

The sdapslayout package*

Benjamin Berg
benjamin@sipsolutions.net

December 15, 2019

1 Documentation

Please refer to <https://sdaps.org/class-doc> for documentation.

2 Implementation

This package uses the L^AT_EX3 language internally, so we need to enable it.

```
1 % We need at least 2011-08-23 for \keys_set_known:nnN
2 \RequirePackage{expl3}[2011/08/23]
3 %\RequirePackage{xparse}
4 \ExplSyntaxOn
```

And we need a number of other packages.

```
5 \ExplSyntaxOff
6
7 \RequirePackage{sdapsbase}
8 \RequirePackage{sdapsarray}
9 \RequirePackage{xparse}
10
11
12 \ExplSyntaxOn
13
```

2.1 Choice Question Layout

2.1.1 Choice Question Matrix Layout

The following macros provide the functionality to layout choice questions in a matrix like fashion.

```
14
15 \tl_new:N \l_sdaps_choicearray_qobject_type_tl
16 \bool_new:N \l_sdaps_choicearray_horizontal_bool
17 \tl_new:N \l_sdaps_choicearray_var_tl
18 \tl_new:N \l_sdaps_choicearray_text_tl
19 \tl_new:N \l_sdaps_choicearray_layouter_tl
20 \tl_new:N \l_sdaps_choicearray_align_tl
```

*This document corresponds to sdapslayout v0.1, dated 2015/04/10.

```

21 \tl_new:N \l_sdaps_choicearray_extra_tl
22 \tl_new:N \l_sdaps_choicearray_type_tl
23 \tl_new:N \l_sdaps_choice_var_tl
24 \tl_new:N \l_sdaps_choice_val_tl
25 \tl_new:N \l_sdaps_choice_text_tl
26 \tl_new:N \l_sdaps_question_var_tl
27 \tl_new:N \l_sdaps_question_text_tl
28 \clist_new:N \l_sdaps_question_range_clist
29 \int_new:N \g_sdaps_choices_box_int
30 \seq_new:N \g_sdaps_choices_filter_seq
31 \seq_new:N \g_sdaps_choices_cell_seq
32 \seq_new:N \g_sdaps_choices_text_seq
33
34 \keys_define:nn { sdaps / choicearray }
35 {
36   horizontal .bool_set:N = \l_sdaps_choicearray_horizontal_bool,
37   horizontal .default:n = true,
38   horizontal .initial:n = true,
39   vertical .bool_set_inverse:N = \l_sdaps_choicearray_horizontal_bool,
40   vertical .default:n = true,
41   var .tl_set:N = \l_sdaps_choicearray_var_tl,
42   text .tl_set:N = \l_sdaps_choicearray_text_tl,
43   layouter .tl_set:N = \l_sdaps_choicearray_layouter_tl,
44   layouter .initial:n = default,
45   align .tl_set:N = \l_sdaps_choicearray_align_tl,
46   align .initial:n = { },
47
48   type .choices:nn = { multichoice, singlechoice } { \tl_set:Nx \l_sdaps_choicearray_type_tl { #1 } },
49   type .initial:n = { multichoice },
50
51   singlechoice .meta:n = { type=singlechoice },
52   multichoice .meta:n = { type=multichoice },
53
54   noalign .meta:n = { align= },
55 }
56
57 \keys_define:nn { sdaps / choicearray / choice }
58 {
59   var .tl_set:N = \l_sdaps_choice_var_tl,
60   val .tl_set:N = \l_sdaps_choice_val_tl,
61   text .tl_set:N = \l_sdaps_choice_text_tl,
62 }
63
64 \keys_define:nn { sdaps / choicearray / question }
65 {
66   var .tl_set:N = \l_sdaps_question_var_tl,
67   text .tl_set:N = \l_sdaps_question_text_tl,
68
69   range .clist_set:N = \l_sdaps_question_range_clist,
70   range .initial:n = {...}
71 }
72
73 \cs_new_protected_nopar:Npn \sdaps_choicearray_preprocess:nn #1#2
74 {

