdf_name_from_unicode_e:n{TH-both}}
     }
  }

```

\l__tbl_header_rows_clist This holds the numbers of the header rows. Negative numbers are possible and count from the last column backwards.

```
\clist_new:N  \l__tbl_header_rows_clist
```

__tbl_set_header_rows:

```
\cs_new_protected:Npn \__tbl_set_header_rows:
  {
    \clist_map_inline:Nn \l__tbl_header_rows_clist
      {
        \clist_set:Ne\l__tbl_tmpa_clist
          {
            \seq_item:Nn \g__tbl_struct_cells_seq {##1}
          }
        \clist_map_inline:Nn \l__tbl_tmpa_clist
          {
```

We can have negative numbers in the list from the multicolumn.

```
            \prop_if_exist:cT { g__tag_struct_####1_prop }
              {
                \__tag_struct_prop_gput:nnn{ ####1 }{S}{/TH}
```

This need refinement once row headers (and perhaps other attributes) are used too, but for now it should be ok.

```
                \prop_get:cnNTF
                  { g__tag_struct_####1_prop }
                  { C }
                  \l__tbl_tmpa_tl
                  {\__tag_struct_prop_gput:nne{ ####1 }{C}{[/TH-col~\l__tbl_tmpa_tl]} }
                  {\__tag_struct_prop_gput:nnn{ ####1 }{C}{/TH-col}}
              }
          }
      }
  }
```

(*End of definition for* __tbl_set_header_rows:.)

And some key support: (See tagpdfsetup-keys.md for the naming scheme.)

```
\keys_define:nn { __tag / setup }
  {
    table/header-rows .clist_set:N = \l__tbl_header_rows_clist,
```

obsolete older name:

```
    table-header-rows .meta:n = {table/header-rows={#1}}
  }
```

5.5 Misc stuff

__tbl_show_curr_cell_data: Show the row/column index and span count for current table cell for debugging.

```
\cs_new_protected:Npn \__tbl_show_curr_cell_data: {
   \__tbl_trace:n { ==>~ current~cell~data:~
                \int_use:N \g__tbl_row_int ,
                \int_use:N \g__tbl_col_int ,
                \g__tbl_span_tl
              }
```

```
}
```

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

`\__tbl_add_missing_cells:`

The storing and use of the number of missing cells must happen at different places as the testing happens at the end of the last cell of a row, but still inside that cell, so we use two commands. The one adding is used in the row/end socket.

```
\NewSocket{tbl/celldata/missingcount}{1}
\NewSocketPlug{tbl/celldata/missingcount}{code}{\tbl_count_missing_cells:n{#1}}
\AssignSocketPlug{tbl/celldata/missingcount}{code}

\cs_new:Npn \__tbl_add_missing_cells:
  {
```

The TD-socket messages are issued after the message about the end-row socket, but the structure is ok, so better issue a message for now to avoid confusion:

```
    \int_compare:nNnT \g__tbl_missing_cells_int > 0
      {
       \__tbl_trace:n {==>~
         ~Inserting~\int_use:N \g__tbl_missing_cells_int \space
          additional~cell(s)~into~previous~row:}
       \int_step_inline:nn { \g__tbl_missing_cells_int }
           {
             \UseTaggingSocket{tbl/cell/begin}
             \UseTaggingSocket{tbl/cell/end}
           }
      }
  }
```

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

`\g__tbl_struct_table_tl`
`\l__tbl_saved_struct_table_tl`
`\g__tbl_struct_rows_seq`
`\l__tbl_saved_struct_rows_seq`
`\g__tbl_struct_cells_seq`
`\l__tbl_saved_struct_cells_seq`
`\g__tbl_struct_cur_seq`
`\l__tbl_saved_struct_cur_seq`

We need to store the structure numbers for the fine tuning in the finalize socket. For now we use a rather simple system: A sequence that hold the numbers for the row structures, and one that holds comma lists for the cells.

`\g__tbl_struct_table_tl` will hold the structure number of the table, `\g__tbl_struct_rows_seq` will hold at index i the structure number of row i, `\g__tbl_struct_cells_seq` will hold at index i a comma list of the cell structure numbers of row i. `\g__tbl_struct_cur_seq` is used as a temporary store for the cell structures of the current row. We need also local version to store and restore the values.

