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

Frank & Ulrike, LaTeX Project*

v0.85n 2024-09-18

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.

1.1 Loading

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.

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.

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.[2]

As mentioned below the colortbl doesn't work fully with the tagging, but when it does, colors inside the table are currently ignored. If such a color has a semantic meaning (like "important value") this meaning will be lost.

---

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

[2]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.

1.2 Untagged and presentation tables

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}` [3] 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.

1.3 Header rows and columns

There is some basic support[4] for headers. With[5]

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

you can declare which (absolute) row and column numbers should be tagged as header rows and header columns. It applies to all tables until it is changed to a different list of row or columns numbers or undone by setting the keys to ⟨*empty*⟩. A row or column number can be negative, then the counting starts from the end of the table. 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.

1.4 Spanning of cells

`\multicolumn` is supported out of the box and will create a structure with a suitable `ColSpan` attribute[6]

For cells spanning rows some preliminary support exists[7]: If you add

> `\tagpdfsetup{table/multirow={`⟨*number of rows*⟩`}`

to a cell the code will add the suitable `RowSpan` attribute and suppress the tagging of affected cells in the following rows. This will also work if the current cell is a `\multicolumn`, then the cell will span both rows and columns. It is the duty of the author to ensure that all cells below and covered by a `RowSpan` are empty! The code neither checks that nor does it tries to suppress content.

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.

---

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

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

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

[6]The code uses actually an attribute *class*. The validator PAC doesn't handle this correctly currently and complains about a missing attribute.

[7]The interface is bound to change!

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 can only be added manually (the `multirow` package is untested).

- The `colortbl` package breaks tagging if there are nested tables.

- 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 fully tested, that includes packages that change rules like `booktabs`, `hhline`, `arydshln`, `hvdashln`. Some problems have been resolved, either in the packages or through a firstaid which can be loaded the `testphase=firstaid`.

- `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).

- 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 and columns are currently supported. Complex headers with subheaders will be handled later as that needs some syntax changes. Tables with more than one header row or column 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.[8]

---

[8]Helpful for us at this stage.

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.

- More packages must be tested.

5 Implementation

```
1 ⟨@@=tbl⟩
2 ⟨*package⟩

3 \ProvidesExplPackage {latex-lab-testphase-table} {\ltlabtbldate} {\ltlabtblversion}
4   {Code related to the tagging of tables}
```

This builds on array so we load it by default:

```
5 \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_tl
\l__tbl_tmpa_str

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

```
6 \tl_new:N  \l__tbl_celltag_tl
7 \tl_set:Nn \l__tbl_celltag_tl {TD}
8 \tl_new:N  \l__tbl_pcelltag_tl
9 \tl_set:Nn \l__tbl_pcelltag_tl {TD}
```

For the rowtag, probably always TR:

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

For the tabletag, probably always Table:

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

And here cell and row attributes:

```
14 \tl_new:N  \l__tbl_cellattribute_tl
15 \tl_set:Nn \l__tbl_cellattribute_tl {}
16 \tl_new:N  \l__tbl_rowattribute_tl
17 \tl_set:Nn \l__tbl_rowattribute_tl {}
```

Variable to store cell info like which cell must be skipped when tagging multirows.

```
18 \prop_new:N\g__tbl_untagged_cells_prop
19 \prop_new:N\l__tbl_saved_untagged_cells_prop
```

Temp variables

```
20 \clist_new:N \l__tbl_tmpa_clist
21 \tl_new:N \l__tbl_tmpa_tl
22 \str_new:N \l__tbl_tmpa_str
```

(*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.

The following two sockets are defined in lttagging, but we check for their existence until the release 11/2024.

```
23 \str_if_exist:cF { l__socket_tagsupport/tbl/init/celldata_plug_str }
24 {
25   \NewSocket{tagsupport/tbl/init/celldata}{0}
26   \NewSocket{tagsupport/tbl/restore/celldata}{0}
27 }
```

default (*plug*) This socket is used inside `\tbl_init_cell_data_for_table` for the case that this a nested table in a cell.

```
28 \NewSocketPlug{tagsupport/tbl/init/celldata}{default}
29   {
30     \prop_set_eq:NN\l__tbl_saved_untagged_cells_prop \g__tbl_untagged_cells_prop
31     \prop_gclear:N \g__tbl_untagged_cells_prop
32   }
33 \AssignSocketPlug{tagsupport/tbl/init/celldata}{default}
```

default (*plug*) This socket is used inside `\tbl_restore_outer_cell_data:` and restores values when a nested table is ended.

