

The `latex-lab-table` package

Changes related to the tagging of tables

Frank & Ulrike, L^AT_EX Project*

v0.85k 2024-02-07

Abstract

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

Contents

1	Documentation	2
2	Limitations	3
3	Introduction	4
4	Technical details and problems	4
	4.1 TODOs	4
5	Implementation	5
	5.1 Variables	5
	5.2 Tagging support sockets	5
	5.3 Environments	12
	5.4 Interfaces to tagging	12
	5.4.1 Tagging helper commands	12
	5.4.2 Disabling/enabling	12
	5.4.3 Header support	14
	5.5 Misc stuff	16
	5.6 <code>longtable</code>	17

*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 table¹ 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`.

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.

There is some basic support³ for headers. With⁴

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

²The key has been renamed. The old name ‘table-tagging’ still works but is deprecated. The value `presentation` refers to the ARIA role “presentation”.

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

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

`\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).

⁵While 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.

⁶Helpful for us at this stage.

5 Implementation

```
1 <@@=tbl>
2 <*package>
3 \ProvidesExplPackage {latex-lab-testphase-table} {\tlabtbldate} {\tlabtblversion}
4 {Code related to the tagging of tables}
```

This builds on array so we load it by default:

```
5 \RequirePackage{array}
```

5.1 Variables

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 {}
```

Temp variables

```
18 \clist_new:N \l__tbl_tmpa_clist
19 \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*)

```
20 \NewSocketPlug{tagsupport/tbl/cell/begin}{TD}
21 {
```

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.

```
22 \__tbl_show_curr_cell_data:
23 \tag_struct_begin:n
24 {
25     tag =\l__tbl_celltag_tl,
26     attribute-class =\l__tbl_cellattribute_tl
27 }
28 \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.

```

29   \int_step_inline:mn { \g__tbl_span_tl - 1 }
30   {
31     \seq_gput_right:Ne \g__tbl_struct_cur_seq { -\tag_get:n {struct_num} }
32   }
33   \tag_mc_begin:n{
34 }

```

TD (*plug*)

```

35 \NewSocketPlug{tagsupport/tbl/cell/end}{TD}
36 {
37   \tag_mc_end:
38   \tag_struct_end:
39 }

```

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

TDpbox (*plug*)

```

40 \NewSocketPlug{tagsupport/tbl/pcell/begin}{TDpbox}
41 {
42   \__tbl_show_curr_cell_data:
43   \tag_struct_begin:n
44   {
45     tag           =\l__tbl_pcelltag_tl,
46     attribute-class =\l__tbl_cellattribute_tl
47   }
48   \seq_gput_right:Ne \g__tbl_struct_cur_seq { \tag_get:n {struct_num} }
49   \int_step_inline:mn { \g__tbl_span_tl - 1 }
50   {
51     \seq_gput_right:Ne \g__tbl_struct_cur_seq { -\tag_get:n {struct_num} }
52   }
53   \tagpdfparaOn
54   \tl_set:Nn \l__tag_para_main_tag_tl {Div}
55 }

```

TDpbox (*plug*)

```

56 \NewSocketPlug{tagsupport/tbl/pcell/end}{TDpbox}
57 {
58   \tag_struct_end:
59 }

```

TR (*plug*)

```

60 \NewSocketPlug{tagsupport/tbl/row/begin}{TR}
61 {
62   \seq_gclear:N \g__tbl_struct_cur_seq
63   \tag_struct_begin:n
64   {
65     tag           =\l__tbl_rowtag_tl,
66     attribute-class =\l__tbl_rowattribute_tl
67   }
68   \seq_gput_right:Ne \g__tbl_struct_rows_seq { \tag_get:n {struct_num} }
69 }

```

TR (*plug*)

```
70 \NewSocketPlug{tag-support/tbl/row/end}{TR}
71 {
72   \tag_if_active:T
73   {
74     \__tbl_add_missing_cells:
75     \seq_gput_right:Ne \g__tbl_struct_cells_seq
76     {
77       \seq_use:Nn \g__tbl_struct_cur_seq {,}
78     }
79     \int_compare:nNnTF { \g__tbl_row_int } =
80       { \seq_count:N \g__tbl_struct_cells_seq }
81     {
82       \__tbl_trace:n
83       {==>~
84         structure-stored-for-row~\int_use:N\g__tbl_row_int :-
85         \seq_use:Nn \g__tbl_struct_cur_seq {,}
86       }
87     }
88     { \ERRORtbl/row } % should not happen ...
89     \tag_struct_end:
90   }
91 }
```

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 reenabale paratagging.

