

The `sdapslayout` package*

Benjamin Berg
`benjamin@sipsolutions.net`

February 15, 2020

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_layouter_tl
18 \tl_new:N \l_sdaps_choicearray_align_tl
19 \tl_new:N \l_sdaps_choicearray_extra_tl
20 \tl_new:N \l_sdaps_choicearray_type_tl
```

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

```

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

```

```

75      \tl_set:Nn \l_sdaps_choicearray_qobject_type_tl { Choice }
76  } {
77    \tl_set:Nn \l_sdaps_choicearray_qobject_type_tl { Option }
78  }
79 }
80 \cs_generate_variant:Nn \_sdaps_choicearray_preprocess:n { V }
81
82 \cs_new_protected_nopar:Npn \_sdaps_choicearray_process_choice_insert_tail_after:w {
83   \bgroup
84     \group_insert_after:N \_sdaps_choicearray_process_choice_tail:
85     \sdaps_array_nested_alignenv:
86     \tex_let:D\next=
87 }
88
89 \cs_new_protected_nopar:Nn \_sdaps_choicearray_process_choice_tail: {
90   \ignorespaces
91 }
92
93 \cs_new_nopar:Nn \_sdaps_choicearray_grab_choice:n {
94   \seq_gput_right:Nn \g_sdaps_choices_text_seq { #1 }
95   \group_begin:
96     \sdaps_array_nested_alignenv:
97     #1
98   \group_end:
99   \_sdaps_choicearray_process_choice_tail:
100 }
101
102 \cs_new_nopar:Npn \_sdaps_choicearray_process_choice:nw #1
103 {
104   % This modifies grouping so it has to be at the start
105   \sdaps_array_alignment:
106   \leavevmode
107
108   \tl_clear:N \l_sdaps_choice_var_tl
109   \tl_clear:N \l_sdaps_choice_val_tl
110   \tl_clear:N \l_sdaps_choice_text_tl
111
112   \keys_set:nn { sdaps / choicearray / choice } { #1 }
113
114   \int_gincr:N \g_sdaps_choices_box_int
115
116   \tl_if_empty:NT \l_sdaps_choice_var_tl {
117     % Prefix with _ to force prefixing with generated variable
118     \tl_set:Nx \l_sdaps_choice_var_tl { \int_use:N \g_sdaps_choices_box_int }
119   }
120   \tl_if_empty:NT \l_sdaps_choice_val_tl {
121     \tl_set:Nx \l_sdaps_choice_val_tl { \int_use:N \g_sdaps_choices_box_int }
122   }
123
124   \tl_if_eq:VnTF \l_sdaps_choicearray_type_tl { multichoice } {
125     \seq_gput_right:NV \g_sdaps_choices_filter_seq \l_sdaps_choice_var_tl
126   } {
127     \seq_gput_right:NV \g_sdaps_choices_filter_seq \l_sdaps_choice_val_tl
128   }

```

