

The **sdapsbase** 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:nN
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:n { 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_bool}
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 ~ l
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_tmbp_tl { \seq_use:Nn \g_sdaps_object_id_seq {.} }
373
374     \sdaps_context_begin:n {#1}
375         \sdaps_context_put:nV {id} \l_sdaps_tmbp_tl
376         \sdaps_info_write:x {QObject-#2=\tl_use:N\l_sdaps_tmbp_tl. ~ \exp_not:n{#3}}
377         \sdaps_context_append:nVn { _implicit_var } \l_sdaps_tmbp_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, nVn, 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_eqCharCode: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_eqCharCode: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_eqCharCode:VNT \l_sdaps_tmpa_tl _ {
474     \sdaps_context_get:nN { _implicit_var } \l_sdaps_tmpb_tl
475     \tl_if_eq:VnF \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:nTF { #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_tma_tl {
587     \tl_put_right:Nn #1 {}
588     \tl_put_right:NV #1 \l_sdaps_tma_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_tma_tl {
593       \tl_put_right:Nn #1 {}
594       \tl_put_right:NV #1 \l_sdaps_tma_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_draw_tl
649 \tl_new:N \l_sdaps_checkbox_var_tl
650 \tl_new:N \l_sdaps_checkbox_value_tl
651 \bool_new:N \l_sdaps_checkbox_draw_check_bool
652
653 \tl_set:Nn \l_sdaps_checkbox_form_tl { box }
654
655 \tl_new:N \l_sdaps_parse_unknown_tl
656
657
658 % Internal overlays
659 \tl_new:N \l_sdaps_overlay_centered_text_tl
660 \tl_new:N \l_sdaps_overlay_minipage_text_tl
661 \tl_new:N \l_sdaps_overlay_minipage_pos_tl
662 \dim_new:N \l_sdaps_overlay_minipage_pad_dim
663
664 % Note that width/height is the *outside* width/height
665 \keys_define:nn { sdaps / checkbox }
666 {
667   linewidth      .dim_set:N  = \l_sdaps_checkbox_linewidth_dim,
668   linewidth      .initial:n = 1bp,
669   width         .dim_set:N  = \l_sdaps_checkbox_width_dim,
670   width         .initial:n = 3.5mm,
671   height        .dim_set:N  = \l_sdaps_checkbox_height_dim,
672   height        .initial:n = 3.5mm,
673   form          .choices:nn = { box, ellipse } { \tl_set:Nx \l_sdaps_checkbox_form_tl { \l_keys
674   value         .tl_set:N  = \l_sdaps_checkbox_value_tl,

```

```

675
676   fill      .tl_set:N    = \l_sdaps_checkbox_fill_tl,
677   fill      .initial:n  = { white },
678
679   draw      .tl_set:N    = \l_sdaps_checkbox_draw_tl,
680   draw      .initial:n  = { . },
681
682   draw_check .bool_set:N = \l_sdaps_checkbox_draw_check_bool,
683   draw_check .default:n = true,
684   draw_check .initial:n = false,
685
686   % Simple node overlay
687   centered_text .tl_set:N    = \l_sdaps_overlay_centered_text_tl,
688   centered_text .initial:n  = {},
689
690   % minipage overlay
691   text      .tl_set:N    = \l_sdaps_overlay_minipage_text_tl,
692   text      .initial:n  = {},
693   text_align .tl_set:N    = \l_sdaps_overlay_minipage_pos_tl,
694   text_align .initial:n  = {c},
695   text_padding .dim_set:N = \l_sdaps_overlay_minipage_pad_dim,
696   text_padding .initial:n = {2bp},
697
698   ellipse   .meta:n   = { form=ellipse },
699   box       .meta:n   = { form=box },
700 }

```

2.5 Checkboxes