```

```

75 \keys_set_known:nnN { sdaps / choicearray } { #1 } \l_sdaps_choicearray_extra_tl
76
77 \tl_if_empty:NTF \l_sdaps_choicearray_text_tl {
78   \sdaps_qobject_begin:nnn { choicearray } { Head } { #2 }
79 } {
80   \sdaps_qobject_begin:nnV { choicearray } { Head } \l_sdaps_choicearray_text_tl
81 }
82
83 \tl_if_empty:NF \l_sdaps_choicearray_var_tl {
84   \sdaps_qobject_append_var:V \l_sdaps_choicearray_var_tl
85 }
86
87 \sdaps_checkbox_set_type:V \l_sdaps_choicearray_type_tl
88 \tl_if_eq:VnTF \l_sdaps_choicearray_type_tl { multichoice } {
89   \tl_set:Nn \l_sdaps_choicearray_qobject_type_tl { Choice }
90 } {
91   \tl_set:Nn \l_sdaps_choicearray_qobject_type_tl { Option }
92 }
93 }
94 \cs_generate_variant:Nn \sdaps_choicearray_preprocess:nn { Vn }
95
96 \cs_new_protected_nopar:Npn \sdaps_choicearray_postprocess:
97 {
98   \sdaps_qobject_end:n { choicearray }
99 }
100
101 \cs_new_protected_nopar:Npn \sdaps_choicearray_process_choice_insert_tail_after:w {
102   \bgroup
103   \group_insert_after:N \sdaps_choicearray_process_choice_tail:
104   \sdaps_array_nested_alignenv:
105   \tex_let:D\next=
106 }
107
108 \cs_new_protected_nopar:Nn \sdaps_choicearray_process_choice_tail: {
109   \ignorespaces
110 }
111
112 \cs_new_nopar:Nn \sdaps_choicearray_grab_choice:n {
113   \seq_gput_right:Nn \g_sdaps_choices_text_seq { #1 }
114   \group_begin:
115     \sdaps_array_nested_alignenv:
116     #1
117   \group_end:
118   \sdaps_choicearray_process_choice_tail:
119 }
120
121 \cs_new_nopar:Npn \sdaps_choicearray_process_choice:nw #1
122 {
123   % This modifies grouping so it has to be at the start
124   \sdaps_array_alignment:
125   \leavevmode
126
127   \tl_clear:N \l_sdaps_choice_var_tl
128   \tl_clear:N \l_sdaps_choice_val_tl

```

```

129 \tl_clear:N \l_sdaps_choice_text_tl
130
131 \keys_set:nn { sdaps / choicearray / choice } { #1 }
132
133 \int_gincr:N \g_sdaps_choices_box_int
134
135 \tl_if_empty:NT \l_sdaps_choice_var_tl {
136   % Prefix with _ to force prefixing with generated variable
137   \tl_set:Nx \l_sdaps_choice_var_tl { \int_use:N \g_sdaps_choices_box_int }
138 }
139 \tl_if_empty:NT \l_sdaps_choice_val_tl {
140   \tl_set:Nx \l_sdaps_choice_val_tl { \int_use:N \g_sdaps_choices_box_int }
141 }
142
143 \tl_if_eq:VnTF \l_sdaps_choicearray_type_tl { multichoice } {
144   \seq_gput_right:NV \g_sdaps_choices_filter_seq \l_sdaps_choice_var_tl
145 } {
146   \seq_gput_right:NV \g_sdaps_choices_filter_seq \l_sdaps_choice_val_tl
147 }
148
149 \seq_gput_right:Nx \g_sdaps_choices_cell_seq { \exp_not:n { \sdaps_checkbox:nn } { _ \l_sda
150
151 \tl_if_empty:NTF \l_sdaps_choice_text_tl {
152   % We need to leave a command in the stream that grabs the next parameter
153   % and outputs it immediately
154   \cs_set_eq:NN \l_tmpa_token \sdaps_choicearray_grab_choice:n
155 } {
156   % Nothing else to do
157   \seq_gput_right:NV \g_sdaps_choices_text_seq { \l_sdaps_choice_text_tl }
158   \cs_set_eq:NN \l_tmpa_token \sdaps_choicearray_process_choice_insert_tail_after:w
159 }
160 \l_tmpa_token
161 }
162 \cs_generate_variant:Nn \sdaps_choicearray_process_choice:nw { Vw }
163
164 \cs_new_protected_nopar:Nn \sdaps_choicearray_process_question_grab:n {
165   \tl_set:Nn \l_sdaps_question_text_tl { #1 }
166
167   \sdaps_choicearray_process_question_head:
168
169   \group_begin:
170     \sdaps_array_nested_alignenv:
171     #1
172   \group_end:
173
174   \sdaps_choicearray_process_question_tail:
175 }
176
177 \cs_new_protected_nopar:Npn \sdaps_choicearray_process_question_inline:w {
178   \sdaps_choicearray_process_question_head:
179   \bgroup
180     \group_insert_after:N \sdaps_choicearray_process_question_tail:
181     \sdaps_array_nested_alignenv:
182     \tex_let:D\next=