```
34 \NewSocketPlug{tagsupport/tbl/restore/celldata}{default}
35   {
36     \prop_gset_eq:NN\g__tbl_untagged_cells_prop \l__tbl_saved_untagged_cells_prop
37   }
38 \AssignSocketPlug{tagsupport/tbl/restore/celldata}{default}
```

TD (*plug*)

```
39 \NewSocketPlug{tagsupport/tbl/cell/begin}{TD}
40   {
```

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.

```
41     \__tbl_show_curr_cell_data:
```

We test if the cell should be tagged at all.

```
42     \__tbl_if_tag_cell:nnT {\g__tbl_row_int } { \g__tbl_col_int }
43       {
```

this sets row headers

```
44          \clist_if_in:NVT \l__tbl_header_columns_clist\g__tbl_col_int
45            {
46              \tl_set:Nn \l__tbl_celltag_tl {TH}
47              \tl_set:Ne \l__tbl_cellattribute_tl {\l__tbl_cellattribute_tl,TH-row}
48            }
```

Here we handle negative value for row headers:

```
49          \tl_set:Ne \l__tbl_tmpa_tl {\int_eval:n { \g__tbl_col_int - 1 - \g__tbl_table_cols_tl
50          \clist_if_in:NoT \l__tbl_header_columns_clist { \l__tbl_tmpa_tl }
51            {
52              \tl_set:Nn \l__tbl_celltag_tl {TH}
53              \tl_set:Ne \l__tbl_cellattribute_tl {\l__tbl_cellattribute_tl,TH-row}
54            }
55          \tag_struct_begin:n
56            {
57              tag             =\l__tbl_celltag_tl,
58              attribute-class ={\l__tbl_cellattribute_tl}
59            }
60          \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.

```
61          \int_step_inline:nn { \g__tbl_span_tl - 1 }
62            {
63              \seq_gput_right:Ne \g__tbl_struct_cur_seq { -\tag_get:n {struct_num} }
64            }
65          \tag_mc_begin:n{}
66        }
67    }
```

TD (*plug*)

```
68  \NewSocketPlug{tagsupport/tbl/cell/end}{TD}
69    {
70      \__tbl_if_tag_cell:nnT {\g__tbl_row_int } { \g__tbl_col_int }
71        {
72          \tag_mc_end:
73          \tag_struct_end:
74        }
75    }
```

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

TDpbox (*plug*)

```
76  \NewSocketPlug{tagsupport/tbl/pcell/begin}{TDpbox}
77    {
78      \__tbl_show_curr_cell_data:
79      \__tbl_if_tag_cell:nnT {\g__tbl_row_int } { \g__tbl_col_int }
80        {
81          \clist_if_in:NVT \l__tbl_header_columns_clist\g__tbl_col_int
82            {
83              \tl_set:Nn \l__tbl_celltag_tl {TH}
84              \tl_set:Ne \l__tbl_cellattribute_tl {\l__tbl_cellattribute_tl,TH-row}
```

```
85                }
86              \tag_struct_begin:n
87                {
88                  tag              =\l__tbl_pcelltag_tl,
89                  attribute-class =={\l__tbl_cellattribute_tl}
90                }
91              \seq_gput_right:Ne \g__tbl_struct_cur_seq { \tag_get:n {struct_num} }
92              \int_step_inline:nn { \g__tbl_span_tl - 1 }
93                {
94                  \seq_gput_right:Ne \g__tbl_struct_cur_seq { -\tag_get:n {struct_num} }
95                }
96              \tagpdfparaOn
97              \tl_set:Nn \l__tag_para_main_tag_tl {Div}
98          }
99      }
```

TDpbox (*plug*)

```
100  \NewSocketPlug{tagsupport/tbl/pcell/end}{TDpbox}
101    {
102      \__tbl_if_tag_cell:nnT {\g__tbl_row_int } { \g__tbl_col_int }
103        {
104          \tag_struct_end:
105          \legacy_if:nT {@endpe}{\par}
106          \mode_if_vertical:T{ \tagpdfparaOff }
107        }
108    }
```

TR (*plug*)

```
109  \NewSocketPlug{tagsupport/tbl/row/begin}{TR}
110    {
111      \seq_gclear:N \g__tbl_struct_cur_seq
112      \tag_struct_begin:n
113        {
114          tag              =\l__tbl_rowtag_tl,
115          attribute-class=\l__tbl_rowattribute_tl
116        }
117      \seq_gput_right:Ne \g__tbl_struct_rows_seq { \tag_get:n {struct_num} }
118    }
```

TR (*plug*)

```
119  \NewSocketPlug{tagsupport/tbl/row/end}{TR}
120    {
121      \tag_if_active:T
122        {
123      \__tbl_add_missing_cells:
124      \seq_gput_right:Ne \g__tbl_struct_cells_seq
125        {
126        \seq_use:Nn \g__tbl_struct_cur_seq {,}
127        }
128      \int_compare:nNnTF { \g__tbl_row_int } =
129                        { \seq_count:N \g__tbl_struct_cells_seq }
130        {
131        \__tbl_trace:n
132          {==>~
```

```
133          structure~stored~for~row~\int_use:N\g__tbl_row_int :~
134          \seq_use:Nn \g__tbl_struct_cur_seq {,}
135        }
136      }
137      { \ERRORtbl/row } % should not happen ...
138    \tag_struct_end:
139  }
140  }
```

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.