```
92 \NewSocketPlug{tag-support/tbl/init}{Table}
93 {
94   \tag_if_active:T
95   {
96     \bool_set_false:N \l__tag_para_bool
97     \__tbl_init_struct_data:
98   }
99 }
```

We also initialize the structure data variables at this point.

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

```
100 \NewSocketPlug{tag-support/tbl/finalize}{Table}
101 {
102   \__tbl_set_header_rows:
103   \__tbl_restore_struct_data:
104 }
```

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

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

```
105 \NewSocketPlug{tag-support/tbl/longtable/init}{Table}
106 {
107   \seq_gclear:N\g__tbl_struct_rows_seq
108   \seq_gclear:N\g__tbl_struct_cells_seq
```

```

109 \seq_gclear:N\g__tbl_struct_cur_seq
110 \seq_gclear:N\g__tbl_LT@firsthead_rows_seq
111 \seq_gclear:N\g__tbl_LT@head_rows_seq
112 \seq_gclear:N\g__tbl_LT@lastfoot_rows_seq
113 \seq_gclear:N\g__tbl_LT@foot_rows_seq
114 }

```

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

```

115 \NewSocketPlug{tagssupport/tbl/longtable/finalize}{Table}
116 {

```

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

```

117 \bool_lazy_and:nnTF
118 { \seq_if_empty_p:N \g__tbl_LT@head_rows_seq }
119 { \seq_if_empty_p:N \g__tbl_LT@firsthead_rows_seq }
120 { \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.

```

121 {
122 \seq_if_empty:NTF \g__tbl_LT@firsthead_rows_seq
123 {
124 \clist_set:Nc \l__tbl_header_rows_clist
125 { \seq_use:Nn \g__tbl_LT@head_rows_seq {,} }
126 \tbl_set_header_rows:
127 }

```

In the other case we use `firsthead`.

```

128 {
129 \clist_set:Nc \l__tbl_header_rows_clist
130 { \seq_use:Nn \g__tbl_LT@firsthead_rows_seq {,} }
131 \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.

```

132 \pdf_version_compare:NnTF < {2.0}
133 {
134 \seq_map_inline:Nn \g__tbl_LT@head_rows_seq
135 {
136 \seq_gset_item:cnn
137 {g__tag_struct_kids_ \g__tbl_struct_table_t1_seq}
138 { ##1 }
139 {}

```

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.

```

140 \tbl_set:Nc \l__tbl_tmpa_t1
141 { \seq_item:Nn\g__tbl_struct_rows_seq {##1} }
142 \prop_if_exist:cT
143 { g__tag_struct_ \l__tbl_tmpa_t1_prop }
144 {

```



```

145         %\prop_gremove:cn {g__tag_struct_ \l__tbl_tmpa_tl _prop} {P}
146     }
147 }
148 }
149 {
150     \seq_map_inline:Nn \g__tbl_LT@head_rows_seq
151     {
152         \tl_set:Ne \l__tbl_tmpa_tl
153         { \seq_item:Nn\g__tbl_struct_rows_seq {##1} }
154         \prop_if_exist:cT
155         { g__tag_struct_ \l__tbl_tmpa_tl _prop }
156         {
157             \__tbl_struct_prop_gput:Vnn \l__tbl_tmpa_tl {S}{/Artifact}
158         }
159     }
160 }
161 }
162 }

```

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.

```

163     \bool_lazy_and:nnF
164     { \seq_if_empty_p:N \g__tbl_LT@foot_rows_seq }
165     { \seq_if_empty_p:N \g__tbl_LT@lastfoot_rows_seq }
166     {

```

If lastfoot is empty move foot to the end.