```

```

183 }
184
185 \cs_new_protected_nopar:Nn \_sdaps_choicearray_process_question_head: {
186   \sdaps_qobject_begin:nnV { choicearray_question } \l_sdaps_choicearray_qobject_type_tl \l_s
187
188   \tl_if_empty:NF \l_sdaps_question_var_tl {
189     \sdaps_qobject_append_var:V \l_sdaps_question_var_tl
190   }
191 }
192
193 \cs_new_protected_nopar:Nn \_sdaps_choicearray_process_question_tail: {
194   \seq_gclear:N \g_tmpa_seq
195   % l_tmpa_bool is whether we are in a run (i.e. ...)
196   \bool_set_false:N \l_tmpa_bool
197
198   \clist_gset_eq:NN \g_tmpa_clist \l_sdaps_question_range_clist
199   \clist_gpop:NN \g_tmpa_clist \l_tmpa_tl
200
201   \seq_map_inline:Nn \g_sdaps_choices_filter_seq {
202     \tl_if_eq:VnT \l_tmpa_tl { ... } {
203       \bool_set_true:N \l_tmpa_bool
204       \clist_gpop:NN \g_tmpa_clist \l_tmpa_tl
205     }
206
207     \tl_if_eq:VnTF \l_tmpa_tl { ##1 } {
208       \seq_gput_right:Nn \g_tmpa_seq { \c_true_bool }
209       \bool_set_false:N \l_tmpa_bool
210       \clist_gpop:NN \g_tmpa_clist \l_tmpa_tl
211     } {
212       % Append if we are handling a run of items and the current item is not
213       % the last one.
214       \bool_if:NTF \l_tmpa_bool {
215         \seq_gput_right:Nn \g_tmpa_seq { \c_true_bool }
216       } {
217         % Otherwise, ignore this item
218         \seq_gput_right:Nn \g_tmpa_seq { \c_false_bool }
219       }
220     }
221   }
222
223   \seq_gset_eq:NN \g_tmpb_seq \g_tmpa_seq
224
225   \seq_map_inline:Nn \g_sdaps_choices_text_seq {
226     \seq_gpop_left:NN \g_tmpa_seq \l_tmpa_tl
227     \tl_if_eq:VnT \l_tmpa_tl { \c_true_bool } {
228       \sdaps_answer:f { ##1 }
229     }
230   }
231
232   \seq_map_inline:Nn \g_sdaps_choices_cell_seq {
233     \sdaps_array_alignment:
234
235     \seq_gpop_left:NN \g_tmpb_seq \l_tmpa_tl
236     \tl_if_eq:VnT \l_tmpa_tl { \c_true_bool } {

```

```

237     ##1
238   }
239 }
240
241 \sdaps_qobject_end:n { choicearray_question }
242 \ignorespaces
243 }
244
245 \cs_new_nopar:Npn \_sdaps_choicearray_process_question:nw #1
246 {
247   \sdaps_array_newline:
248   \leavevmode
249
250   \keys_set:nn { sdaps / choicearray / question } { #1 }
251
252   \tl_if_empty:NTF \l_sdaps_question_text_tl {
253     % We need to leave a command in the stream that grabs the next parameter,
254     % outputs it again, and finishes the question.
255     \cs_set_eq:NN \l_tmpa_token \_sdaps_choicearray_process_question_grab:n
256   } {
257     % Insert the question around the next argument
258     \cs_set_eq:NN \l_tmpa_token \_sdaps_choicearray_process_question_inline:w
259   }
260   \l_tmpa_token
261 }
262 \cs_generate_variant:Nn \_sdaps_choicearray_process_question:nw { Vw }
263
264
265
266 %

