

The `sdapsbase` 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 \LaTeX 3 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 \input{sdapscode128}
8
9 \RequirePackage{qrcode}
10
11 \RequirePackage{tikz}
12 \usetikzlibrary{calc}
13 \usetikzlibrary{positioning}
14 \usetikzlibrary{decorations.pathmorphing}
15 \ExplSyntaxOn
16
17
18 % Define aliases for code128 functions, and generate variants
19 \cs_new_eq:NN \code_render:n \code
20 \cs_generate_variant:Nn \code_render:n { x, V }
21
22 % Also define an alias for the qrcode renderer
23 \cs_new_protected_nopar:Nn \qrcode_render:nn {
```

*This document corresponds to `sdapsbase` v0.1, dated 2015/01/14.

```

24 \qrcode[#1] {#2}
25 }
26 \cs_generate_variant:Nn \qrcode_render:nn { nx, nV }
27
28
29 % Define a method to check for RTL languages, as the relevant commands
30 % may not always be defined.
31 \prg_new_conditional:Npnn \sdaps_if_rtl: { p, T, F, TF }
32 {
33   \cs_if_exist:cTF { if@RTL } {
34     \tl_use:c { if@RTL }
35     \prg_return_true:
36   } else
37     \prg_return_false:
38   \fi
39 } {
40   \prg_return_false:
41 }
42 }
43

```

2.1 Context Handling

When creating complex questionnaires we need a mechanism to handle the current context. By choice this mechanism works in global scope as items inside a nested environment (e.g. `multicol`) need to have an effect on items outside the environment. Because of this, we implement our own context, which works similar to TeX groups containing local definitions.

```

44
45 \cs_generate_variant:Nn \tl_if_eq:nnTF { Vn }
46 \cs_generate_variant:Nn \tl_if_eq:nnT { Vn }
47 \cs_generate_variant:Nn \tl_if_eq:nnF { Vn }
48 \cs_generate_variant:Nn \int_if_odd:nTF { V }
49 \cs_generate_variant:Nn \int_if_odd:nF { V }
50 \cs_generate_variant:Nn \int_if_odd:nT { V }
51 \cs_generate_variant:Nn \tl_set:Nn { Nv }
52 \cs_generate_variant:Nn \msg_error:nnn { nnV }
53 \cs_generate_variant:Nn \exp_not:n { f }
54
55
56 \tl_new:N \l__sdaps_tmpa_tl
57 \tl_new:N \l__sdaps_tmpb_tl
58
59 \dim_new:N \l__sdaps_tmpa_dim %
60 \dim_new:N \l__sdaps_tmpb_dim %
61
62 \prop_new:N \g__sdaps_current_context_prop
63 \tl_new:N \g__sdaps_current_context_id_tl
64

```

```

65 \tl_new:N \g__sdaps_current_context_tl
66
67 \seq_new:N \g__sdaps_context_ids_seq
68 \seq_new:N \g__sdaps_contexts_seq
69
70 % Global metadata write enable variable managed by context
71 \bool_new:N \g_sdaps_write_enable_bool
72 \bool_gset_false:N \g_sdaps_write_enable_bool
73
74 \cs_new_protected_nopar:Nn \_sdaps_context_to_tl:N
75 {
76   \tl_set:Nx #1 {\write_enable=\bool_if:NTF\g_sdaps_write_enable_bool{\c_true_bool}{\c_false_bo
77   \prop_map_inline:Nn \g__sdaps_current_context_prop {
78     % Could we remove some of the braces in the TL?
79     \tl_if_eq:nnTF {\undefined } { ##2 } {
80       \tl_put_right:Nn #1 {\,##1}}
81   } {
82     \tl_put_right:Nn #1 {\,##1}=##2}}
83   }
84 }
85 }
86
87
88 % Create new context using given identifier
89 \cs_new_protected_nopar:Nn \sdaps_context_begin:n
90 {
91   % We need to serialize the current context and save it away.
92
93   \group_begin:
94     % Serialize the current context
95     \_sdaps_context_to_tl:N \l__sdaps_tmpa_tl
96     \tl_gset:NV \g__sdaps_current_context_tl \l__sdaps_tmpa_tl
97     % Stuff it away in our sequence
98     \seq_gput_left:NV \g__sdaps_contexts_seq \g__sdaps_current_context_tl
99     \seq_gput_left:NV \g__sdaps_context_ids_seq \g__sdaps_current_context_id_tl
100
101     % Clear the hooks
102     \sdaps_context_put:nn { _context_hook_end } {}
103     \sdaps_context_put:nn { _context_hook_post_end } {}
104
105     \tl_gset:Nn \g__sdaps_current_context_id_tl { #1 }
106   \group_end:
107 }
108
109 \msg_new:nnn { sdapsbase } { context_end_none_left } { There ~ is ~ no ~ context ~ to ~ end ~ l
110 \msg_new:nnn { sdapsbase } { context_end_broken } { The ~ current ~ context ~ with ~ id ~ #1 ~
111
112 \cs_new_protected_nopar:Nn \__sdaps_context_end:
113 {
114   \seq_if_empty:NTF \g__sdaps_context_ids_seq {

```

```

115     \msg_error:nn { sdapsbase } { context_end_none_left }
116 } {
117
118     \group_begin:
119     \sdaps_context_get:nN { _context_hook_end } \l_tmpa_tl
120     \tl_if_eq:VnF \l_tmpa_tl { \q_no_value } { \tl_use:N \l_tmpa_tl }
121
122     % Grab post end hook
123     \sdaps_context_get:nN { _context_hook_post_end } \l_tmpa_tl
124
125     \sdaps_context_clear:
126     \seq_gpop_left:NN \g__sdaps_contexts_seq \g__sdaps_current_context_tl
127     \seq_gpop_left:NN \g__sdaps_context_ids_seq \l__sdaps_tmpa_tl
128
129     % Unpack context token list
130     \sdaps_context_set:V \g__sdaps_current_context_tl
131
132     \sdaps_context_get:nN { _write_enable } \l_tmpb_tl
133     \bool_gset:Nn \g_sdaps_write_enable_bool { \l_tmpb_tl }
134     \sdaps_context_remove:n { _write_enable }
135
136     \tl_gclear:N \g__sdaps_current_context_tl
137     \tl_gset:NV \g__sdaps_current_context_id_tl \l__sdaps_tmpa_tl
138
139     \tl_if_eq:VnF \l_tmpa_tl { \q_no_value } { \tl_use:N \l_tmpa_tl }
140     \group_end:
141 }
142 }
143
144 \bool_new:N \l__sdaps_tmp_bool
145
146 \cs_new_protected_nopar:Nn \__sdaps_test_context_id:n
147 {
148     \tl_if_eq:VnTF \g__sdaps_current_context_id_tl { #1 } {
149         \bool_set:Nn \l__sdaps_tmp_bool \c_true_bool
150     } {
151         \bool_set:Nn \l__sdaps_tmp_bool \c_false_bool
152     }
153 }
154
155 % Exit first context with passed in identifier
156 \cs_new_protected_nopar:Nn \sdaps_context_end:n
157 {
158     \__sdaps_test_context_id:n { #1 }
159
160     \bool_until_do:nn { \l__sdaps_tmp_bool } {
161         \sdaps_context_end:
162
163         \__sdaps_test_context_id:n { #1 }
164     }

```

```

165 \sdaps_context_end:
166 }
167
168 \cs_new_protected_nopar:Nn \__sdaps_context_end_local_scope:
169 {
170   \__sdaps_test_context_id:n { sdaps_local_scope }
171
172   \bool_until_do:nn { \l__sdaps_tmp_bool } {
173     \__sdaps_context_end:
174
175     \__sdaps_test_context_id:n { sdaps_local_scope }
176   }
177   \__sdaps_context_end:
178 }
179
180 % Exit current context
181 \cs_new_protected_nopar:Nn \sdaps_context_end:
182 {
183   % Ensure the current context is not a local group
184   \tl_if_eq:VnTF \g__sdaps_current_context_id_tl { sdaps_local_scope } {
185     \msg_error:nnV { sdapsbase } { context_end_broken } \g__sdaps_current_context_id_tl
186   } {}
187
188   \__sdaps_context_end:
189 }
190
191 % Create new context using an empty name
192 \cs_new_protected_nopar:Nn \sdaps_context_begin:
193 {
194   \sdaps_context_begin:n {}
195 }
196
197 \cs_new_protected_nopar:Nn \sdaps_context_begin_local:
198 {
199   % Create a new context which will automatically be destroyed at the end of
200   % the current TeX group.
201   \sdaps_context_begin:n { sdaps_local_scope }
202   \group_insert_after:N \__sdaps_context_end_local_scope:
203 }
204
205 \cs_new_protected_nopar:Nn \sdaps_context_put:n
206 {
207   \sdaps_context_put:nn { #1 } { \undefined }
208 }
209
210 \cs_new_protected_nopar:Nn \sdaps_context_remove:n
211 {
212   \prop_gremove:Nn \g__sdaps_current_context_prop { #1 }
213 }
214

```