```

167     \seq_if_empty:NTF \g__tbl_LT@lastfoot_rows_seq
168     {
169         \seq_map_inline:Nn \g__tbl_LT@foot_rows_seq
170         {
171             \tl_set:Ne \l__tbl_tmpa_tl
172             {
173                 \seq_item:cn
174                 {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
175                 {##1}
176             }
177             \seq_gset_item:cnn
178             {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
179             { ##1 }
180             {}
181             \seq_gput_right:cV
182             {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
183             \l__tbl_tmpa_tl
184         }
185     }

```

If lastfoot is not empty we move that.

```

186     {
187         \seq_map_inline:Nn \g__tbl_LT@lastfoot_rows_seq
188         {
189             \tl_set:Ne \l__tbl_tmpa_tl
190             {
191                 \seq_item:cn
192                 {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}

```

```

193         {##1}
194     }
195     \seq_gset_item:cnn
196     {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
197     { ##1 }
198     {}
199     \seq_gput_right:cV
200     {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
201     \l__tbl_tmpa_tl
202 }

```

and we hide foot

```

203     \pdf_version_compare:NnTF < {2.0}
204     {
205         \seq_map_inline:Nn \g__tbl_LT@foot_rows_seq
206         {
207             \seq_gset_item:cnn
208             {g__tag_struct_kids_ \g__tbl_struct_table_tl _seq}
209             { ##1 }
210             {}

```

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.

```

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             %\prop_gremove:cn {g__tag_struct_ \l__tbl_tmpa_tl _prop} {P}
217         }
218     }
219 }
220 {
221     \seq_map_inline:Nn \g__tbl_LT@foot_rows_seq
222     {
223         \tl_set:Ne \l__tbl_tmpa_tl
224         { \seq_item:Nn\g__tbl_struct_rows_seq {##1} }
225         \prop_if_exist:cT
226         { g__tag_struct_ \l__tbl_tmpa_tl _prop }
227         {
228             \__tbl_struct_prop_gput:Vnn \l__tbl_tmpa_tl {S}{/Artifact}
229         }
230     }
231 }
232 }
233 }
234 }

```

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

```

235 \NewSocketPlug{tagsupport/tbl/longtable/head}{Table}
236 {
237     \tagmcbegin{artifact}

```

```

238   \tag_mc_reset_box:N\LT@head
239   \tagmccend
240 }

```

Table (*plug*)

```

241 \NewSocketPlug{tagsupport/tbl/longtable/foot}{Table}
242 {
243   \tagmcbegin{artifact}
244   \tag_mc_reset_box:N \LT@foot
245   \tagmccend
246 }

```

Table (*plug*)

```

247 \NewSocketPlug{tagsupport/tbl/hmode/begin}{Table}
248 {
249   \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.

```

250   \bool_lazy_and:nnT
251     { \bool_if_exist_p:N \l__tag_para_bool } { \l__tag_para_bool }
252     { \tag_struct_end:n { text } }
253   \tag_struct_begin:n {tag=\l__tbl_tabletag_tl}
254   \tl_gset:Ne \g__tbl_struct_table_tl { \tag_get:n {struct_num} }
255 }

```

Table (*plug*)

```

256 \NewSocketPlug{tagsupport/tbl/hmode/end}{Table}
257 {
258   \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.

```

259   \bool_lazy_and:nnT
260     { \bool_if_exist_p:N \l__tag_para_bool } { \l__tag_para_bool }
261     { \tag_struct_begin:n { tag=\l__tag_para_tag_tl } }
262   \tag_mc_begin_pop:n{}
263 }

```

Table (*plug*)

```

264 \NewSocketPlug{tagsupport/tbl/vmode/begin}{Table}
265 {
266   \tag_struct_begin:n {tag=\l__tbl_tabletag_tl}
267   \tl_gset:Ne \g__tbl_struct_table_tl { \tag_get:n {struct_num} }
268 }

```

Table (*plug*)

```

269 \NewSocketPlug{tagsupport/tbl/vmode/end}{Table}
270 {
271   \tag_struct_end:
272   \par
273 }

```

`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`.

```

274 \NewSocketPlug{taggsupport/tbl/colspan}{code}
275 {
276   \int_compare:nNnT {#1}>{1}
277   {
278     \prop_get:NeNF \g__tag_attr_entries_prop
279       {colspan-\int_eval:n{#1}}
280     \l__tbl_tmpa_tl
281     {
282       \__tag_attr_new_entry:ee
283         {colspan-\int_eval:n{#1}}
284         {/0 /Table /ColSpan~\int_eval:n{#1}}
285     }
286     \tl_set:Ne \l__tbl_cellattribute_tl
287       {colspan-\int_eval:n{#1}}
288   }
289 }

```

TODO: move to tagpdf

```

290 \tag_if_active:T
291 { \cs_generate_variant:Nn \__tag_attr_new_entry:nn {ee} }
292

```

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.