```

2.2 Range Question Layout

2.2.1 Range Question Matrix Layout

The following macros provide the functionality to layout range/option questions in a matrix like fashion.

```

267
268
269 \tl_new:N \l_sdaps_rangearray_var_tl
270 \tl_new:N \l_sdaps_rangearray_text_tl
271 \tl_new:N \l_sdaps_rangearray_align_tl
272 \tl_new:N \l_sdaps_rangearray_extra_tl
273 \int_new:N \l_sdaps_rangearray_rangecount_int
274 \bool_new:N \l_sdaps_rangearray_other_bool
275 \tl_new:N \g_sdaps_question_var_tl
276 \tl_new:N \g_sdaps_question_text_tl
277 \tl_new:N \g_sdaps_question_lowertext_tl
278 \tl_new:N \g_sdaps_question_uppertext_tl
279 \tl_new:N \g_sdaps_question_othertext_tl
280
281 \keys_define:nn { sdaps / rangearray }
282 {
283   var          .tl_set:N    = \l_sdaps_rangearray_var_tl,

```

```

284 text      .tl_set:N = \l_sdaps_rangearray_text_tl,
285 count     .int_set:N = \l_sdaps_rangearray_rangecount_int,
286 count     .initial:n = 5,
287 align     .tl_set:N = \l_sdaps_rangearray_align_tl,
288 align     .initial:n = { },
289 other     .bool_set:N = \l_sdaps_rangearray_other_bool,
290 other     .default:n = true,
291 other     .initial:n = false,
292 }
293
294 \keys_define:nn { sdaps / rangearray / question }
295 {
296   var      .tl_gset:N = \g_sdaps_question_var_tl,
297   text     .tl_gset:N = \g_sdaps_question_text_tl,
298   upper    .tl_gset:N = \g_sdaps_question_upper_text_tl,
299   lower    .tl_gset:N = \g_sdaps_question_lower_text_tl,
300   other    .tl_gset:N = \g_sdaps_question_other_text_tl,
301 }
302
303 \cs_new_protected_nopar:Npn \_sdaps_rangearray_preprocess:nn #1#2
304 {
305   \keys_set_known:nnN { sdaps / rangearray } { #1 } \l_sdaps_rangearray_extra_tl
306
307   \tl_if_empty:NTF \l_sdaps_rangearray_text_tl {
308     \sdaps_qobject_begin:nnn { rangearray } { Head } { #2 }
309   } {
310     \sdaps_qobject_begin:nnV { rangearray } { Head } \l_sdaps_rangearray_text_tl
311   }
312
313   \sdaps_checkbox_set_type:n { singlechoice }
314
315   \tl_if_empty:NF \l_sdaps_rangearray_var_tl {
316     \sdaps_qobject_append_var:V \l_sdaps_rangearray_var_tl
317   }
318 }
319 \cs_generate_variant:Nn \_sdaps_rangearray_preprocess:nn { Vn }
320
321 \cs_new_protected_nopar:Npn \_sdaps_rangearray_postprocess:
322 {
323   \sdaps_qobject_end:n { rangearray }
324 }
325
326 % Before/After the different parts
327
328 \cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_before_question: {
329   \sdaps_array_newline:
330   % Note: This needs to be after sdaps_array_newline as the command may be
331   %       discarded otherwise (i.e. it does not make it into the output stream)
332   %       We also need to leave vmode here
333   \leavevmode
334   \sdaps_qobject_begin:nnV { rangearray_question } { Range } \g_sdaps_question_text_tl
335
336   \tl_if_empty:NF \g_sdaps_question_var_tl {
337     \sdaps_qobject_append_var:V \g_sdaps_question_var_tl