```

129
130 \seq_gput_right:Nx \g_sdaps_choices_cell_seq { \exp_not:n { \sdaps_checkbox:nn } { _ \l_sdap
131
132 \tl_if_empty:NTF \l_sdaps_choice_text_tl {
133     % We need to leave a command in the stream that grabs the next parameter
134     % and outputs it immediately
135     \cs_set_eq:NN \l_tmpa_token \sdaps_choicearray_grab_choice:n
136 }
137     % Nothing else to do
138     \seq_gput_right:NV \g_sdaps_choices_text_seq { \l_sdaps_choice_text_tl }
139     \cs_set_eq:NN \l_tmpa_token \sdaps_choicearray_process_choice_insert_tail_after:w
140 }
141 \l_tmpa_token
142 }
143 \cs_generate_variant:Nn \sdaps_choicearray_process_choice:nw { Vw }
144
145 \cs_new_protected_nopar:Nn \sdaps_choicearray_process_question_grab:n
146     \tl_set:Nn \l_sdaps_question_text_tl { #1 }
147
148 \_sdaps_choicearray_process_question_head:
149
150 \group_begin:
151     \sdaps_array_nested_alignenv:
152         #1
153 \group_end:
154
155 \_sdaps_choicearray_process_question_tail:
156 }
157
158 \cs_new_protected_nopar:Npn \_sdaps_choicearray_process_question_inline:w {
159     \_sdaps_choicearray_process_question_head:
160     \bgroup
161         \group_insert_after:N \_sdaps_choicearray_process_question_tail:
162         \sdaps_array_nested_alignenv:
163         \tex_let:D\next=
164 }
165
166 \cs_new_protected_nopar:Nn \_sdaps_choicearray_process_question_head: {
167     \sdaps_qobject_begin:nnV { choicearray_question } \l_sdaps_choicearray_qobject_type_tl \l_s
168
169 \tl_if_empty:NF \l_sdaps_question_var_tl {
170     \sdaps_qobject_append_var:V \l_sdaps_question_var_tl
171 }
172 }
173
174 \cs_new_protected_nopar:Nn \_sdaps_choicearray_process_question_tail: {
175     \seq_gclear:N \g_tmpa_seq
176     % l_tmpa_bool is whether we are in a run (i.e. ....)
177     \bool_set_false:N \l_tmpa_bool
178
179     \clist_gset_eq:NN \g_tmpa_clist \l_sdaps_question_range_clist
180     \clist_gpop>NN \g_tmpa_clist \l_tmpa_tl
181
182     \seq_map_inline:Nn \g_sdaps_choices_filter_seq {

```

```

183 \tl_if_eq:VnT \l_tmpa_tl { ... } {
184     \bool_set_true:N \l_tmpa_bool
185     \clist_gpop>NN \g_tmpa_clist \l_tmpa_tl
186 }
187
188 \tl_if_eq:VnTF \l_tmpa_tl { ##1 } {
189     \seq_gput_right:Nn \g_tmpa_seq { \c_true_bool }
190     \bool_set_false:N \l_tmpa_bool
191     \clist_gpop>NN \g_tmpa_clist \l_tmpa_tl
192 } {
193     % Append if we are handling a run of items and the current item is not
194     % the last one.
195     \bool_if:NTF \l_tmpa_bool {
196         \seq_gput_right:Nn \g_tmpa_seq { \c_true_bool }
197     } {
198         % Otherwise, ignore this item
199         \seq_gput_right:Nn \g_tmpa_seq { \c_false_bool }
200     }
201 }
202 }
203
204 \seq_gset_eq:NN \g_tmpb_seq \g_tmpa_seq
205
206 \seq_map_inline:Nn \g_sdaps_choices_text_seq {
207     \seq_gpop_left>NN \g_tmpa_seq \l_tmpa_tl
208     \tl_if_eq:VnT \l_tmpa_tl { \c_true_bool } {
209         \sdaps_answer:f { ##1 }
210     }
211 }
212
213 \seq_map_inline:Nn \g_sdaps_choices_cell_seq {
214     \sdaps_array_alignment:
215
216     \seq_gpop_left>NN \g_tmpb_seq \l_tmpa_tl
217     \tl_if_eq:VnT \l_tmpa_tl { \c_true_bool } {
218         ##1
219     }
220 }
221
222 \sdaps_qobject_end:n { choicearray_question }
223 \ignorespaces
224 }
225
226 \cs_new_nopar:Npn \_sdaps_choicearray_process_question:nw #1
227 {
228     \sdaps_array_newline:
229     \leavevmode
230
231     \keys_set:nn { sdaps / choicearray / question } { #1 }
232
233 \tl_if_empty:NTF \l_sdaps_question_text_tl {
234     % We need to leave a command in the stream that grabs the next parameter,
235     % outputs it again, and finishes the question.
236     \cs_set_eq:NN \l_tmpa_token \_sdaps_choicearray_process_question_grab:n

```

```

237 } {
238     % Insert the question around the next argument
239     \cs_set_eq:NN \l_tmpa_token \sdaps_choicearray_process_question_inline:w
240 }
241 \l_tmpa_token
242 }
243 \cs_generate_variant:Nn \sdaps_choicearray_process_question:nw { Vw }
244
245
246
247 %

```

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.