```

293 \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:`

```

294 \cs_new_protected:Npn \__tag_tbl_disable:
295 {
296   \AssignSocketPlug{taggsupport/tbl/cell/begin}{noop}
297   \AssignSocketPlug{taggsupport/tbl/cell/end}{noop}
298   \AssignSocketPlug{taggsupport/tbl/pcell/begin}{noop}
299   \AssignSocketPlug{taggsupport/tbl/pcell/end}{noop}
300   \AssignSocketPlug{taggsupport/tbl/row/begin}{noop}
301   \AssignSocketPlug{taggsupport/tbl/row/end}{noop}

```

```

302 \AssignSocketPlug{tagsupport/tbl/init}{noop}
303 \AssignSocketPlug{tagsupport/tbl/finalize}{noop}
304 \AssignSocketPlug{tagsupport/tbl/longtable/init}{noop}
305 \AssignSocketPlug{tagsupport/tbl/longtable/head}{noop}
306 \AssignSocketPlug{tagsupport/tbl/longtable/foot}{noop}
307 \AssignSocketPlug{tagsupport/tbl/longtable/finalize}{noop}
308 \AssignSocketPlug{tagsupport/tbl/hmode/begin}{noop}
309 \AssignSocketPlug{tagsupport/tbl/hmode/end}{noop}
310 \AssignSocketPlug{tagsupport/tbl/vmode/begin}{noop}
311 \AssignSocketPlug{tagsupport/tbl/vmode/end}{noop}
312 \AssignSocketPlug{tagsupport/tbl/colspan}{noop}
313 }

```

(End of definition for _tag_tbl_disable:.)

_tag_tbl_enable:

```

314 \cs_new_protected:Npn \_tag_tbl_enable:
315 {
316   \AssignSocketPlug{tagsupport/tbl/cell/begin}{TD}
317   \AssignSocketPlug{tagsupport/tbl/cell/end}{TD}
318   \AssignSocketPlug{tagsupport/tbl/pcell/begin}{TDpbox}
319   \AssignSocketPlug{tagsupport/tbl/pcell/end}{TDpbox}
320   \AssignSocketPlug{tagsupport/tbl/row/begin}{TR}
321   \AssignSocketPlug{tagsupport/tbl/row/end}{TR}
322   \AssignSocketPlug{tagsupport/tbl/init}{Table}
323   \AssignSocketPlug{tagsupport/tbl/finalize}{Table}
324   \AssignSocketPlug{tagsupport/tbl/longtable/head}{Table}
325   \AssignSocketPlug{tagsupport/tbl/longtable/foot}{Table}
326   \AssignSocketPlug{tagsupport/tbl/longtable/init}{Table}
327   \AssignSocketPlug{tagsupport/tbl/longtable/finalize}{Table}
328   \AssignSocketPlug{tagsupport/tbl/hmode/begin}{Table}
329   \AssignSocketPlug{tagsupport/tbl/hmode/end}{Table}
330   \AssignSocketPlug{tagsupport/tbl/vmode/begin}{Table}
331   \AssignSocketPlug{tagsupport/tbl/vmode/end}{Table}
332   \AssignSocketPlug{tagsupport/tbl/colspan}{code}
333 }

```

(End of definition for _tag_tbl_enable:.)

```

334 % See tagpdfsetup-keys.md in tagpdf/doc for the naming scheme.
335 \keys_define:nn { \_tag / setup }
336 {
337   table/tagging .choices:nn = { true, on }
338   { \_tag_tbl_enable: },
339   table/tagging .choices:nn = { false, off }
340   { \_tag_tbl_disable: },
341   table/tagging .choice:,
342   table/tagging / presentation .code:n =
343   {
344     \_tag_tbl_enable:
345     \tl_set:Nn\l__tbl_rowtag_tl {NonStruct}
346     \tl_set:Nn\l__tbl_pcelltag_tl {NonStruct}
347     \tl_set:Nn\l__tbl_celltag_tl {text}
348     \tl_set:Nn\l__tbl_tabletag_tl {Div}
349   },