```

```

338 }
339
340 \ignorespaces
341 }
342
343 \cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_before_lower: {
344   % right align
345   \sdaps_array_alignment:
346   \leavevmode
347   \sdaps_range:nnV { lower } { 0 } \g_sdaps_question_lowertext_tl
348   \sdaps_if_rtl:F {
349     \hfill
350   }
351   \ignorespaces
352 }
353
354 \cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_before_upper: {
355   \sdaps_array_alignment:
356   \leavevmode
357   \sdaps_range:nnV { upper } { \l_sdaps_rangearray_rangecount_int - 1 } \g_sdaps_question_upper_text_tl
358   \sdaps_if_rtl:T {
359     \hfill
360   }
361   \ignorespaces
362 }
363
364 \cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_before_other: {
365   % Insert an extra empty column for further spacing
366   \sdaps_array_alignment:
367   \sdaps_array_alignment:
368   \leavevmode
369   \sdaps_answer:V \g_sdaps_question_othertext_tl
370   \sdaps_if_rtl:TF {
371     \hfill
372   } {
373     \sdaps_checkbox:nn { } { 0 } {} ~ {}
374   }
375   \ignorespaces
376 }
377
378
379
380 \cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_after_question: {
381   % Insert an extra empty column for further spacing
382   \sdaps_array_alignment:
383   \tl_if_empty:NTF \g_sdaps_question_lowertext_tl {
384     \cs_set_eq:NN \l_tmpa_token \_sdaps_rangearray_process_question_grab_lower:n
385   } {
386     \cs_set_eq:NN \l_tmpa_token \_sdaps_rangearray_process_question_inline_lower:w
387   }
388   \l_tmpa_token
389 }
390
391 \cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_after_lower: {

```



```

392 % Insert the option checkbox column
393 \sdaps_if_rtl:T {
394   \hfill\kern Opt
395 }
396 \sdaps_array_alignment:
397 \leavevmode
398 % Seems like right to left writing mode is not started without a paragraph
399 \sdaps_if_rtl:T { \beginR }
400 % Assume we have at least one checkbox
401 \sdaps_checkbox:nn { } { 1 }
402 \int_step_inline:nnnn { 2 } { 1 } { \l_sdaps_rangearray_rangecount_int } {
403   \hspace{1em} \sdaps_checkbox:nn { } { ##1 }
404 }
405 \sdaps_if_rtl:T { \endR }
406
407 \tl_if_empty:NTF \g_sdaps_question_upper_text_tl {
408   \cs_set_eq:NN \l_tmpa_token \sdaps_rangearray_process_question_grab_upper:n
409 } {
410   \cs_set_eq:NN \l_tmpa_token \sdaps_rangearray_process_question_inline_upper:w
411 }
412 \l_tmpa_token
413 }
414
415 \cs_new_protected_nopar:Nn \sdaps_rangearray_process_question_after_upper: {
416   \sdaps_if_rtl:F {
417     \hfill\kern Opt
418   }
419
420   \bool_if:NTF \l_sdaps_rangearray_other_bool {
421     \tl_if_empty:NTF \g_sdaps_question_othertext_tl {
422       \cs_set_eq:NN \l_tmpa_token \sdaps_rangearray_process_question_grab_other:n
423     } {
424       \cs_set_eq:NN \l_tmpa_token \sdaps_rangearray_process_question_inline_other:w
425     }
426   } {
427     \cs_set_eq:NN \l_tmpa_token \sdaps_rangearray_process_question_finish:
428   }
429   \l_tmpa_token
430 }
431
432 \cs_new_protected_nopar:Nn \sdaps_rangearray_process_question_after_other: {
433   \sdaps_if_rtl:TF {
434     {} ~ \sdaps_checkbox:nn { } { 0 } {}
435   } {
436     \hfill\kern Opt
437   }
438   \ignorespaces
439 }
440
441 \cs_new_protected_nopar:Nn \sdaps_rangearray_process_question_finish: {
442   \sdaps_qobject_end:n { rangearray_question }
443   \ignorespaces
444 }
445