\l__tag_block_flattened_level_int Count the levels of nested blockenvs starting with the first that is "flattened". The counter is defined in lttagging.dtx, but until the next release 11/24 we set it up here too so that we can use it in the following socket

```
141 \int_if_exist:NF \l__tag_block_flattened_level_int
142   {
143     \int_new:N \l__tag_block_flattened_level_int
144   }
```

(*End of definition for* `\l__tag_block_flattened_level_int`.)

Table (*plug*) Inside a TD or TR Part is not allowed as child. For structures that restore paragraph settings we therefore need a special plug that adjust the settings. Currently we set the para main tag to Div. This leads to double Div-structures but flattening the paragraph doesn't work, it errors if there is a list inside.

```
145 \NewSocketPlug{tagsupport/para/restore}{Table}
146   {
147     \tl_set:Nn    \l__tag_para_main_tag_tl {Div}
148     \tl_set_eq:NN \l__tag_para_tag_tl\l__tag_para_tag_default_tl
149     \bool_set_true:N \l__tag_para_bool
150     \bool_set_false:N \l__tag_para_flattened_bool
151     \int_zero:N \l__tag_block_flattened_level_int
152   }
```

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

```
153 \NewSocketPlug{tagsupport/tbl/init}{Table}
154   {
155     \bool_set_false:N \l__tag_para_bool
156     \AssignSocketPlug{tagsupport/para/restore}{Table}
```

We also initialize the structure data variables a this point.

```
157     \__tbl_init_struct_data:
158   }
```

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

```
159 \NewSocketPlug{tagsupport/tbl/finalize}{Table}
160   {
161     \__tbl_set_header_rows:
```

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

```
162     \__tbl_restore_struct_data:
163   }
```

9

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

```
164  \NewSocketPlug{tagsupport/tbl/longtable/init}{Table}
165  {
166    \seq_gclear:N\g__tbl_struct_rows_seq
167    \seq_gclear:N\g__tbl_struct_cells_seq
168    \seq_gclear:N\g__tbl_struct_cur_seq
169    \seq_gclear:N\g__tbl_LT@firsthead_rows_seq
170    \seq_gclear:N\g__tbl_LT@head_rows_seq
171    \seq_gclear:N\g__tbl_LT@lastfoot_rows_seq
172    \seq_gclear:N\g__tbl_LT@foot_rows_seq
173  }
```

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

```
174  \NewSocketPlug{tagsupport/tbl/longtable/finalize}{Table}
175    {
```

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

```
176      \bool_lazy_and:nnTF
177        { \seq_if_empty_p:N \g__tbl_LT@head_rows_seq  }
178        { \seq_if_empty_p:N \g__tbl_LT@firsthead_rows_seq }
179        { \__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.

```
180        {
181          \seq_if_empty:NTF \g__tbl_LT@firsthead_rows_seq
182            {
183              \clist_set:Ne \l__tbl_header_rows_clist
184                {\seq_use:Nn \g__tbl_LT@head_rows_seq {,}}
185              \__tbl_set_header_rows:
186            }
```

In the other case we use firsthead.

```
187            {
188              \clist_set:Ne \l__tbl_header_rows_clist
189                { \seq_use:Nn \g__tbl_LT@firsthead_rows_seq {,} }
190              \__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.

```
191              \pdf_version_compare:NnTF < {2.0}
192                {
193                  \seq_map_inline:Nn \g__tbl_LT@head_rows_seq
194                    {
195                      \seq_gset_item:cnn
196                        {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
197                        { ##1 }
198                        {}
```

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.

```
199          \tl_set:Ne \l__tbl_tmpa_tl
200            { \seq_item:Nn\g__tbl_struct_rows_seq {##1} }
201          \prop_if_exist:cT
202            { g__tag_struct_ \l__tbl_tmpa_tl _prop }
203            {
204              %\prop_gremove:cn {g__tag_struct_ \l__tbl_tmpa_tl _prop} {P}
205            }
206        }
207      }
208      {
209        \seq_map_inline:Nn \g__tbl_LT@head_rows_seq
210          {
211            \tl_set:Ne \l__tbl_tmpa_tl
212              { \seq_item:Nn\g__tbl_struct_rows_seq {##1} }
213            \prop_if_exist:cT
214              { g__tag_struct_ \l__tbl_tmpa_tl _prop }
215              {
216                \__tag_struct_prop_gput:onn { \l__tbl_tmpa_tl } {S}{/Artifact}
217              }
218          }
219      }
220    }
221  }
```