```
\tl_new:N  \g__tbl_struct_table_tl
\tl_new:N  \l__tbl_saved_struct_table_tl
\seq_new:N \g__tbl_struct_rows_seq
\seq_new:N \l__tbl_saved_struct_rows_seq
\seq_new:N \g__tbl_struct_cells_seq
\seq_new:N \l__tbl_saved_struct_cells_seq
\seq_new:N \g__tbl_struct_cur_seq
\seq_new:N \l__tbl_saved_struct_cur_seq
```

(*End of definition for* `\g__tbl_struct_table_tl` *and others.*)

`\__tbl_init_struct_data:`

Save the global structure data variables locally so the we can restore them when the table ends (important in case of nested tables).

This is also the right point to initialize them. `\g__tbl_struct_table_tl` is set elsewhere.

```
\cs_new_protected:Npn \__tbl_init_struct_data: {
  \tl_set_eq:NN  \l__tbl_saved_struct_table_tl \g__tbl_struct_table_tl
  \seq_set_eq:NN \l__tbl_saved_struct_rows_seq \g__tbl_struct_rows_seq
  \seq_set_eq:NN \l__tbl_saved_struct_cells_seq \g__tbl_struct_cells_seq
  \seq_set_eq:NN \l__tbl_saved_struct_cur_seq \g__tbl_struct_cur_seq
%
  \seq_gclear:N\g__tbl_struct_rows_seq
  \seq_gclear:N\g__tbl_struct_cells_seq
  \seq_gclear:N\g__tbl_struct_cur_seq
}
```

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

`\__tbl_restore_struct_data:`

```
\cs_new_protected:Npn \__tbl_restore_struct_data: {
  \tl_gset_eq:NN  \g__tbl_struct_table_tl \l__tbl_saved_struct_table_tl
  \seq_gset_eq:NN \g__tbl_struct_rows_seq \l__tbl_saved_struct_rows_seq
  \seq_gset_eq:NN \g__tbl_struct_cells_seq\l__tbl_saved_struct_cells_seq
  \seq_gset_eq:NN \g__tbl_struct_cur_seq  \l__tbl_saved_struct_cur_seq
}
```

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

5.6 longtable

`\__tbl_patch_\LT@makecaption`

This patch is quite similar to the one for LaTeX's `\@makecaption` we also have to change the parbox sockets.

```
\def\__tbl_patch_LT@makecaption#1#2#3{%
  \LT@mcol\LT@cols c{%
  % test can go after merge
    \str_if_exist:cT { l__socket_tagsupport/parbox/before_plug_str }
      {
        \AssignSocketPlug{tagsupport/parbox/before}{noop}
        \AssignSocketPlug{tagsupport/parbox/after}{noop}
      }
    \hbox to\z@{\hss\parbox[t]\LTcapwidth{%
    \reset@font
    \tag_stop:n{caption}
    \sbox\@tempboxa{#1{#2:~}#3}%
    \tag_start:n{caption}
    \ifdim\wd\@tempboxa>\hsize
      #1{#2:~}#3%
    \else
      \hbox to\hsize{\hfil#1{#2:~}#3\hfil}%
    \fi
    \endgraf\vskip\baselineskip}%
  \hss}}}
```

(*End of definition for* `\__tbl_patch_\LT@makecaption`.)

Overwrite the longtable definition. That will probably break somewhere as they are various package which patch too.

```
\AddToHook{package/longtable/after}
```

```
  {
   \cs_set_eq:NN \LT@makecaption\__tbl_patch_LT@makecaption
  }
```

⟨/package⟩

⟨*latex-lab⟩

```
\ProvidesFile{table-latex-lab-testphase.ltx}
        [\ltlabtbldate\space v\ltlabtblversion\space latex-lab wrapper table]
\RequirePackage{latex-lab-testphase-table}
```

⟨/latex-lab⟩

⟨*latex-lab-alias⟩

```
\ProvidesFile{tabular-latex-lab-testphase.ltx}
        [\ltlabtbldate\space v\ltlabtblversion\space latex-lab wrapper tabular]
\RequirePackage{latex-lab-testphase-table}
```

⟨/latex-lab-alias⟩