```

```

446 % Processors for inline processing/grabbing the argument
447
448 \cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_grab_question:n {
449   \tl_gset:Nn \g_sdaps_question_text_tl { #1 }
450
451   \_sdaps_rangearray_process_question_before_question:
452   \group_begin:
453     \sdaps_array_nested_alignenv:
454     #1
455   \group_end:
456   \_sdaps_rangearray_process_question_after_question:
457 }
458
459 \cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_grab_lower:n {
460   \tl_gset:Nn \g_sdaps_question_lowertext_tl { #1 }
461
462   \_sdaps_rangearray_process_question_before_lower:
463   \group_begin:
464     \sdaps_array_nested_alignenv:
465     #1
466   \group_end:
467   \_sdaps_rangearray_process_question_after_lower:
468 }
469
470 \cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_grab_upper:n {
471   \tl_gset:Nn \g_sdaps_question_uptertext_tl { #1 }
472
473   \_sdaps_rangearray_process_question_before_upper:
474   \group_begin:
475     \sdaps_array_nested_alignenv:
476     #1
477   \group_end:
478   \_sdaps_rangearray_process_question_after_upper:
479 }
480
481 \cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_grab_other:n {
482   \tl_gset:Nn \g_sdaps_question_othertext_tl { #1 }
483
484   % If the text is empty, assume that this particular question does not have
485   % an alternative choice. Note that this column might not exist and this
486   % macro will not even be called in that case.
487   % If we skip the optional "other" option then we still need to insert the
488   % alignment to create the column.
489   \tl_if_empty:NTF \g_sdaps_question_othertext_tl {
490     \sdaps_array_alignment:
491     \leavevmode
492     \ignorespaces
493   } {
494     \_sdaps_rangearray_process_question_before_other:
495     \group_begin:
496       \sdaps_array_nested_alignenv:
497       #1
498     \group_end:
499     \_sdaps_rangearray_process_question_after_other:

```

```

500 }
501 \_sdaps_rangearray_process_question_finish:
502 }
503
504
505
506 \cs_new_protected_nopar:Npn \_sdaps_rangearray_process_question_inline_question:w {
507   \_sdaps_rangearray_process_question_before_question:
508   \bgroup
509   \group_insert_after:N \_sdaps_rangearray_process_question_after_question:
510   \sdaps_array_nested_alignenv:
511   \tex_let:D\next=
512 }
513
514 \cs_new_protected_nopar:Npn \_sdaps_rangearray_process_question_inline_lower:w {
515   \_sdaps_rangearray_process_question_before_lower:
516   \bgroup
517   \group_insert_after:N \_sdaps_rangearray_process_question_after_lower:
518   \tex_let:D\next=
519 }
520
521 \cs_new_protected_nopar:Npn \_sdaps_rangearray_process_question_inline_upper:w {
522   \_sdaps_rangearray_process_question_before_upper:
523   \bgroup
524   \group_insert_after:N \_sdaps_rangearray_process_question_after_upper:
525   \tex_let:D\next=
526 }
527
528 \cs_new_protected_nopar:Npn \_sdaps_rangearray_process_question_inline_other:w {
529   \_sdaps_rangearray_process_question_before_other:
530   % If we reach this macro then a text has been set for the other item. This
531   % means we never need to ignore the "other" parameter at this point.
532   \bgroup
533   \group_insert_after:N \_sdaps_rangearray_process_question_after_other:
534   \group_insert_after:N \_sdaps_rangearray_process_question_finish:
535   \tex_let:D\next=
536 }
537
538
539
540
541 \cs_new_nopar:Npn \_sdaps_rangearray_process_question:nw #1
542 {
543   % Is there a better way other than clearing these before parsing?
544   \tl_gclear:N \g_sdaps_question_var_tl
545   \tl_gclear:N \g_sdaps_question_text_tl
546   \tl_gclear:N \g_sdaps_question_upper_text_tl
547   \tl_gclear:N \g_sdaps_question_lower_text_tl
548   \tl_gclear:N \g_sdaps_question_other_text_tl
549
550   \keys_set:nn { sdaps / rangearray / question } { #1 }
551
552   \tl_if_empty:NTF \g_sdaps_question_text_tl {
553     % We need to leave a command in the stream that grabs the next parameter,