```

248
249
250 \tl_new:N \l_sdaps_rangearray_align_tl
251 \tl_new:N \l_sdaps_rangearray_extra_tl
252 \int_new:N \l_sdaps_rangearray_rangecount_int
253 \bool_new:N \l_sdaps_rangearray_other_bool
254 \tl_new:N \g_sdaps_question_var_tl
255 \tl_new:N \g_sdaps_question_text_tl
256 \tl_new:N \g_sdaps_question_lowertext_tl
257 \tl_new:N \g_sdaps_question_uppertext_tl
258 \tl_new:N \g_sdaps_question_othertext_tl
259
260 \msg_new:nnn { sdapslayout } { option_not_supported } { The~#1~option~is~not~supported~by~this~macro~}
261
262 \keys_define:nn { sdaps / rangearray }
263 {
264     count      .int_set:N = \l_sdaps_rangearray_rangecount_int,
265     count      .initial:n = 5,
266     align      .tl_set:N = \l_sdaps_rangearray_align_tl,
267     align      .initial:n = { },
268     % Override and disallow flipping; it does not work currently
269     flip       .code:n = { \msg_error:nnn { sdapslayout } { option_not_supported } { flip } },
270     other      .bool_set:N = \l_sdaps_rangearray_other_bool,
271     other      .default:n = true,
272     other      .initial:n = false,
273 }
274
275 \keys_define:nn { sdaps / rangearray / question }
276 {
277     var        .tl_gset:N = \g_sdaps_question_var_tl,
278     text       .tl_gset:N = \g_sdaps_question_text_tl,
279     upper      .tl_gset:N = \g_sdaps_question_uppertext_tl,
280     lower      .tl_gset:N = \g_sdaps_question_lowertext_tl,
281     other      .tl_gset:N = \g_sdaps_question_othertext_tl,
282 }
283

```

```

284 \cs_new_protected_nopar:Npn \_sdaps_rangearray_preprocess:n #1
285 {
286     \keys_set_known:nnN { sdaps / rangearray } { #1 } \l_sdaps_rangearray_extra_tl
287
288     \sdaps_checkbox_set_type:n { singlechoice }
289 }
290 \cs_generate_variant:Nn \_sdaps_rangearray_preprocess:n { V }
291
292 % Before/After the different parts
293
294 \cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_before_question: {
295     \sdaps_array_newline:
296     % Note: This needs to be after sdaps_array_newline as the command may be
297     %        discarded otherwise (i.e. it does not make it into the output stream)
298     %        We also need to leave vmode here
299     \leavevmode
300     \sdaps_qobject_begin:nnV { rangearray_question } { Range } \g_sdaps_question_text_tl
301
302     \tl_if_empty:NF \g_sdaps_question_var_tl {
303         \sdaps_qobject_append_var:V \g_sdaps_question_var_tl
304     }
305
306     \ignorespaces
307 }
308
309 \cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_before_lower: {
310     % right align
311     \sdaps_array_alignment:
312     \leavevmode
313     \sdaps_range:nnV { lower } { 0 } \g_sdaps_question_lowertext_tl
314     \sdaps_if_rtl:F {
315         \hfill
316     }
317     \ignorespaces
318 }
319
320 \cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_before_upper: {
321     \sdaps_array_alignment:
322     \leavevmode
323     \sdaps_range:nnV { upper } { \l_sdaps_rangearray_rangecount_int - 1 } \g_sdaps_question_upp
324     \sdaps_if_rtl:T {
325         \hfill
326     }
327     \ignorespaces
328 }
329
330 \cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_before_other: {
331     % Insert an extra empty column for further spacing
332     \sdaps_array_alignment:
333     \sdaps_array_alignment:
334     \leavevmode
335     \sdaps_answer:V \g_sdaps_question_othertext_tl
336     \sdaps_if_rtl:TF {
337         \hfill

```