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.

```
222      \bool_lazy_and:nnF
223        { \seq_if_empty_p:N \g__tbl_LT@foot_rows_seq  }
224        { \seq_if_empty_p:N \g__tbl_LT@lastfoot_rows_seq }
225        {
```

If lastfoot is empty move foot to the end.

```
226          \seq_if_empty:NTF \g__tbl_LT@lastfoot_rows_seq
227            {
228              \seq_map_inline:Nn \g__tbl_LT@foot_rows_seq
229                {
230                  \tl_set:Ne \l__tbl_tmpa_tl
231                    {
232                      \seq_item:cn
233                        {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
234                        {##1}
235                    }
236                  \seq_gset_item:cnn
237                    {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
238                    { ##1 }
239                    {}
240                  \seq_gput_right:cV
241                    {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
242                    \l__tbl_tmpa_tl
243                }
244            }
```

If lastfoot is not empty we move that.

```
245            {
246              \seq_map_inline:Nn \g__tbl_LT@lastfoot_rows_seq
```

```
247                       {
248                         \tl_set:Ne \l__tbl_tmpa_tl
249                           {
250                             \seq_item:cn
251                               {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
252                               {##1}
253                           }
254                         \seq_gset_item:cnn
255                           {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
256                           { ##1 }
257                           {}
258                         \seq_gput_right:cV
259                           {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
260                           \l__tbl_tmpa_tl
261                       }
```

and we hide foot

```
262                       \pdf_version_compare:NnTF < {2.0}
263                         {
264                           \seq_map_inline:Nn \g__tbl_LT@foot_rows_seq
265                           {
266                             \seq_gset_item:cnn
267                               {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
268                               { ##1 }
269                               {}
```

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.

```
270                             \tl_set:Ne \l__tbl_tmpa_tl
271                               { \seq_item:Nn\g__tbl_struct_rows_seq {##1} }
272                             \prop_if_exist:cT
273                               { g__tag_struct_ \l__tbl_tmpa_tl _prop }
274                               {
275                                 %\prop_gremove:cn {g__tag_struct_ \l__tbl_tmpa_tl _prop} {P}
276                               }
277                           }
278                         }
279                         {
280                           \seq_map_inline:Nn \g__tbl_LT@foot_rows_seq
281                           {
282                             \tl_set:Ne \l__tbl_tmpa_tl
283                               { \seq_item:Nn\g__tbl_struct_rows_seq {##1} }
284                             \prop_if_exist:cT
285                               { g__tag_struct_ \l__tbl_tmpa_tl _prop }
286                               {
287                                 \__tag_struct_prop_gput:onn {\l__tbl_tmpa_tl} {S}{/Artifact}
288                               }
289                           }
290                         }
291                       }
292                   }
293               }
```

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

```
294 \NewSocketPlug{tagsupport/tbl/longtable/head}{Table}
295 {
296   \tagmcbegin{artifact}
297     \tag_mc_reset_box:N\LT@head
298   \tagmcend
299 }
```

Table (*plug*)

```
300 \NewSocketPlug{tagsupport/tbl/longtable/foot}{Table}
301 {
302     \tagmcbegin{artifact}
303     \tag_mc_reset_box:N \LT@foot
304     \tagmcend
305 }
```

Table (*plug*)

```
306 \NewSocketPlug{tagsupport/tbl/hmode/begin}{Table}
307   {
308       \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.

```
309       \bool_lazy_and:nnT
310         { \bool_if_exist_p:N \l__tag_para_bool } { \l__tag_para_bool }
311         { \tag_struct_end:n { text } }
312       \tag_struct_begin:n {tag=\l__tbl_tabletag_tl}
313       \tl_gset:Ne \g__tbl_struct_table_tl { \tag_get:n {struct_num} }
314   }
```

Table (*plug*)

```
315 \NewSocketPlug{tagsupport/tbl/hmode/end}{Table}
316   {
317       \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.

```
318       \bool_lazy_and:nnT
319         { \bool_if_exist_p:N \l__tag_para_bool } { \l__tag_para_bool }
320         { \tag_struct_begin:n { tag=\l__tag_para_tag_tl } }
321       \tag_mc_begin_pop:n{}
322   }
```