```

215 \msg_new:nnn { sdapsbase } { context_key_broken } { Keys ~ may ~ not ~ contain ~ any ~ special
216 % Directly set a certain key
217 \cs_new_protected_nopar:Nn \sdaps_context_put:nn
218 {
219   % TODO: How can I ensure that {} are not contained?
220   % Though it would not be that bad actually.
221   \tl_if_in:nnTF {#1} {,} {
222     \msg_error:nnn { sdapsbase } { context_key_broken } {#1}
223   } {
224   }
225
226   \tl_if_in:nnTF {#1} {=} {
227     \msg_error:nnn { sdapsbase } { context_key_broken } {#1}
228   } {
229   }
230
231   \prop_gput:Nnn \g__sdaps_current_context_prop { #1 } { #2 }
232 }
233 \cs_generate_variant:Nn \sdaps_context_put:nn { nV }
234
235 % Set a set of keys using comma separated list of key/value pairs
236 \cs_new_protected_nopar:Nn \sdaps_context_set:n
237 {
238   \keyval_parse:NNn \sdaps_context_put:n \sdaps_context_put:nn { #1 }
239 }
240 \cs_generate_variant:Nn \sdaps_context_set:n { V }
241
242 \cs_new_protected_nopar:Nn \sdaps_context_get:nN
243 {
244   \prop_get:NnN \g__sdaps_current_context_prop { #1 } #2
245 }
246
247 \cs_new_protected_nopar:Nn \sdaps_context_gget:nN
248 {
249   \prop_get:NnN \g__sdaps_current_context_prop { #1 } \l_tmpa_tl
250   \tl_gset_eq:NN #2 \l_tmpa_tl
251 }
252
253 \cs_new_protected_nopar:Nn \sdaps_context_append:nnn
254 {
255   \sdaps_context_get:nN { #1 } \l__sdaps_tmpa_tl
256   \tl_if_eq:VnTF \l__sdaps_tmpa_tl { \q_no_value } {
257     \sdaps_context_put:nn { #1 } { #2 }
258   } {
259     \tl_put_right:Nn \l__sdaps_tmpa_tl { #3 }
260     \tl_put_right:Nn \l__sdaps_tmpa_tl { #2 }
261     \sdaps_context_put:nV { #1 } \l__sdaps_tmpa_tl
262   }
263 }
264 \cs_generate_variant:Nn \sdaps_context_append:nnn { nVn }

```

```

265
266
267 \cs_new_protected_nopar:Nn \sdaps_context_append:nn
268 {
269   \sdaps_context_append:nnn { #1 } { #2 } { , }
270 }
271
272 \cs_new_protected_nopar:Nn \sdaps_context_hook_end:n
273 {
274   \sdaps_context_append:nnn { _context_hook_end } { #1 } { }
275 }
276
277 \cs_new_protected_nopar:Nn \sdaps_context_hook_post_end:n
278 {
279   \sdaps_context_append:nnn { _context_hook_post_end } { #1 } { }
280 }
281
282 \cs_new_protected_nopar:Nn \sdaps_context_enable_writing:
283 {
284   \bool_gset_true:N \g_sdaps_write_enable_bool
285 }
286
287 \cs_new_protected_nopar:Nn \sdaps_context_disable_writing:
288 {
289   \bool_gset_false:N \g_sdaps_write_enable_bool
290 }
291
292 \cs_new_protected_nopar:Nn \sdaps_context_clear:
293 {
294   \prop_gclear:N \g__sdaps_current_context_prop
295 }
296
297 \cs_new:Nn \sdaps_context_map_function:N
298 {
299   \prop_map_function:NN \g__sdaps_current_context_prop #1
300 }
301
302 \cs_new_protected_nopar:Nn \sdaps_get:Nn
303 {
304   \prop_get:NnN \g__sdaps_current_context_prop { #2 } #1
305 }
306
307 \cs_new_protected_nopar:Nn \sdaps_get_empty:Nn
308 {
309   \prop_get:NnN \g__sdaps_current_context_prop { #2 } #1
310   \tl_if_eq:VnT #1 { \q_no_value } {
311     \tl_set:Nn #1 {}
312   }
313 }
314

```

```

315 \cs_new_protected_nopar:Nn \__sdaps_append_from_context:nN
316 {
317   \prop_get:NnN \g__sdaps_current_context_prop { #1 } \l__sdaps_tmpb_tl
318   \tl_if_eq:VnF \l__sdaps_tmpb_tl { \q_no_value } {
319     \tl_put_right:Nn #2 {,}
320     \tl_put_right:NV #2 {\l__sdaps_tmpb_tl}
321   }
322 }
323 \cs_generate_variant:Nn \__sdaps_append_from_context:nN { VN }
324
325

```

2.2 Defining Question and Headings

SDAPS needs to know about questions and headings/sections. Internally SDAPS uses the context management system to number these correctly and assign the variable names including the variable name concatenation automatically.

Note that the infrastructure present here will not prevent you from nesting questions, and SDAPS should actually handle this case just fine (ever wanted to put a question inside a textbox?).

It is important to keep these balanced. Please note that the SDAPS class does not use TeX groups here, so you could for example start a context inside a table and end it outside of it.

```

326
327 \seq_new:N \g__sdaps_object_id_seq
328 \int_new:N \g__sdaps_object_id_int
329 \int_gzero:N \g__sdaps_object_id_int
330
331 \cs_new_protected_nopar:Nn \sdaps_qobject_end_hook:
332 {
333   % Take the current implicit variable
334   \sdaps_context_get:nN { _implicit_var } \l__sdaps_tmpa_tl
335
336   % Prepend explicit variable name; we assume that either _implicit_var or _var
337   % have a proper value.
338   \sdaps_context_get:nN { _var } \l__sdaps_tmpb_tl
339   \tl_if_eq:VnF \l__sdaps_tmpb_tl { \q_no_value } {
340     \tl_if_eq:VnTF \l__sdaps_tmpa_tl { \q_no_value } {
341       \tl_clear:N \l__sdaps_tmpa_tl
342     } {
343       \tl_put_left:Nn \l__sdaps_tmpa_tl { _ }
344     }
345     \tl_put_left:NV \l__sdaps_tmpa_tl \l__sdaps_tmpb_tl
346   }
347
348   \sdaps_context_get:nN { id } \l__sdaps_tmpb_tl
349
350   \tl_if_eq:VnTF \l__sdaps_tmpb_tl { \q_no_value } {

```



```

351     \msg_warning:nn { sdapsbase } { no_qid }
352   } {
353     \sdaps_info_write_x:x{
354       Variable[\l__sdaps_tmpb_tl]=\l__sdaps_tmpa_tl
355     }
356   }
357 }
358
359 \cs_new_protected_nopar:Nn \sdaps_qobject_post_hook:
360 {
361   \seq_gpop_right:NN \g__sdaps_object_id_seq \l__sdaps_tmpa_tl
362   \int_gset:NV \g__sdaps_object_id_int \l__sdaps_tmpa_tl
363 }
364
365 \cs_new_protected_nopar:Nn \sdaps_qobject_begin:nnn
366 {
367   \int_gincr:N \g__sdaps_object_id_int
368   \seq_gput_right:NV \g__sdaps_object_id_seq \g__sdaps_object_id_int
369   \tl_set:Nx \l__sdaps_tmpa_tl { \int_use:N \g__sdaps_object_id_int }
370   \int_gzero:N \g__sdaps_object_id_int
371
372   \tl_set:Nx \l__sdaps_tmpb_tl { \seq_use:Nn \g__sdaps_object_id_seq {.} }
373
374   \sdaps_context_begin:n {#1}
375   \sdaps_context_put:nV {id} \l__sdaps_tmpb_tl
376   \sdaps_info_write:x {QObject-#2=\tl_use:N\l__sdaps_tmpb_tl. ~ \exp_not:n{#3}}
377   \sdaps_context_append:nVn { _implicit_var } \l__sdaps_tmpa_tl { _ }
378   \sdaps_context_hook_end:n { \sdaps_qobject_end_hook: }
379   \sdaps_context_hook_post_end:n { \sdaps_qobject_post_hook: }
380 }
381 \cs_generate_variant:Nn \sdaps_qobject_begin:nnn { nnV, nVV, nVn }
382
383
384 \cs_new_protected_nopar:Nn \sdaps_qobject_end:n
385 {
386   % End the context in question, everything else is done from the close hook
387   \sdaps_context_end:n {#1}
388 }
389
390 \cs_new_protected_nopar:Nn \sdaps_qobject_begin:nn
391 {
392   \sdaps_qobject_begin:nnn { unnamed_qobject } { #1 } { #2 }
393 }
394
395 \cs_new_protected_nopar:Nn \sdaps_qobject_begin_local:nn
396 {
397   % Empty local context which automatically closes the qobject
398   \sdaps_context_begin_local:
399   \sdaps_qobject_begin:nnn { unnamed_local_qobject } { #1 } { #2 }
400 }

```

```

401
402 \cs_new_protected_nopar:Nn \sdaps_qobject_end:
403 {
404   \sdaps_qobject_end:n { unnamed_qobject }
405 }
406
407 \cs_new_protected_nopar:Nn \sdaps_qobject_append_var:n
408 {
409   % If the given variable name starts with _ then include the implicitly
410   % generated variable name.
411   \tl_if_head_eq_charcode:nNTF { #1 } _ {
412     \sdaps_context_get:nN { _implicit_var } \l__sdaps_tmpa_tl
413     \tl_if_eq:VnF \l__sdaps_tmpa_tl { \q_no_value } {
414       \sdaps_context_append:nVn { _var } \l__sdaps_tmpa_tl { _ }
415     }
416
417     \sdaps_context_append:nnn { _var } { #1 } { }
418   } {
419     \sdaps_context_append:nnn { _var } { #1 } { _ }
420   }
421
422   % We have a proper variable name now, delete the implicit one
423   \sdaps_context_remove:n { _implicit_var }
424 }
425 \cs_generate_variant:Nn \sdaps_qobject_append_var:n { V }
426
427 \msg_new:nnn { sdapsbase } { no_qid } { Trying~to~output~metadata~but~no~question~ID~is~set~on~}
428
429 \cs_new_protected_nopar:Nn \sdaps_answer:n
430 {
431   \sdaps_context_get:nN { id } \l__sdaps_tmpa_tl
432
433   \tl_if_eq:VnTF \l__sdaps_tmpa_tl { \q_no_value } {
434     \msg_warning:nn { sdapsbase } { no_qid }
435   } {
436     \sdaps_info_write:x {
437       Answer[\tl_use:N \l__sdaps_tmpa_tl]=\exp_not:n { #1 }
438     }
439   }
440 }
441 \cs_generate_variant:Nn \sdaps_answer:n { o }
442 \cs_generate_variant:Nn \sdaps_answer:n { f }
443 \cs_generate_variant:Nn \sdaps_answer:n { V }
444
445 \cs_new_protected_nopar:Nn \sdaps_range:nnn
446 {
447   \sdaps_context_get:nN { id } \l__sdaps_tmpa_tl
448
449   \tl_if_eq:VnTF \l__sdaps_tmpa_tl { \q_no_value } {
450     \msg_warning:nn { sdapsbase } { no_qid }

```