```

338 } {
339   \sdaps_checkbox:nn { } { 0 } {} ~ {}
340 }
341 \ignorespaces
342 }

343
344
345
346 \cs_new_protected_nopar:Nn \sdaps_rangearray_process_question_after_question: {
347   % Insert an extra empty column for further spacing
348   \sdaps_array_alignment:
349   \tl_if_empty:NTF \g_sdaps_question_lowertext_tl {
350     \cs_set_eq:NN \l_tmpa_token \sdaps_rangearray_process_question_grab_lower:n
351   } {
352     \cs_set_eq:NN \l_tmpa_token \sdaps_rangearray_process_question_inline_lower:w
353   }
354   \l_tmpa_token
355 }
356
357 \cs_new_protected_nopar:Nn \sdaps_rangearray_process_question_after_lower: {
358   % Insert the option checkbox column
359   \sdaps_if_rtl:T {
360     \hfill\kern 0pt
361   }
362   \sdaps_array_alignment:
363   \leavevmode
364   % Seems like right to left writing mode is not started without a paragraph
365   \sdaps_if_rtl:T { \beginR }
366   % Assume we have at least one checkbox
367   \sdaps_checkbox:nn { } { 1 }
368   \int_step_inline:nnnn { 2 } { 1 } { \l_sdaps_rangearray_rangecount_int } {
369     \hspace{1em} \sdaps_checkbox:nn { } { ##1 }
370   }
371   \sdaps_if_rtl:T { \endR }

372
373   \tl_if_empty:NTF \g_sdaps_question_uppertext_tl {
374     \cs_set_eq:NN \l_tmpa_token \sdaps_rangearray_process_question_grab_upper:n
375   } {
376     \cs_set_eq:NN \l_tmpa_token \sdaps_rangearray_process_question_inline_upper:w
377   }
378   \l_tmpa_token
379 }

380 \cs_new_protected_nopar:Nn \sdaps_rangearray_process_question_after_upper: {
381   \sdaps_if_rtl:F {
382     \hfill\kern 0pt
383   }
384 }

385
386 \bool_if:NTF \l_sdaps_rangearray_other_bool {
387   \tl_if_empty:NTF \g_sdaps_question_others_text_tl {
388     \cs_set_eq:NN \l_tmpa_token \sdaps_rangearray_process_question_grab_other:n
389   } {
390     \cs_set_eq:NN \l_tmpa_token \sdaps_rangearray_process_question_inline_other:w
391   }

```

```

392 } {
393   \cs_set_eq:NN \l_tmpa_token \sdaps_rangearray_process_question_finish:
394 }
395 \l_tmpa_token
396 }
397
398 \cs_new_protected_nopar:Nn \sdaps_rangearray_process_question_after_other: {
399   \sdaps_if_rtl:TF {
400     {} ~ \sdaps_checkbox:nn { } { 0 } {}
401   } {
402     \hfill\kern .0pt
403   }
404   \ignorespaces
405 }
406
407 \cs_new_protected_nopar:Nn \sdaps_rangearray_process_question_finish: {
408   \sdaps_qobject_end:n { rangearray_question }
409   \ignorespaces
410 }
411
412 % Processors for inline processing/grabbing the argument
413
414 \cs_new_protected_nopar:Nn \sdaps_rangearray_process_question_grab_question:n {
415   \tl_gset:Nn \g_sdaps_question_text_tl { #1 }
416
417   \sdaps_rangearray_process_question_before_question:
418   \group_begin:
419     \sdaps_array_nested_alignenv:
420     #1
421   \group_end:
422   \sdaps_rangearray_process_question_after_question:
423 }
424
425 \cs_new_protected_nopar:Nn \sdaps_rangearray_process_question_grab_lower:n {
426   \tl_gset:Nn \g_sdaps_question_lowertext_tl { #1 }
427
428   \sdaps_rangearray_process_question_before_lower:
429   \group_begin:
430     \sdaps_array_nested_alignenv:
431     #1
432   \group_end:
433   \sdaps_rangearray_process_question_after_lower:
434 }
435
436 \cs_new_protected_nopar:Nn \sdaps_rangearray_process_question_grab_upper:n {
437   \tl_gset:Nn \g_sdaps_question_uppertext_tl { #1 }
438
439   \sdaps_rangearray_process_question_before_upper:
440   \group_begin:
441     \sdaps_array_nested_alignenv:
442     #1
443   \group_end:
444   \sdaps_rangearray_process_question_after_upper:
445 }

```