Table (*plug*)

```
323 \NewSocketPlug{tagsupport/tbl/vmode/begin}{Table}
324   {
325     \tag_struct_begin:n {tag=\l__tbl_tabletag_tl}
326     \tl_gset:Ne \g__tbl_struct_table_tl { \tag_get:n {struct_num} }
327   }
```

Table (*plug*)

```
328 \NewSocketPlug{tagsupport/tbl/vmode/end}{Table}
329   {
330     \tag_struct_end:
331     \par
332   }
```

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`.

```
333 \NewSocketPlug{tagsupport/tbl/colspan}{code}
334   {
335     \int_compare:nNnT {#1}>{1}
336       {
337         \prop_get:NeNF \g__tag_attr_entries_prop
338           {colspan-\int_eval:n{#1}}
339           \l__tbl_tmpa_tl
340           {
341             \__tag_attr_new_entry:ee
342               {colspan-\int_eval:n{#1}}
343               {/O /Table /ColSpan~\int_eval:n{#1}}
344           }
345         \tl_set:Ne \l__tbl_cellattribute_tl
346           {\l__tbl_cellattribute_tl,colspan-\int_eval:n{#1}}
347       }
348   }
```

code (*plug*) The sockets will be in lttagging.dtx, but that may only happen in the next main release, so for now we test if they are in the format and if not define them now.

```
349 \str_if_exist:cF { l__socket_tagsupport/tbl/leaders/end_plug_str }
350   {
351     \NewSocket{tagsupport/tbl/leaders/begin}{0}
352     \NewSocket{tagsupport/tbl/leaders/end}{0}
353   }
354 \NewSocketPlug{tagsupport/tbl/leaders/begin}{code}
355   { \tag_mc_begin:n{artifact} }
356 \NewSocketPlug{tagsupport/tbl/leaders/end}{code}
357   { \tag_mc_end: }
```

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.

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

14

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:

```
359 \cs_new_protected:Npn \__tag_tbl_disable:
360  {
361    \AssignSocketPlug{tagsupport/tbl/cell/begin}{noop}
362    \AssignSocketPlug{tagsupport/tbl/cell/end}{noop}
363    \AssignSocketPlug{tagsupport/tbl/pcell/begin}{noop}
364    \AssignSocketPlug{tagsupport/tbl/pcell/end}{noop}
365    \AssignSocketPlug{tagsupport/tbl/row/begin}{noop}
366    \AssignSocketPlug{tagsupport/tbl/row/end}{noop}
367    \AssignSocketPlug{tagsupport/tbl/init}{noop}
368    \AssignSocketPlug{tagsupport/tbl/finalize}{noop}
369    \AssignSocketPlug{tagsupport/tbl/longtable/init}{noop}
370    \AssignSocketPlug{tagsupport/tbl/longtable/head}{noop}
371    \AssignSocketPlug{tagsupport/tbl/longtable/foot}{noop}
372    \AssignSocketPlug{tagsupport/tbl/longtable/finalize}{noop}
373    \AssignSocketPlug{tagsupport/tbl/hmode/begin}{noop}
374    \AssignSocketPlug{tagsupport/tbl/hmode/end}{noop}
375    \AssignSocketPlug{tagsupport/tbl/vmode/begin}{noop}
376    \AssignSocketPlug{tagsupport/tbl/vmode/end}{noop}
377    \AssignSocketPlug{tagsupport/tbl/colspan}{noop}
378    \AssignSocketPlug{tagsupport/tbl/leaders/begin}{noop}
379    \AssignSocketPlug{tagsupport/tbl/leaders/end}{noop}
380  }
```

(*End of definition for* __tag_tbl_disable:.)

__tag_tbl_enable:

```
381 \cs_new_protected:Npn \__tag_tbl_enable:
382  {
383    \AssignSocketPlug{tagsupport/tbl/cell/begin}{TD}
384    \AssignSocketPlug{tagsupport/tbl/cell/end}{TD}
385    \AssignSocketPlug{tagsupport/tbl/pcell/begin}{TDpbox}
386    \AssignSocketPlug{tagsupport/tbl/pcell/end}{TDpbox}
387    \AssignSocketPlug{tagsupport/tbl/row/begin}{TR}
388    \AssignSocketPlug{tagsupport/tbl/row/end}{TR}
389    \AssignSocketPlug{tagsupport/tbl/init}{Table}
390    \AssignSocketPlug{tagsupport/tbl/finalize}{Table}
391    \AssignSocketPlug{tagsupport/tbl/longtable/head}{Table}
392    \AssignSocketPlug{tagsupport/tbl/longtable/foot}{Table}
393    \AssignSocketPlug{tagsupport/tbl/longtable/init}{Table}
394    \AssignSocketPlug{tagsupport/tbl/longtable/finalize}{Table}
395    \AssignSocketPlug{tagsupport/tbl/hmode/begin}{Table}
396    \AssignSocketPlug{tagsupport/tbl/hmode/end}{Table}
397    \AssignSocketPlug{tagsupport/tbl/vmode/begin}{Table}
398    \AssignSocketPlug{tagsupport/tbl/vmode/end}{Table}
399    \AssignSocketPlug{tagsupport/tbl/colspan}{code}
```

```
400     \AssignSocketPlug{tagsupport/tbl/leaders/begin}{code}
401     \AssignSocketPlug{tagsupport/tbl/leaders/end}{code}
402   }
```