```

451 } {
452   \sdaps_info_write:x {
453     Range-#1[\tl_use:N \l__sdaps_tmpa_tl]=\int_eval:n{#2},\exp_not:n { #3 }
454   }
455 }
456 }
457 \cs_generate_variant:Nn \sdaps_range:nnn { nno }
458 \cs_generate_variant:Nn \sdaps_range:nnn { nnf }
459 \cs_generate_variant:Nn \sdaps_range:nnn { nnV }
460
461 \cs_generate_variant:Nn \tl_if_head_eq_charcode:nNT { VN }
462
463 \cs_new_protected_nopar:Nn \sdaps_generate_var:nN
464 {
465   \tl_set:Nn \l__sdaps_tmpa_tl { #1 }
466
467   % Generate a variable name if there is none (prepended with _ prefix)
468   \tl_if_empty:VT \l__sdaps_tmpa_tl {
469     \tl_set:Nx \l__sdaps_tmpa_tl { _ \int_eval:n { \g__sdaps_object_id_int + 1 } }
470   }
471
472   % Prepend any implicitly generated variable names if prefixed by _
473   \tl_if_head_eq_charcode:VNT \l__sdaps_tmpa_tl _ {
474     \sdaps_context_get:nN { _implicit_var } \l__sdaps_tmpb_tl
475     \tl_if_eq:VnTF \l__sdaps_tmpb_tl { \q_no_value } {
476       \tl_remove_once:Nn \l__sdaps_tmpa_tl { _ }
477     } {
478       \tl_put_left:NV \l__sdaps_tmpa_tl \l__sdaps_tmpb_tl
479     }
480   }
481
482   % Prepend explicit variable name
483   \sdaps_context_get:nN { _var } \l__sdaps_tmpb_tl
484   \tl_if_eq:VnF \l__sdaps_tmpb_tl { \q_no_value } {
485     \tl_put_left:Nn \l__sdaps_tmpa_tl { _ }
486     \tl_put_left:NV \l__sdaps_tmpa_tl \l__sdaps_tmpb_tl
487   }
488
489   \tl_set:NV #2 \l__sdaps_tmpa_tl
490 }
491
492 \cs_new_protected_nopar:Nn \sdaps_box_inc_object_id:
493 {
494   \bool_if:NT \g_sdaps_write_enable_bool {
495     \int_gincr:N \g__sdaps_object_id_int
496   }
497 }
498

```

2.3 Data handling and override specification

Often it is useful to set certain flags for specific checkboxes. As the checkbox may only be generated internally by SDAPS it is impossible to pass a flag to it directly. Because of this `sdapsbase` has some mechanisms to maintain a tree with options.

```
\sdaps_overrides_init:n{*={  
  *={},  
  checkbox2={ellipse},  
  checkbox3,1={width=6mm},  
}}
```

```
499  
500 \cs_generate_variant:Nn \keyval_parse:NNn { NNV }  
501 \cs_generate_variant:Nn \int_gset:Nn { NV }  
502  
503 \seq_new:N \g__sdaps_checkbox_overlays_seq  
504 \seq_new:N \g__sdaps_textbox_overlays_seq  
505  
506  
507 \prop_new:N \g__sdaps_id_to_overrides_prop  
508 \prop_new:N \g__sdaps_overrides_prop  
509 \prop_new:N \g__sdaps_id_overrides_prop  
510  
511 \cs_new_protected_nopar:Nn \__sdaps_questionnaire_overrides_set:nn  
512 {  
513   \str_if_eq_x:nnTF { #1 } { * } {  
514     \__sdaps_parse_overrides:n{ #2 }  
515   } {  
516     \prop_put:Nnn \g__sdaps_id_to_overrides_prop { #1 } { #2 }  
517   }  
518 }  
519  
520 \cs_new_protected_nopar:Nn \sdaps_overrides_init:n  
521 {  
522   \keyval_parse:NNn \use_none:n \__sdaps_questionnaire_overrides_set:nn { #1 }  
523 }  
524  
525  
526 \cs_new_protected_nopar:Nn \__sdaps_overrides_set:nn  
527 {  
528   \prop_gput:Nnn \g__sdaps_overrides_prop { #1 } { #2 }  
529 }  
530  
531 \cs_new_protected_nopar:Nn \__sdaps_id_overrides_set:nn  
532 {  
533   \prop_gput:Nnn \g__sdaps_id_overrides_prop { #1 } { #2 }  
534 }  
535
```

```

536 \cs_new_protected_nopar:Nn \__sdaps_parse_overrides:n
537 {
538   \prop_gclear:N \g__sdaps_overrides_prop
539   \keyval_parse:NNn \use_none:n \__sdaps_overrides_set:nn { #1 }
540 }
541
542 \tl_new:N \l__sdaps_set_qid_tl
543 \cs_new_protected_nopar:Nn \sdaps_set_questionnaire_id:n
544 {
545   \tl_gset:Nn \g__sdaps_questionnaire_id_tl { #1 }
546   \prop_gclear:N \g__sdaps_id_overrides_prop
547   \prop_get:NnNT \g__sdaps_id_to_overrides_prop { #1 } \l__sdaps_set_qid_tl {
548     \keyval_parse:NNV \use_none:n \__sdaps_id_overrides_set:nn \l__sdaps_set_qid_tl
549   }
550 }
551 \cs_generate_variant:Nn \sdaps_set_questionnaire_id:n { V }
552
553
554 \cs_new_protected_nopar:Nn \__sdaps_append_override_options:Nnn
555 {
556   % Global definition
557   % First generic for all items
558   \prop_get:NnNT \g__sdaps_overrides_prop { * } \l__sdaps_tmpa_tl {
559     \tl_put_right:Nn #1 {,}
560     \tl_put_right:NV #1 \l__sdaps_tmpa_tl
561   }
562   \tl_if_empty:nF { #2 } {
563     % Items with same variable name
564     \prop_get:NnNT \g__sdaps_overrides_prop { #2 } \l__sdaps_tmpa_tl {
565       \tl_put_right:Nn #1 {,}
566       \tl_put_right:NV #1 \l__sdaps_tmpa_tl
567     }
568     \tl_if_empty:nF { #3 } {
569       % Items with same variable name and value
570       \prop_get:NnNT \g__sdaps_overrides_prop { #2&#3 } \l__sdaps_tmpa_tl {
571         \tl_put_right:Nn #1 {,}
572         \tl_put_right:NV #1 \l__sdaps_tmpa_tl
573       }
574     }
575   }
576
577
578   % Local (questionnaire ID specific) definition
579   % First generic for all items
580   \prop_get:NnNT \g__sdaps_id_overrides_prop { * } \l__sdaps_tmpa_tl {
581     \tl_put_right:Nn #1 {,}
582     \tl_put_right:NV #1 \l__sdaps_tmpa_tl
583   }
584   \tl_if_empty:nF { #2 } {
585     % Items with same variable name

```

```

586 \prop_get:NnNT \g__sdaps_id_overrides_prop { #2 } \l__sdaps_tmpa_tl {
587   \tl_put_right:Nn #1 {,}
588   \tl_put_right:NV #1 \l__sdaps_tmpa_tl
589 }
590 \tl_if_empty:nF { #3 } {
591   % Items with same variable name and value
592   \prop_get:NnNT \g__sdaps_id_overrides_prop { #2&#3 } \l__sdaps_tmpa_tl {
593     \tl_put_right:Nn #1 {,}
594     \tl_put_right:NV #1 \l__sdaps_tmpa_tl
595   }
596 }
597 }
598 }
599 \cs_generate_variant:Nn \__sdaps_append_override_options:Nnn { NVn }
600

```

First we define constants and global variables for later use.

```

601 \dim_new:N \g_sdaps_linewidth_dim
602 \g_sdaps_linewidth_dim=1bp
603 \tl_new:N \g__sdaps_checkbox_last_info_tl
604
605 \int_new:N \g__sdaps_textbox_num_int
606 \int_set:Nn \g__sdaps_textbox_num_int 0
607
608 \tl_new:N \l_sdaps_var_tl
609 \dim_new:N \l_sdaps_x_dim
610 \dim_new:N \l_sdaps_y_dim
611 \dim_new:N \l_sdaps_width_dim
612 \dim_new:N \l_sdaps_height_dim
613

```

We need to be able to output data into the .sdaps file. On startup the output is opened but writing is disabled. Note that the write enabled variable is managed by the context.

```

614
615 \iow_new:N \g_sdaps_infofile_iow
616 \iow_open:Nn \g_sdaps_infofile_iow { \c_sys_jobname_str . sdaps }
617 \int_new:N \g__sdaps_infofile_line_int
618 \int_gset:Nn \g__sdaps_infofile_line_int { 0 }
619
620 \cs_new_protected_nopar:Nn \sdaps_info_write:n
621 {
622   \bool_if:NT \g_sdaps_write_enable_bool {
623     \int_gincr:N \g__sdaps_infofile_line_int
624     \iow_shipout:Nx \g_sdaps_infofile_iow { [ \int_use:N \g__sdaps_infofile_line_int ] \exp_not
625   }
626 }
627 \cs_generate_variant:Nn \sdaps_info_write:n { x }
628
629 \cs_new_protected_nopar:Nn \sdaps_info_write:x:n

```

```

630 {
631   \bool_if:NT \g_sdaps_write_enable_bool {
632     \int_gincr:N \g_sdaps_infofile_line_int
633     \iow_shipout_x:Nx \g_sdaps_infofile_iow { [ \int_use:N \g_sdaps_infofile_line_int ] \exp_n
634   }
635 }
636 \cs_generate_variant:Nn \sdaps_info_write_x:n { x }
637
638

```

Set some options at the beginning of the document.