```

446
447 \cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_grab_other:n {
448   \tl_gset:Nn \g_sdaps_question_othertext_tl { #1 }
449
450   % If the text is empty, assume that this particular question does not have
451   % an alternative choice. Note that this column might not exist and this
452   % macro will not even be called in that case.
453   % If we skip the optional "other" option then we still need to insert the
454   % alignment to create the column.
455   \tl_if_empty:NTF \g_sdaps_question_othertext_tl {
456     \sdaps_array_alignment:
457     \leavevmode
458     \ignorespaces
459   } {
460     \_sdaps_rangearray_process_question_before_other:
461     \group_begin:
462       \sdaps_array_nested_alignenv:
463       #1
464     \group_end:
465     \_sdaps_rangearray_process_question_after_other:
466   }
467   \_sdaps_rangearray_process_question_finish:
468 }
469
470
471
472 \cs_new_protected_nopar:Npn \_sdaps_rangearray_process_question_inline_question:w {
473   \_sdaps_rangearray_process_question_before_question:
474   \bgroup
475     \group_insert_after:N \_sdaps_rangearray_process_question_after_question:
476     \sdaps_array_nested_alignenv:
477     \tex_let:D\next=
478 }
479
480 \cs_new_protected_nopar:Npn \_sdaps_rangearray_process_question_inline_lower:w {
481   \_sdaps_rangearray_process_question_before_lower:
482   \bgroup
483     \group_insert_after:N \_sdaps_rangearray_process_question_after_lower:
484     \tex_let:D\next=
485 }
486
487 \cs_new_protected_nopar:Npn \_sdaps_rangearray_process_question_inline_upper:w {
488   \_sdaps_rangearray_process_question_before_upper:
489   \bgroup
490     \group_insert_after:N \_sdaps_rangearray_process_question_after_upper:
491     \tex_let:D\next=
492 }
493
494 \cs_new_protected_nopar:Npn \_sdaps_rangearray_process_question_inline_other:w {
495   \_sdaps_rangearray_process_question_before_other:
496   % If we reach this macro then a text has been set for the other item. This
497   % means we never need to ignore the "other" parameter at this point.
498   \bgroup
499     \group_insert_after:N \_sdaps_rangearray_process_question_after_other:

```

```

500     \group_insert_after:N \_sdaps_rangearray_process_question_finish:
501     \tex_let:D\next=
502 }
503
504
505
506
507 \cs_new_nopar:Npn \_sdaps_rangearray_process_question:nw #1
508 {
509     % Is there a better way other than clearing these before parsing?
510     \tl_gclear:N \g_sdaps_question_var_tl
511     \tl_gclear:N \g_sdaps_question_text_tl
512     \tl_gclear:N \g_sdaps_question_uppertext_tl
513     \tl_gclear:N \g_sdaps_question_lowertext_tl
514     \tl_gclear:N \g_sdaps_question_othertext_tl
515
516     \keys_set:nn { sdaps / rangearray / question } { #1 }
517
518     \tl_if_empty:NTF \g_sdaps_question_text_tl {
519         % We need to leave a command in the stream that grabs the next parameter,
520         % outputs it again, and finishes the question.
521         \cs_set_eq:NN \l_tmpa_token \_sdaps_rangearray_process_question_grab_question:n
522     } {
523         % We need to generate the question after the next group stops
524         \cs_set_eq:NN \l_tmpa_token \_sdaps_rangearray_process_question_inline_question:w
525     }
526     \l_tmpa_token
527 }
528 \cs_generate_variant:Nn \_sdaps_rangearray_process_question:nw { Vw }
529
530
531
532 %

```

2.3 Export user facing environments