```

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

```

```

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

```

```

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

```

```

822 \tl_if_empty:NF \l_sdaps_overlay_minipage_text_tl {
823   \node[anchor=center,inner sep=0pt,outer sep=0pt] at ($(\l_sdaps_x_dim, \l_sdaps_y_dim) + 0.
824     \dim_set:Nn \l_sdaps_width_dim { \l_sdaps_width_dim - 2\l_sdaps_overlay_minipage_pad_dim
825     \dim_set:Nn \l_sdaps_height_dim { \l_sdaps_height_dim - 2\l_sdaps_overlay_minipage_pad_dim
826
827   \begin{minipage}[t][\l_sdaps_height_dim][\l_sdaps_overlay_minipage_pos_tl]{\l_sdaps_width_dim}
828     % Hm, is this sane?
829     \tex_let:D \textheight{\l_sdaps_height_dim}
830     \l_sdaps_overlay_minipage_text_tl
831   \end{minipage}
832 };
833 }
834 }

835 \seq_put_left:Nn \g__sdaps_checkbox_overlays_seq \sdaps_overlay_minipage:
836 \seq_put_left:Nn \g__sdaps_textbox_overlays_seq \sdaps_overlay_minipage:
837
838

```

2.6 Textboxes

```

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

```

```

869 % minipage overlay
870 text .tl_set:N = \l_sdaps_overlay_minipage_text_tl,
871 text .initial:n = {},
872 text_align .tl_set:N = \l_sdaps_overlay_minipage_pos_tl,
873 text_align .initial:n = {c},
874 text_padding .dim_set:N = \l_sdaps_overlay_minipage_pad_dim,
875 text_padding .initial:n = {2bp},
876 }
877
878
879
880
881 \dim_new:N \l_sdaps_textbox_dp_dim
882 \dim_new:N \l_sdaps_textbox_ht_dim
883 \dim_new:N \l_sdaps_textbox_wd_dim
884 \dim_new:N \l_sdaps_textbox_pad_dim
885
886
887 \msg_new:nnn { sdapsbase } { textbox_wrong_mode } { Impossible~to~layout~a~#1~textbox~in~#2~mod
888
889 \cs_new_protected_nopar:Nn \__sdaps_textbox_prepare:n
890 {
891   \tl_set:Nn \l_sdaps_parse_unknown_tl {}
892
893   \__sdaps_generate_var:nN { #1 } \l_sdaps_var_tl
894
895   \sdaps_context_get:nN { textboxtype } \l_sdaps_tmpa_tl
896   \tl_if_eq:VnTF \l_sdaps_tmpa_tl { codebox } {
897     \tl_set:Nn \l_sdaps_textbox_boxtype_tl { Codebox }
898   }
899   \tl_set:Nn \l_sdaps_textbox_boxtype_tl { Textbox }
900 }
901
902 \__sdaps_append_from_context:nN { * } \l_sdaps_parse_unknown_tl
903 \__sdaps_append_from_context:VN \l_sdaps_tmpa_tl \l_sdaps_parse_unknown_tl
904 \__sdaps_append_override_options:NVn \l_sdaps_parse_unknown_tl \l_sdaps_var_tl { }
905
906 \keys_set_known:nVN { sdaps / textbox } \l_sdaps_parse_unknown_tl \l_sdaps_parse_unknown_tl
907
908 \__sdaps_box_inc_object_id:
909 }
910
911 \cs_new_protected_nopar:Nn \sdaps_textbox_vhstretch:nnn
912 {

```

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.

```
913 \if_mode_math:
```

```

914     \msg_error:nnnn { sdapsbase } { textbox_wrong_mode } { vhstretch } { math }
915     \else:
916     \tex_par:D
917     \if_mode_horizontal:
918         \msg_error:nnnn { sdapsbase } { textbox_wrong_mode } { vhstretch } { horizontal }
919     \else:
920         % Go into vertical mode by ending the current paragraph and insert
921         % \cs{widowpenalty} so that we don't get an unwelcome page break. It is
922         % assumed that any explanation for the textbox will be on top.
923         \tex_penalty:D \tex_widowpenalty:D
924
925     \group_begin:%
926
927     \__sdaps_textbox_prepare:n { #1 }
928

```

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.