```

639 %\AtBeginDocument{}

```

2.4 Definition for keyword parameters

```

640
641
642
643 \dim_new:N \l_sdaps_checkbox_linewidth_dim
644 \dim_new:N \l_sdaps_checkbox_width_dim
645 \dim_new:N \l_sdaps_checkbox_height_dim
646 \tl_new:N \l_sdaps_checkbox_form_tl
647 \tl_new:N \l_sdaps_checkbox_fill_tl
648 \tl_new:N \l_sdaps_checkbox_var_tl
649 \tl_new:N \l_sdaps_checkbox_value_tl
650 \bool_new:N \l_sdaps_checkbox_draw_check_bool
651
652 \tl_set:Nn \l_sdaps_checkbox_form_tl { box }
653
654 \tl_new:N \l_sdaps_parse_unknown_tl
655
656
657 % Internal overlays
658 \tl_new:N \l_sdaps_overlay_centered_text_tl
659 \tl_new:N \l_sdaps_overlay_minipage_text_tl
660 \tl_new:N \l_sdaps_overlay_minipage_pos_tl
661 \dim_new:N \l_sdaps_overlay_minipage_pad_dim
662
663 % Note that width/height is the *outside* width/height
664 \keys_define:nn { sdaps / checkbox }
665 {
666   linewidth      .dim_set:N    = \l_sdaps_checkbox_linewidth_dim,
667   linewidth      .initial:n     = 1bp,
668   width          .dim_set:N    = \l_sdaps_checkbox_width_dim,
669   width          .initial:n     = 3.5mm,
670   height         .dim_set:N    = \l_sdaps_checkbox_height_dim,
671   height         .initial:n     = 3.5mm,
672   form           .choices:nn    = { box, ellipse } { \tl_set:Nx \l_sdaps_checkbox_form_tl { \l_keys
673   value          .tl_set:N      = \l_sdaps_checkbox_value_tl,
674

```

```

675 fill          .tl_set:N    = \l_sdaps_checkbox_fill_tl,
676 fill          .initial:n   = { white },
677
678 draw_check     .bool_set:N  = \l_sdaps_checkbox_draw_check_bool,
679 draw_check     .default:n   = true,
680 draw_check     .initial:n   = false,
681
682 % Simple node overlay
683 centered_text  .tl_set:N    = \l_sdaps_overlay_centered_text_tl,
684 centered_text  .initial:n   = {},
685
686 % minipage overlay
687 text           .tl_set:N    = \l_sdaps_overlay_minipage_text_tl,
688 text           .initial:n   = {},
689 text_align     .tl_set:N    = \l_sdaps_overlay_minipage_pos_tl,
690 text_align     .initial:n   = {c},
691 text_padding   .dim_set:N   = \l_sdaps_overlay_minipage_pad_dim,
692 text_padding   .initial:n   = {2bp},
693
694 ellipse        .meta:n      = { form=ellipse },
695 box            .meta:n      = { form=box },
696 }

```

2.5 Checkboxes

```

697
698 \cs_new_protected_nopar:Nn \__sdaps_checkbox_internal:nn
699 {
700   \mbox{
701     \sdaps_if_rtl:T {\beginL}
702     \pdfsavepos
703
704     % Position of page and baseline offset
705     \dim_set:Nn \l_sdaps_x_dim { \hoffset }
706     \dim_set:Nn \l_sdaps_y_dim { \voffset + \l_sdaps_checkbox_height_dim - \dim_eval:n { 0.5\l_
707
708     % Size
709     \dim_set:Nn \l_sdaps_width_dim { \l_sdaps_checkbox_width_dim }
710     \dim_set:Nn \l_sdaps_height_dim { \l_sdaps_checkbox_height_dim }
711
712     \bool_if:NT \g_sdaps_write_enable_bool {
713       % pdflast[xy]pos is the PDF position of the baseline at the start of the box
714       % excluding the page origin offset.
715       \sdaps_context_get:nN {id} \l__sdaps_tmpa_tl
716       \tl_if_eq:VnTF \l__sdaps_tmpa_tl { \q_no_value } {
717         \msg_warning:nn { sdapsbase } { no_qid }
718       } {
719         \sdaps_info_write_x:x{
720           Box[\l__sdaps_tmpa_tl]=Checkbox,
721           \exp_not:n{\int_use:N\g_sdaps_page_int},

```



```

722         \exp_not:n{\dim_eval:n} { \exp_not:f {\dim_use:N \l_sdaps_x_dim + \the\pdflastxpos sp
723         \exp_not:n{\dim_eval:n} { \exp_not:f {\dim_use:N \l_sdaps_y_dim + \the\pdflastypos sp
724         \dim_use:N \l_sdaps_width_dim,
725         \dim_use:N \l_sdaps_height_dim,
726         \tl_to_str:N\l_sdaps_checkbox_form_tl,
727         \dim_use:N \l_sdaps_checkbox_linewidth_dim,
728         #1,\tl_if_empty:nTF { #2 } { \int_use:N \g__sdaps_object_id_int } { #2 }
729     }
730 }
731 }
732
733
734 \tikz[baseline={0.5\l_sdaps_checkbox_height_dim-0.8ex}]{%
735     \tl_if_eq:VnT \l_sdaps_checkbox_form_tl { box } {
736         \draw[line~width=\l_sdaps_checkbox_linewidth_dim,fill=\l_sdaps_checkbox_fill_tl] (0.5\l
737     }
738     \tl_if_eq:VnT \l_sdaps_checkbox_form_tl { ellipse } {
739         \draw[line~width=\l_sdaps_checkbox_linewidth_dim,fill=\l_sdaps_checkbox_fill_tl] (0.5\l
740     }
741
742     % For the overlay we actually position the nodes relative to the checkbox
743     % and not absolute on the page.
744     \dim_set:Nn \l_sdaps_x_dim { 0pt }
745     \dim_set:Nn \l_sdaps_y_dim { \l_sdaps_checkbox_height_dim }
746
747     % Use overlay so that nothing happens if a node is larger than the checkbox
748     \begin{scope}[overlay]
749         \seq_map_inline:Nn \g__sdaps_checkbox_overlays_seq {##1}
750     \end{scope}
751 }
752 \sdaps_if_rtl:T {\endL}
753 }
754 }
755 \cs_generate_variant:Nn \__sdaps_checkbox_internal:nn { Vn }
756
757 \sdaps_context_set:n { checkboxtype=multichoice }
758 \cs_new_protected_nopar:Nn \sdaps_checkbox_set_type:n
759 {
760     \sdaps_context_set:n { checkboxtype={#1} }
761 }
762 \cs_generate_variant:Nn \sdaps_checkbox_set_type:n { V }
763
764 \cs_new_protected_nopar:Nn \sdaps_checkbox:nn
765 {
766     \group_begin:%
767
768     \sdaps_generate_var:nN { #1 } \l_sdaps_var_tl
769
770     \sdaps_context_get:nN { checkboxtype } \l__sdaps_tmpa_tl
771     \tl_if_eq:VnTF \l__sdaps_tmpa_tl { multichoice } {

```

```

772     \tl_set:Nn \l_sdaps_parse_unknown_tl { box }
773   } {
774     \tl_set:Nn \l_sdaps_parse_unknown_tl { ellipse }
775   }
776
777   \__sdaps_append_from_context:nN { * } \l_sdaps_parse_unknown_tl
778   \__sdaps_append_from_context:VN \l__sdaps_tmpa_tl \l_sdaps_parse_unknown_tl
779   \__sdaps_append_override_options:NVn \l_sdaps_parse_unknown_tl \l_sdaps_var_tl { #2 }
780
781   \keys_set_known:nVN { sdaps / checkbox } \l_sdaps_parse_unknown_tl \l_sdaps_parse_unknown_t
782
783   \sdaps_box_inc_object_id:
784
785   \__sdaps_checkbox_internal:Vn \l_sdaps_var_tl { #2 }
786   \group_end:%
787   \ignorespaces
788 }
789 \cs_generate_variant:Nn \sdaps_checkbox:nn { Vn, VV, nV }
790
791
792 \cs_new_protected_nopar:Nn \sdaps_overlay_check:
793 {
794   \bool_if:NT \l_sdaps_checkbox_draw_check_bool {
795     \begin{scope}[decoration={random~steps,segment~length=4pt,amplitude=1pt}]
796       \draw[line~width=\l_sdaps_checkbox_linewidth_dim, decorate] ($(0, 0) - (2pt,2pt)$) -- (0.
797       \draw[line~width=\l_sdaps_checkbox_linewidth_dim, decorate] ($(0, \l_sdaps_checkbox_heigh
798     \end{scope}
799   }
800 }
801 \seq_put_left:Nn \g__sdaps_checkbox_overlays_seq \sdaps_overlay_check:
802
803
804 \cs_new_protected_nopar:Nn \sdaps_overlay_centered:
805 {
806   \tl_if_empty:NF \l_sdaps_overlay_centered_text_tl {
807     \node[anchor=center,inner~sep=0pt,outer~sep=0pt] at ($(\l_sdaps_x_dim, \l_sdaps_y_dim) + 0.
808     \l_sdaps_overlay_centered_text_tl
809   };
810 }
811 }
812 \seq_put_left:Nn \g__sdaps_checkbox_overlays_seq \sdaps_overlay_centered:
813 \seq_put_left:Nn \g__sdaps_textbox_overlays_seq \sdaps_overlay_centered:
814
815
816 \cs_new_protected_nopar:Nn \sdaps_overlay_minipage:
817 {
818   \tl_if_empty:NF \l_sdaps_overlay_minipage_text_tl {
819     \node[anchor=center,inner~sep=0pt,outer~sep=0pt] at ($(\l_sdaps_x_dim, \l_sdaps_y_dim) + 0.
820     \dim_set:Nn \l_sdaps_width_dim { \l_sdaps_width_dim - 2\l_sdaps_overlay_minipage_pad_dim
821     \dim_set:Nn \l_sdaps_height_dim { \l_sdaps_height_dim - 2\l_sdaps_overlay_minipage_pad_di

```