```

```

554 % outputs it again, and finishes the question.
555 \cs_set_eq:NN \l_tmpa_token \sdaps_rangearray_process_question_grab_question:n
556 } {
557 % We need to generate the question after the next group stops
558 \cs_set_eq:NN \l_tmpa_token \sdaps_rangearray_process_question_inline_question:w
559 }
560 \l_tmpa_token
561 }
562 \cs_generate_variant:Nn \sdaps_rangearray_process_question:nw { Vw }
563
564
565
566 %

```

2.3 Export user facing environments

```

567 %
568
569
570 \newenvironment { choicearray } [ 2 ] []
571 {
572 \group_begin:
573
574 \sdaps_context_get:nN { choicearray } \l_tmpa_tl
575 \tl_if_eq:NNT \l_tmpa_tl \q_no_value {
576 \tl_set:Nn \l_tmpa_tl {}
577 }
578
579 \tl_if_empty:nF { #1 } {
580 \tl_if_empty:NTF \l_tmpa_tl {
581 \tl_set:Nn \l_tmpa_tl { #1 }
582 } {
583 \tl_set:Nf \l_tmpa_tl { \l_tmpa_tl, #1 }
584 }
585 }
586
587 \sdaps_choicearray_preprocess:Vn \l_tmpa_tl { #2 }
588 % Clear the variables
589 \seq_gclear:N \g_sdaps_choices_filter_seq
590 \seq_gclear:N \g_sdaps_choices_cell_seq
591 \seq_gclear:N \g_sdaps_choices_text_seq
592
593 \int_gzero:N \g_sdaps_choices_box_int
594
595 % Define new commands
596 \newcommand \choice [ 1 ] [] {
597 \sdaps_choicearray_process_choice:nw { ##1 }
598 }
599 \newcommand \question [ 1 ] [] {
600 \sdaps_choicearray_process_question:nw { ##1 }
601 }
602
603 \group_begin:
604

```

```

605     \tl_set:Nx \l_tmpb_tl {keepenv,layouter=\tl_use:N\l_sdaps_choicearray_layouter_tl,align=left}
606     \expandafter\sdapsarray\expandafter[\l_tmpb_tl]
607 }
608 {
609     \endsdapsarray
610     \group_end:
611     % Process keys
612     \_sdaps_choicearray_postprocess:
613
614     \group_end:
615 }
616
617 \newenvironment { optionarray } [ 2 ] []
618 {
619     \choicearray[singlechoice,#1] { #2 }
620 }
621 {
622     \endchoicearray
623 }
624
625 \newenvironment { rangearray } [ 2 ] []
626 {
627     \group_begin:
628
629     \sdaps_context_get:nN { rangearray } \l_tmpa_tl
630     \tl_if_eq:NNT \l_tmpa_tl \q_no_value {
631         \tl_set:Nn \l_tmpa_tl {}
632     }
633
634     \tl_if_empty:nF { #1 } {
635         \tl_if_empty:NTF \l_tmpa_tl {
636             \tl_set:Nn \l_tmpa_tl { #1 }
637         } {
638             \tl_set:Nf \l_tmpa_tl { \l_tmpa_tl, #1 }
639         }
640     }
641
642     \_sdaps_rangearray_preprocess:Vn \l_tmpa_tl { #2 }
643
644     \newcommand \range [ 1 ] [] {
645         \_sdaps_rangearray_process_question:nw { ##1 }
646     }
647
648     \group_begin:
649
650     \tl_set:Nx \l_tmpb_tl {keepenv,align=\l_sdaps_rangearray_align_tl,\l_sdaps_rangearray_layouter_tl}
651     \expandafter\sdapsarray\expandafter[\l_tmpb_tl]
652 }
653 {
654     \endsdapsarray
655     \group_end:
656     % Process keys
657     \_sdaps_rangearray_postprocess:
658

```

```
659 \group_end:  
660 }  
661  
662  
663 \ExplSyntaxOff  
664  
665 %
```