(*End of definition for* __tag_tbl_enable:.)

```
403 % See tagpdfsetup-keys.md in tagpdf/doc for the naming scheme.
404 \keys_define:nn { __tag / setup }
405   {
406     table/tagging .choices:nn = { true, on }
407       { \__tag_tbl_enable: },
408     table/tagging .choices:nn = { false, off }
409       { \__tag_tbl_disable: },
410     table/tagging .choice:,
411     table/tagging / presentation .code:n =
412      {
413        \__tag_tbl_enable:
414        \tl_set:Nn\l__tbl_rowtag_tl   {NonStruct}
415        \tl_set:Nn\l__tbl_pcelltag_tl {NonStruct}
416        \tl_set:Nn\l__tbl_celltag_tl  {text}
417        \tl_set:Nn\l__tbl_tabletag_tl {Div}
418        \clist_clear:N \l__tbl_header_rows_clist
419        \clist_clear:N \l__tbl_header_columns_clist
420      },
421     table/tagging .default:n = true,
422     table/tagging .initial:n = true
423   }
```

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

```
424 \keys_define:nn { __tag / setup }
425   {
426     table-tagging .meta:n = {table/tagging={#1}}
427   }
```

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 `IDs` 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.

```
428 \AddToHook{begindocument}
429   {
430     \tag_if_active:T
431       {
432         \tagpdfsetup
```

```
433              {
434               role/new-attribute =
435                {TH-col}{/O /Table /Scope /Column},
436               role/new-attribute =
437                {TH-row}{/O /Table /Scope /Row},
438               role/new-attribute =
439                {TH-both}{/O /Table /Scope /Both},
440              }
```

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

```
441               \seq_gput_left:Ne\g__tag_attr_class_used_seq
442                 {\pdf_name_from_unicode_e:n{TH-col}}
443               \seq_gput_left:Ne\g__tag_attr_class_used_seq
444                 {\pdf_name_from_unicode_e:n{TH-row}}
445               \seq_gput_left:Ne\g__tag_attr_class_used_seq
446                 {\pdf_name_from_unicode_e:n{TH-both}}
447          }
448       }
449
```

`\l__tbl_header_rows_clist`
`\l__tbl_header_columns_clist`

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

```
450 \clist_new:N  \l__tbl_header_rows_clist
451 \clist_new:N  \l__tbl_header_columns_clist
```

`\__tbl_set_header_rows:`

```
452 \cs_new_protected:Npn \__tbl_set_header_rows:
453   {
454     \clist_map_inline:Nn \l__tbl_header_rows_clist
455       {
456         \clist_set:Ne\l__tbl_tmpa_clist
457           {
458             \seq_item:Nn \g__tbl_struct_cells_seq {##1}
459           }
460         \clist_map_inline:Nn \l__tbl_tmpa_clist
461           {
```

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

```
462             \prop_if_exist:cT { g__tag_struct_####1_prop }
463               {
464                 \__tag_struct_prop_gput:nnn{ ####1 }{S}{/TH}
```

This sets the scope class. If the header has already a row attribute we replace by TH-both.

```
465             \prop_get:cnNTF
466               { g__tag_struct_####1_prop }
467               { C }
468               \l__tbl_tmpa_tl
469               {
470                 \str_set:Ne \l__tbl_tmpa_str {\l__tbl_tmpa_tl}
471                 \str_remove_once:Nn \l__tbl_tmpa_str {[}
```

17

```
472                        \str_remove_once:Nn \l__tbl_tmpa_str {]}
473                        \str_if_in:NnTF\l__tbl_tmpa_str{/TH-row}
474                         {
475                           \str_replace_once:Nnn \l__tbl_tmpa_str {/TH-row}{/TH-both}
476                           \__tag_struct_prop_gput:nne{ ####1 }{C}{[\l__tbl_tmpa_str]}
477                         }
478                         {
479                           \__tag_struct_prop_gput:nne{ ####1 }{C}{[/TH-col~\l__tbl_tmpa_str]}
480                         }
481                      }
482                      {\__tag_struct_prop_gput:nnn{ ####1 }{C}{/TH-col}}
483                 }
484             }
485         }
486     }
```

(*End of definition for* __tbl_set_header_rows:.)