```

822
823     \begin{minipage}[t][\l_sdaps_height_dim][\l_sdaps_overlay_minipage_pos_tl]{\l_sdaps_width
824         % Hm, is this sane?
825         \tex_let:D \textheight\l_sdaps_height_dim
826         \l_sdaps_overlay_minipage_text_tl
827     \end{minipage}
828 };
829 }
830 }
831 \seq_put_left:Nn \g__sdaps_checkbox_overlays_seq \sdaps_overlay_minipage:
832 \seq_put_left:Nn \g__sdaps_textbox_overlays_seq \sdaps_overlay_minipage:
833
834

```

2.6 Textboxes

```

835
836 \sdaps_context_set:n { textboxtype=textbox }
837 \cs_new_protected_nopar:Nn \sdaps_textbox_set_type:n
838 {
839     \sdaps_context_set:n { textboxtype={#1} }
840 }
841 \cs_generate_variant:Nn \sdaps_textbox_set_type:n { V }
842
843 \dim_new:N \l_sdaps_textbox_linewidth_dim
844 \tl_new:N \l_sdaps_textbox_var_tl
845 \tl_new:N \l_sdaps_textbox_fill_tl
846 \tl_new:N \l__sdaps_textbox_boxtype_tl
847
848
849 \keys_define:nn { sdaps / textbox }
850 {
851     linewidth .dim_set:N = \l_sdaps_textbox_linewidth_dim,
852     linewidth .initial:n = 1bp,
853
854     fill .tl_set:N = \l_sdaps_textbox_fill_tl,
855     fill .initial:n = { white },
856
857     % Simple node overlay
858     centered_text .tl_set:N = \l_sdaps_overlay_centered_text_tl,
859     centered_text .initial:n = {},
860
861     % minipage overlay
862     text .tl_set:N = \l_sdaps_overlay_minipage_text_tl,
863     text .initial:n = {},
864     text_align .tl_set:N = \l_sdaps_overlay_minipage_pos_tl,
865     text_align .initial:n = {c},
866     text_padding .dim_set:N = \l_sdaps_overlay_minipage_pad_dim,
867     text_padding .initial:n = {2bp},
868 }

```

```

869
870
871
872
873 \dim_new:N \l__sdaps_textbox_dp_dim
874 \dim_new:N \l__sdaps_textbox_ht_dim
875 \dim_new:N \l__sdaps_textbox_wd_dim
876 \dim_new:N \l__sdaps_textbox_pad_dim
877
878
879 \msg_new:nnn { sdapsbase } { textbox_wrong_mode } { Impossible~to~layout~a~#1~textbox~in~#2~mod
880
881 \cs_new_protected_nopar:Nn \__sdaps_textbox_prepare:n
882 {
883   \tl_set:Nn \l_sdaps_parse_unknown_tl {}
884
885   \sdaps_generate_var:nN { #1 } \l_sdaps_var_tl
886
887   \sdaps_context_get:nN { textboxtype } \l__sdaps_tmpa_tl
888   \tl_if_eq:VnTF \l__sdaps_tmpa_tl { codebox } {
889     \tl_set:Nn \l__sdaps_textbox_boxttype_tl { Codebox }
890   } {
891     \tl_set:Nn \l__sdaps_textbox_boxttype_tl { Textbox }
892   }
893
894   \__sdaps_append_from_context:nN { * } \l_sdaps_parse_unknown_tl
895   \__sdaps_append_from_context:VN \l__sdaps_tmpa_tl \l_sdaps_parse_unknown_tl
896   \__sdaps_append_override_options:NvN \l_sdaps_parse_unknown_tl \l_sdaps_var_tl { }
897
898   \keys_set_known:nVN { sdaps / textbox } \l_sdaps_parse_unknown_tl \l_sdaps_parse_unknown_tl
899
900   \sdaps_box_inc_object_id:
901 }
902
903 \cs_new_protected_nopar:Nn \sdaps_textbox_vhstretch:nnn
904 {

```

At first we need to ensure that we are not in math mode. We also error out if switching into vertical mode (by inserting a paragraph break) fails.

The scaling feature will likely fail in inner vertical mode, but it is still permissible.

```

905   \if_mode_math:
906     \msg_error:nnnn { sdapsbase } { textbox_wrong_mode } { vhstretch } { math }
907   \else:
908     \tex_par:D
909     \if_mode_horizontal:
910       \msg_error:nnnn { sdapsbase } { textbox_wrong_mode } { vhstretch } { horizontal }
911     \else:
912       % Go into vertical mode by ending the current paragraph and insert
913       % \cs{widowpenalty} so that we don't get an unwelcome page break. It is

```

```

914 % assumed that any explanation for the textbox will be on top.
915 \tex_penalty:D \tex_widowpenalty:D
916
917 \group_begin:%
918
919 \__sdaps_textbox_prepare:n { #1 }
920

```

First we define the height and depth of the box that will be set. Setting the height works by inserting a rule into the first box and adding a negative skip afterwards. The depth is compensated by simply doing the same thing at the bottom.

Note that we need to add nobreak/nointerlineskip everywhere so that we don't insert extra spacing and no page break will happen.

The first box has a defined height and stores the current position for the frame code later on. Then two cleaders follow which simply implement the required stretching (cleaders are used to prevent page breaking, I do not know whether this is sane, but it seems to work just fine). After this the box containing the main drawing code is placed. We simply use a vbox with right aligned content. The last thing happening is a vskip plus hbox to set the correct depth.

```

921 % TODO: Make this configurable
922 \dim_set:Nn \l__sdaps_textbox_dp_dim { 0.5ex }
923 \dim_set:Nn \l__sdaps_textbox_ht_dim { 1.7ex }
924
925 \vbox:n {
926   \sdaps_if_rtl:T {\beginL}
927   \leftskip=0pt
928   \rightskip=0pt plus 1fill
929   \noindent
930   \tex_vrule:D height \l__sdaps_textbox_ht_dim depth 0pt width 0pt
931   \pgfsys@markposition{textboxtop\int_use:N\g__sdaps_textbox_num_int}
932   \sdaps_if_rtl:T {\endL}
933 }
934
935 \tex_penalty:D 10000
936 \nointerlineskip
937
938 \tex_cleaders:D\tex_hbox:D{\skip_vertical:n{#2 - \l__sdaps_textbox_ht_dim}}
939
940 \tex_penalty:D 10000
941 \nointerlineskip
942
943 \tex_cleaders:D\tex_hbox:D{\skip_vertical:n{\stretch{#3}}}
944
945 \tex_penalty:D 10000
946 \nointerlineskip
947
948 \vbox:n {
949   \sdaps_if_rtl:T {\beginL}

```

```

950 \noindent
951 \leftskip=0pt plus 1fill
952 \rightskip=0pt
953 \begin{tikzpicture}[remember~picture,overlay,shift=(current~page.south~west)]
954 \pgfsys@getposition{pgfp@pageorigin}{\@sdaps@pageorigin}
955 \pgfsys@getposition{textboxtop\int_use:N\g__sdaps_textbox_num_int}{\@sdaps@textboxtoppos}
956 \pgfpointadd{\@sdaps@textboxtoppos}{
957 \pgfpointadd{\@sdaps@pageorigin}{\pgfpoint{0}{\l__sdaps_textbox_ht_dim}}
958 }
959 \pgfgetlastxy{\l__sdaps_x}{\l__sdaps_y}
960 \dim_set:Nn \l_sdaps_x_dim {\l__sdaps_x}
961 \dim_set:Nn \l_sdaps_y_dim {\l__sdaps_y}
962
963 \pgfsys@getposition{pgfp@pictureid}{\@sdaps@textboxbottompos}
964 \pgfpointdiff{\@sdaps@textboxtoppos}{\@sdaps@textboxbottompos}
965
966 % Is there a more elegant way to multiply the height with -1?
967 \pgfgetlastxy{\l__sdaps_width}{\l__sdaps_height}
968 % Add the minimum height (i.e. depth below the last baseline) to the
969 % overall height.
970 \pgfpoint{\l__sdaps_width}{-\l__sdaps_height + \l__sdaps_textbox_ht_dim}
971 \pgfgetlastxy{\l__sdaps_width}{\l__sdaps_height}
972 \dim_set:Nn \l_sdaps_width_dim {\l__sdaps_width}
973 \dim_set:Nn \l_sdaps_height_dim {\l__sdaps_height}
974
975 % Draw the rectangle
976 \draw[line~width=\l_sdaps_textbox_linewidth_dim,fill=\l_sdaps_textbox_fill_tl] ($(\l_sdaps_
977
978 \begin{scope}
979 \seq_map_inline:Nn \g__sdaps_textbox_overlays_seq {##1}
980 \end{scope}
981
982 \bool_if:NT \g_sdaps_write_enable_bool {
983 \sdaps_context_get:nN {id} \l__sdaps_tmpa_tl
984 \tl_if_eq:VnTF \l__sdaps_tmpa_tl { \q_no_value } {
985 \msg_warning:nn { sdapsbase } { no_qid }
986 } {
987 \sdaps_info_write_x:x {
988 Box[\l__sdaps_tmpa_tl]=\l_sdaps_textbox_boxttype_tl,
989 \exp_not:n{\int_use:N\g_sdaps_page_int},
990 \dim_use:N \l_sdaps_x_dim,
991 \dim_use:N \l_sdaps_y_dim,
992 \dim_use:N \l_sdaps_width_dim,
993 \dim_use:N \l_sdaps_height_dim,
994 \dim_use:N \l_sdaps_textbox_linewidth_dim,
995 \tl_use:N \l_sdaps_var_tl,
996 }
997 }
998 }
999 \end{tikzpicture}

```