```

```

350     table/tagging .default:n = true,
351     table/tagging .initial:n = true
352 }

```

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

```

353 \keys_define:nn { __tag / setup }
354 {
355     table-tagging .meta:n = {table/tagging={#1}}
356 }

```

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.

```

357 \AddToHook{begindocument}
358 {
359     \tag_if_active:T
360     {
361         \tagpdfsetup
362         {
363             role/new-attribute =
364             {TH-col}{/O /Table /Scope /Column},
365             role/new-attribute =
366             {TH-row}{/O /Table /Scope /Row},
367             role/new-attribute =
368             {TH-both}{/O /Table /Scope /Both},
369         }
370     }

```

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

```

370         \seq_gput_left:Ne\g__tag_attr_class_used_seq
371         {\pdf_name_from_unicode_e:n{TH-col}}
372         \seq_gput_left:Ne\g__tag_attr_class_used_seq
373         {\pdf_name_from_unicode_e:n{TH-row}}
374         \seq_gput_left:Ne\g__tag_attr_class_used_seq
375         {\pdf_name_from_unicode_e:n{TH-both}}
376     }
377 }
378

```

`\l__tbl_header_rows_clist` This holds the numbers of the header rows. Negative numbers are possible and count from the back.

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

`__tbl_set_header_rows:` TEMP: Next tagpdf will have the right command which also updates the debug info. For now a temporary command:

```
380 \cs_if_free:NTF \__tag_struct_prop_gput:nnn
381 {
382   \cs_new_protected:Npn \__tbl_struct_prop_gput:nnn #1#2#3
383   { \prop_gput:cn { g__tag_struct_#1_prop }{#2}{#3} }
384 }
385 { \cs_new_protected:Npn \__tbl_struct_prop_gput:nnn #1#2#3
386   { \__tag_struct_prop_gput:nnn {#1}{#2}{#3} }
387 }
388 \cs_generate_variant:Nn \__tbl_struct_prop_gput:nnn {nne,Vnn}
389 \cs_new_protected:Npn \__tbl_set_header_rows:
390 {
391   \clist_map_inline:Nn \l__tbl_header_rows_clist
392   {
393     \clist_set:Nc\l__tbl_tmpa_clist
394     {
395       \seq_item:Nn \g__tbl_struct_cells_seq {##1}
396     }
397     \clist_map_inline:Nn \l__tbl_tmpa_clist
398     {
```

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

```
399   \prop_if_exist:cT { g__tag_struct_####1_prop }
400   {
401     \__tbl_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.

```
402   \prop_get:cnNTF
403   { g__tag_struct_####1_prop }
404   { C }
405   \l__tbl_tmpa_tl
406   {\__tbl_struct_prop_gput:nne{ ####1 }{C}{[/TH-col-\l__tbl_tmpa_tl]} }
407   {\__tbl_struct_prop_gput:nnn{ ####1 }{C}{/TH-col}}
408   }
409   }
410 }
411 }
```

(End of definition for `__tbl_set_header_rows:`.)

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

```
412 \keys_define:nn { __tag / setup }
413 {
414   table/header-rows .clist_set:N = \l__tbl_header_rows_clist,
obsolete older name:
415   table-header-rows .meta:n = {table/header-rows={#1}}
416 }
```

5.5 Misc stuff

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

```

417 \cs_new_protected:Npn \__tbl_show_curr_cell_data: {
418   \__tbl_trace:n { ==>~ current~cell~data:~
419     \int_use:N \g__tbl_row_int ,
420     \int_use:N \g__tbl_col_int ,
421     \g__tbl_span_tl
422   }
423 }
```

(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.