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

```
487 \keys_define:nn { __tag / setup }
488   {
489     table/header-rows .clist_set:N = \l__tbl_header_rows_clist,
490     table/header-columns .clist_set:N = \l__tbl_header_columns_clist,
```

obsolete older name:

```
491     table-header-rows .meta:n = {table/header-rows={#1}}
492   }
```

5.5 Multirow support

__tbl_multirow:n This command makes the current cell into a multirow cell: it creates, if needed, an RowSpan-attribute, adds it to the attributes of the cell structure, and marks all following cells spanned by the multirow as cells that should not be tagged. The argument is the number of spanned row (including the current row).

```
493 \cs_if_exist:NT \__tag_struct_prop_gput:nnn
494 {
495   \cs_generate_variant:Nn \__tag_struct_prop_gput:nnn {one}
496 }
497 \cs_new_protected:Npn \__tbl_multirow:n #1
498 {
```

Create an attribute if needed:

```
499     \prop_get:NeNF \g__tag_attr_entries_prop
500       {rowspan-\int_eval:n{#1}}
501       \l__tbl_tmpa_tl
502       {
503         \__tag_attr_new_entry:ee
504           {rowspan-\int_eval:n{#1}}
505           {/O /Table /RowSpan~\int_eval:n{#1}}
506       }
```

ensure that the attribute is marked as used:

```
507     \seq_gput_left:Ne\g__tag_attr_class_used_seq
508       {\pdf_name_from_unicode_e:n{rowspan-\int_eval:n{#1}}}
```

Get the structure number of the current cell

```
509     \seq_get_right:NN\g__tbl_struct_cur_seq \l__tbl_tmpb_tl
```

If we are in a multicolumn the number can be negative and this must be changed

```
510     \tl_set:Ne \l__tbl_tmpb_tl { \int_abs:n{\l__tbl_tmpb_tl} }
```

no we must update an existing attribute. TODO: simplify this ... (see also colspan handling).

```
511     \prop_get:cnNTF
512        { g__tag_struct_\l__tbl_tmpb_tl _prop }
513        { C }
514        \l__tbl_tmpa_tl
515        {
516          \tl_remove_once:Nn \l__tbl_tmpa_tl {[}
517          \tl_remove_once:Nn \l__tbl_tmpa_tl {]}
518          \__tag_struct_prop_gput:one{ \l__tbl_tmpb_tl }
519            {C}
520            {[/rowspan-\int_eval:n{#1}~\l__tbl_tmpa_tl]}
521        }
522        {
523          \__tag_struct_prop_gput:one{ \l__tbl_tmpb_tl }
524            {C}
525            {[/rowspan-\int_eval:n{#1}]}
526        }
```

Now mark the spanned cells that should be ignored.

```
527       \__tbl_gset_untagged_row_cells:nn {#1-1}{\g__tbl_span_tl}
528   }
```

(*End of definition for* __tbl_multirow:n.)

```
529 \keys_define:nn{ __tag / setup }
530   { table/multirow .code:n = {\__tbl_multirow:n {#1} }
531    ,table/multirow .default:n = 1
532   }
```

__tbl_gset_untagged_row_cells:nn   This command stores the row and column numbers of the cells in the following row(s) that should not be tagged as they are a part of a rowspan.

```
533 \cs_new_protected:Npn \__tbl_gset_untagged_row_cells:nn #1 #2 % #1 number of rows, #2 number
534   {
535     \int_step_inline:nn {#1}
536       {
537         \int_step_inline:nn {#2}
538           {
539             \prop_gput:Nee \g__tbl_untagged_cells_prop
540               { \int_eval:n {\g__tbl_row_int + ##1},\int_eval:n{\g__tbl_col_int + ####1 -1 } }{}
541           }
542       }
543   }
```

(*End of definition for* __tbl_gset_untagged_row_cells:nn.)