```

1000     \sdaps_if_rtl:T {\endL}
1001   }
1002
1003   % We are done here, but we need the correct depth, so skip around a bit
1004   \tex_penalty:D 10000
1005   \nointerlineskip
1006
1007   \skip_vertical:n{ - \l__sdaps_textbox_dp_dim}
1008
1009   \tex_penalty:D 10000
1010   \nointerlineskip
1011
1012   \hbox:n { \vrule depth \l__sdaps_textbox_dp_dim height Opt width Opt }
1013
1014   \int_gincr:N\g__sdaps_textbox_num_int
1015
1016   \group_end:
1017   \fi:
1018   \fi:
1019 }
1020 \cs_generate_variant:Nn \sdaps_textbox_vhstretch:nnn { Vnn }
1021
1022 \cs_new_protected_nopar:Nn \sdaps_textbox_vhstretch:nn
1023 {
1024   \sdaps_textbox_vhstretch:nnn { #1 } { #2 } { 1 }
1025 }
1026
1027
1028 \cs_new_protected_nopar:Nn \sdaps_textbox_hstretch:nnnnn
1029 {
1030   \group_begin:
1031     \sdaps_if_rtl:T {\beginL}
1032
1033     \__sdaps_textbox_prepare:n { #1 }
1034
1035     \dim_set:Nn \l_tmpa_dim { #2 }
1036     \dim_set:Nn \l_tmpb_dim { #3 }
1037
1038     % Place a vrule to make space for the top/bottom padding
1039     \tex_vrule:D depth \dim_use:N \l_tmpa_dim height \dim_use:N \l_tmpb_dim width Opt \nobreak
1040     \pgfsys@markposition{textboxstart\int_use:N\g__sdaps_textbox_num_int} \nobreak
1041
1042     \skip_horizontal:n { #4 + \stretch{#5} } \nobreak
1043
1044     % The textbox (rendered on the background)
1045     \begin{tikzpicture}[remember~picture,overlay,shift=(current~page.south~west)]
1046       \pgfsys@getposition{pgfpagelocation}{\@sdaps@pagelocation}
1047       \pgfsys@getposition{textboxstart\int_use:N\g__sdaps_textbox_num_int}{\@sdaps@textboxpos}
1048       % The position here is the position of the baseline.
1049       % So move up by height (param 2) to get the correct vertical position.

```

```

1050 \pgfpointadd{\@sdaps@textboxpos}{
1051 \pgfpointadd{\@sdaps@pageorigin}{\pgfpoint{0}{ \dim_use:N \l_tmpb_dim}}
1052 }
1053 \pgfgetlastxy{\l__sdaps_x}{\l__sdaps_y}
1054 \dim_set:Nn \l_sdaps_x_dim {\l__sdaps_x}
1055 \dim_set:Nn \l_sdaps_y_dim {\l__sdaps_y}
1056
1057 \pgfsys@getposition{\pgfpictureid}{\@sdaps@textboxendpos}
1058 % Calculate width and add the height to it
1059 \pgfpointadd{
1060 \pgfpointdiff{\@sdaps@textboxpos}{\@sdaps@textboxendpos}
1061 }{\pgfpoint{0pt}{\dim_use:N \l_tmpa_dim + \dim_use:N \l_tmpb_dim}}
1062 \pgfgetlastxy{\l__sdaps_width}{\l__sdaps_height}
1063 \dim_set:Nn \l_sdaps_width_dim {\l__sdaps_width}
1064 \dim_set:Nn \l_sdaps_height_dim {\l__sdaps_height}
1065
1066 % Draw the rectangle
1067 \draw[line~width=\l_sdaps_textbox_linewidth_dim,fill=\l_sdaps_textbox_fill_tl] ($(\l_sdap
1068
1069 \begin{scope}
1070 \seq_map_inline:Nn \g__sdaps_textbox_overlays_seq {##1}
1071 \end{scope}
1072
1073 \bool_if:NT \g_sdaps_write_enable_bool {
1074 \sdaps_context_get:nN {id} \l__sdaps_tmpa_tl
1075 \tl_if_eq:VnTF \l__sdaps_tmpa_tl { \q_no_value } {
1076 \msg_warning:nn { sdapsbase } { no_qid }
1077 } {
1078 \sdaps_info_write_x:x {
1079 Box[\l__sdaps_tmpa_tl]=\l__sdaps_textbox_boxttype_tl,
1080 \exp_not:n{\int_use:N\g_sdaps_page_int},
1081 \dim_use:N \l_sdaps_x_dim,
1082 \dim_use:N \l_sdaps_y_dim,
1083 \dim_use:N \l_sdaps_width_dim,
1084 \dim_use:N \l_sdaps_height_dim,
1085 \dim_use:N \l_sdaps_textbox_linewidth_dim,
1086 \tl_use:N \l_sdaps_var_tl,
1087 }
1088 }
1089 }
1090 \end{tikzpicture}
1091
1092 \int_gincr:N\g__sdaps_textbox_num_int
1093
1094 \sdaps_if_rtl:T {\endL}
1095 \group_end:
1096 }
1097
1098 \cs_new_protected_nopar:Nn \__sdaps_textbox_prepare_coffin:
1099 {

```



```

1100 \dim_set:Nn \l__sdaps_textbox_ht_dim { \coffin_ht:N \l__sdaps_textbox_coffin }
1101 \dim_set:Nn \l__sdaps_textbox_dp_dim { \coffin_dp:N \l__sdaps_textbox_coffin }
1102 \dim_set:Nn \l__sdaps_textbox_wd_dim { \coffin_wd:N \l__sdaps_textbox_coffin }
1103
1104 \hcoffin_set:Nn \l_tmpa_coffin {
1105   \tex_vrule:D depth Opt height \dim_eval:n { \l__sdaps_textbox_ht_dim + \l__sdaps_textbox_pa
1106
1107   \pdfsavepos
1108
1109   \dim_set:Nn \l_sdaps_width_dim { \l__sdaps_textbox_wd_dim + 2 \l__sdaps_textbox_pad_dim }
1110   \dim_set:Nn \l_sdaps_height_dim { \l__sdaps_textbox_ht_dim + \l__sdaps_textbox_dp_dim + 2 \l_
1111
1112   % pdflast[xy]pos is the PDF position of the top left corner excluding the
1113   % origin
1114   \bool_if:NT \g_sdaps_write_enable_bool {
1115     \sdaps_context_get:nN {id} \l__sdaps_tmpa_tl
1116     \tl_if_eq:VnTF \l__sdaps_tmpa_tl { \q_no_value } {
1117       \msg_warning:nn { sdapsbase } { no_qid }
1118     } {
1119       \sdaps_info_write_x:x {
1120         Box[\l__sdaps_tmpa_tl]=\l__sdaps_textbox_boxttype_tl,
1121         \exp_not:n{\int_use:N \g_sdaps_page_int},
1122         \exp_not:n{\dim_eval:n { \hoffset + \the\pdflastxpos sp }},
1123         \exp_not:n{\dim_eval:n} { \exp_not:n { \voffset + \the\pdflastypos sp } + \dim_use:N \l_
1124         \dim_use:N \l_sdaps_width_dim,
1125         \dim_use:N \l_sdaps_height_dim,
1126         \dim_use:N \l_sdaps_textbox_linewidth_dim,
1127         \tl_use:N \l_sdaps_var_tl,
1128       }
1129     }
1130   }
1131
1132   \dim_set:Nn \l_sdaps_x_dim { Opt }
1133   \dim_set:Nn \l_sdaps_y_dim { \l__sdaps_textbox_ht_dim + \l__sdaps_textbox_pad_dim }
1134
1135   % The textbox (rendered on the background)
1136   \begin{tikzpicture}[overlay]
1137     % Draw the rectangle
1138     \draw[line~width=\l_sdaps_textbox_linewidth_dim,fill=\l_sdaps_textbox_fill_tl] ($(\l_sdap
1139
1140     \begin{scope}
1141       \seq_map_inline:Nn \g__sdaps_textbox_overlays_seq {##1}%
1142     \end{scope}
1143   \end{tikzpicture}
1144   \skip_horizontal:n { \l__sdaps_textbox_pad_dim }
1145 }
1146
1147 \hcoffin_set:Nn \l_tmpb_coffin {
1148   \skip_horizontal:n { \l__sdaps_textbox_pad_dim }
1149   \tex_vrule:D depth \l__sdaps_textbox_pad_dim height Opt width Opt

```

```

1150 }
1151
1152 \coffin_join:NnnNnnnn \l_tmpa_coffin { r } { H } \l__sdaps_textbox_coffin { l } { H } { Opt }
1153 \coffin_join:NnnNnnnn \l_tmpa_coffin { r } { b } \l_tmpb_coffin { l } { t } { Opt } { Opt }
1154
1155 \coffin_set_eq:NN \l__sdaps_textbox_coffin \l_tmpa_coffin
1156
1157
1158 \int_gincr:N\g__sdaps_textbox_num_int
1159 }
1160
1161 \coffin_new:N \l__sdaps_textbox_coffin
1162 \cs_new_protected_nopar:Nn \sdaps_textbox_hbox:nnn
1163 {
1164   \group_begin:
1165
1166     \__sdaps_textbox_prepare:n { #1 }
1167
1168     \hcoffin_set:Nn \l__sdaps_textbox_coffin { \tl_trim_spaces:n { #3 } }
1169
1170     \dim_set:Nn \l__sdaps_textbox_pad_dim { #2 }
1171
1172     \__sdaps_textbox_prepare_coffin:
1173
1174     \coffin_typeset:Nnnnn \l__sdaps_textbox_coffin { l } { H } { Opt } { Opt }
1175
1176   \group_end:
1177 }
1178
1179 \cs_new_protected:Nn \sdaps_textbox_vbox:nnnn
1180 {
1181   \group_begin:
1182
1183     \__sdaps_textbox_prepare:n { #1 }
1184
1185     \dim_set:Nn \l__sdaps_textbox_pad_dim { #3 }
1186     \dim_set:Nn \l__sdaps_textbox_wd_dim { #2 - 2\l__sdaps_textbox_pad_dim }
1187
1188     \vcoffin_set:Nnn \l__sdaps_textbox_coffin { \l__sdaps_textbox_wd_dim } { \tl_trim_spaces:n
1189
1190     \__sdaps_textbox_prepare_coffin:
1191
1192     \coffin_typeset:Nnnnn \l__sdaps_textbox_coffin { l } { H } { Opt } { Opt }
1193
1194   \group_end:
1195 }
1196

```

2.7 Page Marking

SDAPS depends on certain markings on every page. The following function implement drawing these marks.