```

424 \NewSocket{tbl/celldata/missingcount}{1}
425 \NewSocketPlug{tbl/celldata/missingcount}{code}{\tbl_count_missing_cells:n{#1}}
426 \AssignSocketPlug{tbl/celldata/missingcount}{code}
427
428 \cs_new:Npn \__tbl_add_missing_cells:
429   {
```

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:

```

430   \int_compare:nNnT \g__tbl_missing_cells_int > 0
431   {
432     \__tbl_trace:n {==>~
433       ~Inserting~\int_use:N \g__tbl_missing_cells_int \space
434       additional~cell(s)~into~previous~row:}
435     \int_step_inline:nn { \g__tbl_missing_cells_int }
436     {
437       \UseTaggingSocket{tbl/cell/begin}
438       \UseTaggingSocket{tbl/cell/end}
439     }
440   }
441 }
```

(End of definition for `__tbl_add_missing_cells:.`)

`\g__tbl_struct_table_tl` 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.

`\l__tbl_saved_struct_table_tl` `\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*.

`\l__tbl_saved_struct_rows_seq` `\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.

```

442 \tl_new:N \g__tbl_struct_table_tl
443 \tl_new:N \l__tbl_saved_struct_table_tl
444 \seq_new:N \g__tbl_struct_rows_seq
445 \seq_new:N \l__tbl_saved_struct_rows_seq
446 \seq_new:N \g__tbl_struct_cells_seq
```



```

447 \seq_new:N \l__tbl_saved_struct_cells_seq
448 \seq_new:N \g__tbl_struct_cur_seq
449 \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.

```

450 \cs_new_protected:Npn \__tbl_init_struct_data: {
451   \tl_set_eq:NN \l__tbl_saved_struct_table_tl \g__tbl_struct_table_tl
452   \seq_set_eq:NN \l__tbl_saved_struct_rows_seq \g__tbl_struct_rows_seq
453   \seq_set_eq:NN \l__tbl_saved_struct_cells_seq \g__tbl_struct_cells_seq
454   \seq_set_eq:NN \l__tbl_saved_struct_cur_seq \g__tbl_struct_cur_seq
455   %
456   \seq_gclear:N\g__tbl_struct_rows_seq
457   \seq_gclear:N\g__tbl_struct_cells_seq
458   \seq_gclear:N\g__tbl_struct_cur_seq
459 }

```

(End of definition for `__tbl_init_struct_data:`.)

`__tbl_restore_struct_data:`

```

460 \cs_new_protected:Npn \__tbl_restore_struct_data: {
461   \tl_gset_eq:NN \g__tbl_struct_table_tl \l__tbl_saved_struct_table_tl
462   \seq_gset_eq:NN \g__tbl_struct_rows_seq \l__tbl_saved_struct_rows_seq
463   \seq_gset_eq:NN \g__tbl_struct_cells_seq \l__tbl_saved_struct_cells_seq
464   \seq_gset_eq:NN \g__tbl_struct_cur_seq \l__tbl_saved_struct_cur_seq
465 }

```

(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.

```

466 \def\__tbl_patch_LT@makecaption#1#2#3{%
467   \LT@mcol\LT@cols c{%
468     % test can go after merge
469     \str_if_exist:cT { l__socket_tagsupport/parbox/before_plug_str }
470     {
471       \AssignSocketPlug{tagsupport/parbox/before}{noop}
472       \AssignSocketPlug{tagsupport/parbox/after}{noop}
473     }
474     \hbox to\z@{\hss\parbox[t]\LTcapwidth{%
475       \reset@font
476       \tag_stop:n{caption}
477       \sbox\@tempboxa{#1{#2:~}#3}%
478       \tag_start:n{caption}
479       \ifdim\wd\@tempboxa>\hsize
480         #1{#2:~}#3%
481       \else
482         \hbox to\hsize{\hfil#1{#2:~}#3\hfil}%

```

```

483   \fi
484   \endgraf\vskip\baselineskip}%
485   \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.

```

486 \AddToHook{package/longtable/after}
487 {
488   \cs_set_eq:NN \LT@makecaption\tbl_patch_LT@makecaption
489 }
490 </package>
491 <*latex-lab>
492 \ProvidesFile{table-latex-lab-testphase.ltx}
493   [\tblabtbldate\space v\tblabtblversion\space latex-lab wrapper table]
494 \RequirePackage{latex-lab-testphase-table}
495 </latex-lab>
496 <*latex-lab-alias>
497 \ProvidesFile{tabular-latex-lab-testphase.ltx}
498   [\tblabtbldate\space v\tblabtblversion\space latex-lab wrapper tabular]
499 \RequirePackage{latex-lab-testphase-table}
500 </latex-lab-alias>

```