__tbl_if_tag_cell:nn   We must be able to detect if a cell should be tagged. For this we define a conditional — if more options are needed it can be extended.

```
544 \prg_new_protected_conditional:Npnn\__tbl_if_tag_cell:nn #1 #2 %#1 row, #2 col
545   { T,TF }
```

```
546   {
547     \prop_get:NeNTF \g__tbl_untagged_cells_prop
548       {\int_eval:n{#1},\int_eval:n{#2}}\l__tbl_tmpa_tl
549       { \prg_return_false:}
550       { \prg_return_true: }
551   }
```

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

5.6   Misc stuff

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

```
552 \cs_new_protected:Npn \__tbl_show_curr_cell_data: {
553   \__tbl_trace:n { ==>~ current~cell~data:~
554             \int_use:N \g__tbl_row_int ,
555             \int_use:N \g__tbl_col_int ,
556             \g__tbl_span_tl
557           }
558 }
```

(*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.

```
559 \cs_new:Npn \__tbl_add_missing_cells:
560   {
```

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:

```
561     \int_compare:nNnT \g__tbl_missing_cells_int > 0
562       {
563       \__tbl_trace:n {==>~
564         ~Inserting~\int_use:N \g__tbl_missing_cells_int \space
565         additional~cell(s)~into~previous~row:}
566       \int_step_inline:nn { \g__tbl_missing_cells_int }
567         {
568           \int_gincr:N\g__tbl_col_int
569           \UseTaggingSocket{tbl/cell/begin}
570           \UseTaggingSocket{tbl/cell/end}
571         }
572     }
573   }
```

(*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.

```
574 \tl_new:N  \g__tbl_struct_table_tl
575 \tl_new:N  \l__tbl_saved_struct_table_tl
576 \seq_new:N \g__tbl_struct_rows_seq
577 \seq_new:N \l__tbl_saved_struct_rows_seq
578 \seq_new:N \g__tbl_struct_cells_seq
579 \seq_new:N \l__tbl_saved_struct_cells_seq
580 \seq_new:N \g__tbl_struct_cur_seq
581 \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.

```
582 \cs_new_protected:Npn \__tbl_init_struct_data: {
583   \tl_set_eq:NN  \l__tbl_saved_struct_table_tl \g__tbl_struct_table_tl
584   \seq_set_eq:NN \l__tbl_saved_struct_rows_seq \g__tbl_struct_rows_seq
585   \seq_set_eq:NN \l__tbl_saved_struct_cells_seq \g__tbl_struct_cells_seq
586   \seq_set_eq:NN \l__tbl_saved_struct_cur_seq \g__tbl_struct_cur_seq
587 %
588   \seq_gclear:N\g__tbl_struct_rows_seq
589   \seq_gclear:N\g__tbl_struct_cells_seq
590   \seq_gclear:N\g__tbl_struct_cur_seq
591 }
```

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

__tbl_restore_struct_data:

```
592 \cs_new_protected:Npn \__tbl_restore_struct_data: {
593   \tl_gset_eq:NN  \g__tbl_struct_table_tl \l__tbl_saved_struct_table_tl
594   \seq_gset_eq:NN \g__tbl_struct_rows_seq \l__tbl_saved_struct_rows_seq
595   \seq_gset_eq:NN \g__tbl_struct_cells_seq\l__tbl_saved_struct_cells_seq
596   \seq_gset_eq:NN \g__tbl_struct_cur_seq  \l__tbl_saved_struct_cur_seq
597 }
```

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

5.7 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.

```
598 \def\__tbl_patch_LT@makecaption#1#2#3{%
599   \LT@mcol\LT@cols c{%
600   % test can go after merge
601     \str_if_exist:cT { l__socket_tagsupport/parbox/before_plug_str }
602       {
603         \AssignSocketPlug{tagsupport/parbox/before}{noop}
604         \AssignSocketPlug{tagsupport/parbox/after}{noop}
605       }
606     \hbox to\z@{\hss\parbox[t]\LTcapwidth{%
607     \reset@font
608     \tag_suspend:n{caption}
609     \sbox\@tempboxa{#1{#2:~}#3}%
```

```
610    \tag_resume:n{caption}
611    \ifdim\wd\@tempboxa>\hsize
612      #1{#2:~}#3%
613    \else
614      \hbox to\hsize{\hfil#1{#2:~}#3\hfil}%
615    \fi
616    \endgraf\vskip\baselineskip}%
617  \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.

```
618 \AddToHook{package/longtable/after}
619   {
620    \cs_set_eq:NN \LT@makecaption\__tbl_patch_LT@makecaption
621   }
622 ⟨/package⟩

623 ⟨*latex-lab⟩
624 \ProvidesFile{table-latex-lab-testphase.ltx}
625        [\ltlabtbldate\space v\ltlabtblversion\space latex-lab wrapper table]
626 \RequirePackage{latex-lab-testphase-table}
627 ⟨/latex-lab⟩

628 ⟨*latex-lab-alias⟩
629 \ProvidesFile{tabular-latex-lab-testphase.ltx}
630        [\ltlabtbldate\space v\ltlabtblversion\space latex-lab wrapper tabular]
631 \RequirePackage{latex-lab-testphase-table}
632 ⟨/latex-lab-alias⟩
```