```
1197
1198 \int_new:N \g_sdaps_page_int
1199 \int_set:Nn \g_sdaps_page_int { 0 }
1200
1201 % Disabling recognition disables all drawings related to automatic recognition
1202 \bool_new:N \g_sdaps_recognition_bool
1203 \bool_new:N \g_sdaps_draft_bool
1204 \bool_new:N \g_sdaps_twoside_bool
1205 \bool_new:N \g_sdaps_print_questionnaire_id_bool
1206
1207 \bool_set:Nn \g_sdaps_recognition_bool \c_true_bool
1208 \bool_set:Nn \g_sdaps_draft_bool \c_true_bool
1209 \bool_set:Nn \g_sdaps_twoside_bool \c_false_bool
1210 \bool_set:Nn \g_sdaps_print_questionnaire_id_bool \c_false_bool
1211
1212
1213
1214 \tl_new:N \g_sdaps_style_tl
1215 \tl_new:N \g_sdaps_checkmode_tl
1216 \dim_new:N \g_sdaps_edge_left_margin_dim
1217 \dim_new:N \g_sdaps_edge_right_margin_dim
1218 \dim_new:N \g_sdaps_edge_top_margin_dim
1219 \dim_new:N \g_sdaps_edge_bottom_margin_dim
1220 \dim_new:N \g_sdaps_edge_marker_linewidth_dim
1221 \dim_new:N \g_sdaps_edge_marker_length_dim
1222
1223 \dim_new:N \g_sdaps_classic_boxpad_dim
1224 \dim_new:N \g_sdaps_classic_boxsize_dim
1225
1226
1227 \tl_gset:Nn \g_sdaps_style_tl { code128 }
1228 \tl_gset:Nn \g_sdaps_twoside_barcode_tl { both }
1229 \tl_gset:Nn \g_sdaps_checkmode_tl { checkcorrect }
1230 \dim_gset:Nn \g_sdaps_edge_left_margin_dim { 10mm }
1231 \dim_gset:Nn \g_sdaps_edge_right_margin_dim { 10mm }
1232 \dim_gset:Nn \g_sdaps_edge_top_margin_dim { 12mm }
1233 \dim_gset:Nn \g_sdaps_edge_bottom_margin_dim { 12mm }
1234 \dim_gset:Nn \g_sdaps_edge_marker_linewidth_dim { 1bp }
1235 \dim_gset:Nn \g_sdaps_edge_marker_length_dim { 20mm }
1236
1237 \dim_gset:Nn \g_sdaps_classic_boxpad_dim { 3mm }
1238 \dim_gset:Nn \g_sdaps_classic_boxsize_dim { 3.5mm }
1239
1240
1241 \tl_new:N \g__sdaps_questionnaire_id_tl
1242 \tl_gset:Nn \g__sdaps_questionnaire_id_tl { }
```

```

1243
1244 \tl_new:N \g_sdaps_questionnaire_id_label_tl
1245 \tl_gset:Nn \g_sdaps_questionnaire_id_label_tl { }
1246
1247
1248 \tl_new:N \g_sdaps_survey_id_tl
1249 \tl_gset:Nn \g_sdaps_survey_id_tl { 32498923 }
1250
1251 \tl_new:N \g_sdaps_global_id_tl
1252 \tl_gset:Nn \g_sdaps_global_id_tl { }
1253
1254 \tl_new:N \g_sdaps_global_id_label_tl
1255 \tl_gset:Nn \g_sdaps_global_id_label_tl { }
1256
1257
1258
1259
1260 % Settings for code128 barcodes (used in code128 style, who would have thought?)
1261 \dim_new:N \c_sdaps_barcode_height_dim
1262 \dim_gset:Nn \c_sdaps_barcode_height_dim {6.5mm}
1263 % This is the same as the default
1264 \dim_new:N \c_sdaps_barcode_bar_width_dim
1265 \dim_gset:Nn \c_sdaps_barcode_bar_width_dim {0.33mm}
1266 % Same as default. Barwidth is decreased for printing by this value.
1267 \dim_new:N \c_sdaps_barcode_bcorr_dim
1268 \dim_gset:Nn \c_sdaps_barcode_bcorr_dim {0.020mm}
1269
1270 % The padding on the left/right of a barcode. This is the distance that the
1271 % barcodes will be printed from the cornermarks
1272 % Chosen to be the same as the barcode height. Note that the Code-128
1273 % standard requires a quiet zone of max(10*modulesize, ~6.4mm).
1274 \dim_new:N \c_sdaps_barcode_hpad_dim
1275 \dim_gset:Nn \c_sdaps_barcode_hpad_dim {6.5mm}
1276
1277 % This needs to be smaller than 6.5mm because otherwise the content is too close.
1278 % We set it so that it forms a golden ratio  $6.5\text{mm} * (\sqrt{5/4} - 0.5)$ . This means
1279 % we have about 4.5mm padding to the content.
1280 \dim_new:N \c_sdaps_barcode_vpad_dim
1281 \dim_gset:Nn \c_sdaps_barcode_vpad_dim {4.02mm}
1282
1283
1284 \cs_new_protected_nopar:Nn \sdaps_draw_codes:
1285 {
1286   \group_begin:
1287     \begin{scope}[line~width=\g_sdaps_edge_marker_linewidth_dim, shift={(\current~page.south~west)}]
1288       \str_if_eq:VnT \g_sdaps_style_tl { qr } {
1289         \tl_if_empty:VF \g__sdaps_questionnaire_id_tl {
1290           \begin{scope}[shift={(\g_sdaps_edge_left_margin_dim, \g_sdaps_edge_bottom_margin_dim)}]
1291             \node(barcode)[anchor=south~west,outer~sep=0,inner~sep=0]{
1292               \qrcode_render:nV {nolink,version=2,level=H,padding,height=10mm} \g__sdaps_questi

```

```

1293         };
1294     \end{scope}
1295 }
1296
1297 % We unconditionally print this barcode, it is required for the recognition
1298 % process.
1299 \begin{scope}[shift={($(\paperwidth, 0) + (-\g_sdaps_edge_right_margin_dim, \g_sdaps_ed
1300     \pgfmathsetbasenumberlength{4}
1301     \pgfmathdectobase{\paddedpage}{\int_use:N\g_sdaps_page_int}{10}% Yes, padding to 4 ch
1302     \node(barcode)[anchor=south~east,outer~sep=0,inner~sep=0]{
1303         \qrcode_render:nx {nolink,version=2,level=H,padding,height=10mm} {\tl_use:N\g_sdaps
1304     };
1305 \end{scope}
1306
1307 \tl_if_empty:VF \g_sdaps_global_id_tl {
1308     \begin{scope}[shift={($(\paperwidth/2-\g_sdaps_edge_right_margin_dim/2+\g_sdaps_edge_
1309         \node(barcode)[anchor=south,outer~sep=0,inner~sep=0]{
1310             \qrcode_render:nV {nolink,version=2,level=H,padding,height=10mm} \g_sdaps_global_
1311         };
1312     \end{scope}
1313 }
1314 }
1315
1316 \str_if_eq:VnT \g_sdaps_style_tl { code128 } {
1317     \tl_if_empty:VF \g__sdaps_questionnaire_id_tl {
1318         % TODO: do not hardcode the font!
1319         \ttfamily\footnotesize
1320
1321         \begin{scope}[shift={($(\g_sdaps_edge_left_margin_dim, \g_sdaps_edge_bottom_margin_di
1322             \node(barcode)[anchor=south~west,outer~sep=0,inner~sep=0]{
1323                 \X = \c_sdaps_barcode_bar_width_dim
1324                 \bcorr = \c_sdaps_barcode_bcorr_dim
1325                 \barheight = \c_sdaps_barcode_height_dim
1326                 \code_render:V \g__sdaps_questionnaire_id_tl
1327             };
1328             \node[below=1mm~of~barcode,distance=0,anchor=north,outer~sep=0,inner~sep=0]{
1329                 \tl_if_empty:VTF \g_sdaps_questionnaire_id_label_tl {
1330                     \tl_use:N \g__sdaps_questionnaire_id_tl
1331                 } {
1332                     \tl_use:N \g_sdaps_questionnaire_id_label_tl
1333                 }
1334             };
1335         \end{scope}
1336     }
1337
1338 % We unconditionally print this barcode, it is required for the recognition
1339 % process.
1340 \begin{scope}[shift={($(\paperwidth, 0) + (-\g_sdaps_edge_right_margin_dim, \g_sdaps_ed
1341     \pgfmathsetbasenumberlength{4}
1342     \pgfmathdectobase{\paddedpage}{\int_use:N\g_sdaps_page_int}{10}% Yes, padding to 4 ch

```