```

533 %
534
535
536 \newenvironment { choicearray } [ 1 ] []
537 {
538     \group_begin:
539
540     \sdaps_context_get:nN { choicearray } \l_tmpa_tl
541     \tl_if_eq:NNT \l_tmpa_tl \q_no_value {
542         \tl_set:Nn \l_tmpa_tl {}
543     }
544
545     \tl_if_empty:nF { #1 } {
546         \tl_if_empty:NTF \l_tmpa_tl {
547             \tl_set:Nn \l_tmpa_tl { #1 }
548         } {
549             \tl_set:Nf \l_tmpa_tl { \l_tmpa_tl, #1 }
550         }
551

```

```

551 }
552
553 \_sdaps_choicearray_preprocess:V \l_tmpa_tl
554 % Clear the variables
555 \seq_gclear:N \g_sdaps_choices_filter_seq
556 \seq_gclear:N \g_sdaps_choices_cell_seq
557 \seq_gclear:N \g_sdaps_choices_text_seq
558
559 \int_gzero:N \g_sdaps_choices_box_int
560
561 % Define new commands
562 \newcommand \choice [ 1 ] []
563   \_sdaps_choicearray_process_choice:nw { ##1 }
564 }
565 \newcommand \question [ 1 ] []
566   \_sdaps_choicearray_process_question:nw { ##1 }
567 }
568
569 \group_begin:
570
571   \tl_set:Nx \l_tmpb_tl {keepenv,layouter=\tl_use:N\l_sdaps_choicearray_layouter_tl,align}
572   \expandafter\sdapsarray\expandafter[\l_tmpb_tl]
573 }
574 {
575   \endsdapsarray
576 \group_end:
577
578 \group_end:
579 }
580
581 \newenvironment { optionarray } [ 1 ] []
582 {
583   \choicearray[singlechoice,#1]
584 }
585 {
586   \endchoicearray
587 }
588
589 \newenvironment { rangearray } [ 1 ] []
590 {
591   \group_begin:
592
593   \sdaps_context_get:nN { rangearray } \l_tmpa_tl
594   \tl_if_eq:NNT \l_tmpa_tl \q_no_value {
595     \tl_set:Nn \l_tmpa_tl {}
596   }
597
598   \tl_if_empty:nF { #1 } {
599     \tl_if_empty:NTF \l_tmpa_tl {
600       \tl_set:Nn \l_tmpa_tl { #1 }
601     }
602     \tl_set:Nf \l_tmpa_tl { \l_tmpa_tl, #1 }
603   }
604 }

```

```

605   \_sdaps_rangearray_preprocess:V \l_tmpa_tl
606
607   \newcommand \range [ 1 ] [] {
608     \_sdaps_rangearray_process_question:nw { ##1 }
609   }
610
611   \group_begin:
612
613     \tl_set:Nx \l_tmpb_tl {keepenv,align=\l_sdaps_rangearray_align_tl,no_header,\l_sdaps_ran
614       \expandafter\sdapsarray\expandafter[\l_tmpb_tl]
615     }
616   {
617     \endsdapsarray
618   }
619   \group_end:
620
621   \group_end:
622 }
623
624 \ExplSyntaxOff
625
626 %
627 %

```