```

1343 \node(barcode)[anchor=south~east,outer~sep=0,inner~sep=0]{
1344 \X = \c_sdaps_barcode_bar_width_dim
1345 \bcorr = \c_sdaps_barcode_bcorr_dim
1346 \barheight = \c_sdaps_barcode_height_dim
1347 \code_render:x {\tl_use:N\g_sdaps_survey_id_tl \paddedpage}
1348 };
1349 \node[below=1mm~of~barcode,distance=0,anchor=north,outer~sep=0,inner~sep=0]{
1350 % TODO: do not hardcode the font!
1351 \ttfamily\footnotesize
1352
1353 \tl_use:N\g_sdaps_survey_id_tl\,\paddedpage
1354 };
1355 \end{scope}
1356
1357 \tl_if_empty:VF \g_sdaps_global_id_tl {
1358 \begin{scope}[shift={(\paperwidth/2-\g_sdaps_edge_right_margin_dim/2+\g_sdaps_edge_
1359 \node(barcode)[anchor=south,outer~sep=0,inner~sep=0]{
1360 \X = \c_sdaps_barcode_bar_width_dim
1361 \bcorr = \c_sdaps_barcode_bcorr_dim
1362 \barheight = \c_sdaps_barcode_height_dim
1363 \code_render:V \g_sdaps_global_id_tl
1364 };
1365 \node[below=1mm~of~barcode,distance=0,anchor=north,outer~sep=0,inner~sep=0]{
1366 % TODO: do not hardcode the font!
1367 \ttfamily\footnotesize
1368
1369 \tl_if_empty:VTF \g_sdaps_global_id_label_tl {
1370 \tl_use:N \g_sdaps_global_id_tl
1371 } {
1372 \tl_use:N \g_sdaps_global_id_label_tl
1373 }
1374 };
1375 \end{scope}
1376 }
1377 }
1378 \end{scope}%
1379 \group_end:
1380 }
1381
1382 \msg_new:nnn { sdapsbase } { classic_too_many_pages } { You~cannot~have~more~than~six~pages~wit
1383 \msg_new:nnn { sdapsbase } { odd_page_count } { You~have~an~odd~number~of~pages,~this~does~not~
1384
1385 % This needs to be called once for each page!
1386 \cs_new_protected_nopar:Nn \sdaps_page_end: {
1387 \bool_if:NTF \g_sdaps_recognition_bool {
1388 \group_begin:
1389
1390 \int_gincr:N\g_sdaps_page_int%
1391 \sdaps_info_write:x{Pages=\int_use:N\g_sdaps_page_int}
1392

```

```

1393 \normalfont%
1394
1395 \begin{tikzpicture}[remember~picture,overlay]%
1396
1397 %-----
1398 % corner marks and corner boxes in classic mode
1399 %-----
1400 \begin{scope}[line~width=\g_sdaps_edge_marker_linewidth_dim, line~join=miter]
1401   \begin{scope}[shift={{$(current~page.north~west) + (\g_sdaps_edge_left_margin_dim, -\g_
1402     \draw (0,-\g_sdaps_edge_marker_length_dim) -- (0, 0) -- (\g_sdaps_edge_marker_length_
1403
1404     \str_if_eq:VnT \g_sdaps_style_tl { classic } {
1405       % TODO: Is the filled/non-filled mapping still correct?
1406       \int_case:nnTF \g_sdaps_page_int {
1407         { 1 } { \draw }
1408         { 2 } { \fill }
1409         { 3 } { \fill }
1410         { 4 } { \fill }
1411         { 5 } { \fill }
1412         { 6 } { \draw }
1413       } {
1414         (\g_sdaps_classic_boxpad_dim, -\g_sdaps_classic_boxpad_dim) rectangle +(\g_sdaps_
1415       } {
1416         % Error out (first time)
1417         \msg_error:nn { sdapsbase } { classic_too_many_pages }
1418       }
1419     }
1420   \end{scope}
1421   \begin{scope}[shift={{$(current~page.north~east) + (-\g_sdaps_edge_right_margin_dim, -\
1422     \draw (0,-\g_sdaps_edge_marker_length_dim) -- (0,0) -- (-\g_sdaps_edge_marker_length_
1423
1424     \str_if_eq:VnT \g_sdaps_style_tl { classic } {
1425       % TODO: Is the filled/non-filled mapping still correct?
1426       \int_case:nnT \g_sdaps_page_int {
1427         { 1 } { \fill }
1428         { 2 } { \fill }
1429         { 3 } { \draw }
1430         { 4 } { \draw }
1431         { 5 } { \draw }
1432         { 6 } { \draw }
1433       } {
1434         (-\g_sdaps_classic_boxpad_dim, -\g_sdaps_classic_boxpad_dim) rectangle +(-\g_sdap
1435       }
1436     }
1437   \end{scope}
1438   \begin{scope}[shift={{$(current~page.south~west) + (\g_sdaps_edge_left_margin_dim, \g_s
1439     \draw (0,\g_sdaps_edge_marker_length_dim) -- (0, 0) -- (\g_sdaps_edge_marker_length_d
1440
1441     \str_if_eq:VnT \g_sdaps_style_tl { classic } {
1442       % TODO: Is the filled/non-filled mapping still correct?

```

```

1443         \int_case:nnT \g_sdaps_page_int {
1444             { 1 } { \fill }
1445             { 2 } { \draw }
1446             { 3 } { \fill }
1447             { 4 } { \fill }
1448             { 5 } { \draw }
1449             { 6 } { \draw }
1450         } {
1451             (\g_sdaps_classic_boxpad_dim, \g_sdaps_classic_boxpad_dim) rectangle +(\g_sdaps_c
1452         }
1453     }
1454 \end{scope}
1455 \begin{scope}[shift={($(\current~page.south~east) + (-\g_sdaps_edge_right_margin_dim, \g
1456     \draw (0,\g_sdaps_edge_marker_length_dim) -- (0mm, 0mm) -- (-\g_sdaps_edge_marker_len
1457
1458     \str_if_eq:VnT \g_sdaps_style_tl { classic } {
1459         % TODO: Is the filled/non-filled mapping still correct?
1460         \int_case:nnT \g_sdaps_page_int {
1461             { 1 } { \fill }
1462             { 2 } { \draw }
1463             { 3 } { \fill }
1464             { 4 } { \draw }
1465             { 5 } { \draw }
1466             { 6 } { \fill }
1467         } {
1468             (-\g_sdaps_classic_boxpad_dim, \g_sdaps_classic_boxpad_dim) rectangle +(-\g_sdaps
1469         }
1470     }
1471 \end{scope}
1472 \end{scope}
1473
1474 %-----
1475 % barcodes/qr codes
1476 %-----
1477 \bool_if:NTF \g__sdaps_last_page_bool {
1478
1479     \int_compare:nNnTF { \g_sdaps_page_int } = { 1 } {
1480         % if we have a one page document, just always stamp the last
1481         % page. This means that SDAPS can fall back to simplex mode
1482         % automatically.
1483         \sdaps_draw_codes:
1484     } {
1485
1486         \bool_if:NTF \g_sdaps_twoside_bool {
1487             \int_if_odd:VT \g_sdaps_page_int {
1488                 % This should no happen; draw a barcode though to keep things
1489                 % somewhat sane.
1490                 \msg_warning:nn { sdapsbase } { odd_page_count }
1491                 \sdaps_draw_codes:
1492             } {

```



```

1493         % Even page (i.e. back) has a barcode except in "front" mode
1494         \str_if_eq:VnF \g_sdaps_twoside_barcode_tl { front } {
1495             \sdaps_draw_codes:
1496         }
1497     }
1498 } {
1499     \sdaps_draw_codes:
1500 }
1501 }
1502
1503 % TODO: Check whether the page count is as expected?
1504 } {
1505     \bool_if:NTF \g_sdaps_twoside_bool {
1506         \str_if_eq:VnTF \g_sdaps_twoside_barcode_tl { both } {
1507             \sdaps_draw_codes:
1508         } {
1509             \str_if_eq:VnTF \g_sdaps_twoside_barcode_tl { front } {
1510                 \int_if_odd:VT \g_sdaps_page_int {
1511                     \sdaps_draw_codes:
1512                 }
1513             } { % And back is the only thing left
1514                 \int_if_odd:VF \g_sdaps_page_int {
1515                     \sdaps_draw_codes:
1516                 }
1517             }
1518         }
1519     } {
1520         \sdaps_draw_codes:
1521     }
1522 }
1523
1524 %-----
1525 % watermark for non final mode
1526 %-----
1527 \bool_if:NT \g_sdaps_draft_bool {
1528     \node [rotate=60,scale=10,text~opacity=0.2,color=red]
1529         at (current~page.center) {\textsc{draft}};
1530 }
1531 \end{tikzpicture}
1532
1533 \group_end:
1534 } {
1535     \int_gincr:N\g_sdaps_page_int
1536 }
1537 }
1538
1539

```

2.8 Starting/Ending an SDAPS context

These need to be used to begin rendering into an SDAPS context and finishing everything off.

Note that this base package does not automatically call `\sdaps_page_end`: for you, so if you are using this class directly you will need to make sure that this handler is called after all form elements on a page. An easy of doing this is to call it from inside the page footer.

```
1540
1541 \bool_new:N \g__sdaps_last_page_bool
1542
1543 \cs_new_protected_nopar:Nn \sdaps_begin: {
1544   \bool_gset:Nn \g__sdaps_last_page_bool \c_false_bool
1545
1546   % TODO: We really want to make sure nobody modifies the values after \sdaps_begin:
1547   \sdaps_info_write:x{Duplex=\bool_if:NTF \g_sdaps_twoside_bool {true} {false}}
1548   \sdaps_info_write:x{PrintQuestionnaireId=\bool_if:NTF \g_sdaps_print_questionnaire_id_bool {1
1549   \sdaps_info_write:x{
1550     PageSize=\the\paperwidth, \the\paperheight
1551   }
1552   \sdaps_info_write:x{Style=\g_sdaps_style_tl}
1553   \sdaps_info_write:x{CheckMode=\g_sdaps_checkmode_tl}
1554   \sdaps_info_write:x{GlobalID=\g_sdaps_global_id_tl}
1555   \sdaps_info_write:x{GlobalIDLabel=\g_sdaps_global_id_label_tl}
1556
1557   \int_gset:Nn \g_sdaps_page_int { 0 }
1558 }
1559
1560 \cs_new_protected_nopar:Nn \sdaps_end: {
1561   % Note that using \sdaps_info_write_x:n may not work in some cases.
1562   % For this reason we write the out the Pages counter after each page,
1563   % which is fine to do.
1564   % Note that this means that the below "hack" to make onesided documents
1565   % work even in twoside (duplex) mode may not always work either.
1566
1567   \bool_gset:Nn \g__sdaps_last_page_bool \c_true_bool
1568 }
1569
1570
1571 %
```

Change History

v0.1
General: Initial version 1