```

929     % TODO: Make this configurable
930     \dim_set:Nn \l_sdaps_textbox_dp_dim { 0.5ex }
931     \dim_set:Nn \l_sdaps_textbox_ht_dim { 1.7ex }
932
933     \vbox:n {
934         \sdaps_if_rtl:T {\beginL}
935         \leftskip=0pt
936         \rightskip=0pt plus 1fill
937         \noindent
938         \tex_vrule:D height \l_sdaps_textbox_ht_dim depth 0pt width 0pt
939         \pgf@sys@markposition{textboxtop}\int_use:N\g_sdaps_textbox_num_int}
940         \sdaps_if_rtl:T {\endL}
941     }
942
943     \tex_penalty:D 10000
944     \nointerlineskip
945
946     \tex_cleaders:D\tex_hbox:D{} \skip_vertical:n{#2 - \l_sdaps_textbox_ht_dim}
947
948     \tex_penalty:D 10000
949     \nointerlineskip

```

```

950
951 \tex_cleaders:D\tex_hbox:D{}\skip_vertical:n{\stretch{#3}}
952
953 \tex_penalty:D 10000
954 \nointerlineskip
955
956 \vbox:n {
957   \sdaps_if_rtl:T {\beginL}
958   \noindent
959   \leftskip=0pt plus 1fill
960   \rightskip=0pt
961   \begin{tikzpicture}[remember picture, overlay, shift=(current page.south west)]
962     \pgfsys@getposition{\pgfpictureorigin}{\@sdaps@pageorigin}
963     \pgfsys@getposition{textboxtop}\int_use:N\g_sdaps_textbox_num_int{\@sdaps@textboxtoppo
964     \pgfpointadd{\@sdaps@textboxtoppos}%
965       \pgfpointadd{\@sdaps@pageorigin}{\pgfpoint{0}{\l_sdaps_textbox_ht_dim}}
966   }
967   \pgfgetlastxy{\l_sdaps_x}{\l_sdaps_y}
968   \dim_set:Nn \l_sdaps_x_dim {\l_sdaps_x}
969   \dim_set:Nn \l_sdaps_y_dim {\l_sdaps_y}
970
971   \pgfsys@getposition{\pgfpictureid}{\@sdaps@textboxbottompos}
972   \pgfpointdiff{\@sdaps@textboxtoppos}{\@sdaps@textboxbottompos}
973
974   % Is there a more elegant way to multiply the height with -1?
975   \pgfgetlastxy{\l_sdaps_width}{\l_sdaps_height}
976   % Add the minimum height (i.e. depth below the last baseline) to the
977   % overall height.
978   \pgfpoint{\l_sdaps_width}{-\l_sdaps_height + \l_sdaps_textbox_ht_dim}
979   \pgfgetlastxy{\l_sdaps_width}{\l_sdaps_height}
980   \dim_set:Nn \l_sdaps_width_dim {\l_sdaps_width}
981   \dim_set:Nn \l_sdaps_height_dim {\l_sdaps_height}
982
983   % Draw the rectangle
984   \draw[line width=\l_sdaps_textbox_linewidth_dim, fill=\l_sdaps_textbox_fill_tl, draw=\l_s
985
986   \begin{scope}
987     \seq_map_inline:Nn \g_sdaps_textbox_overlays_seq {##1}
988   \end{scope}
989
990   \bool_if:NT \g_sdaps_write_enable_bool {
991     \sdaps_context_get:nN {id} \l_sdaps_tmpa_tl
992     \tl_if_eq:VnTF \l_sdaps_tmpa_tl { \q_no_value } {
993       \msg_warning:nn { sdapsbase } { no_qid }
994     } {
995       \sdaps_info_write_x:x {
996         Box[\l_sdaps_tmpa_tl]=\l_sdaps_textbox_boxtype_tl,
997         \exp_not:n{\int_use:N\g_sdaps_page_int},
998         \dim_use:N \l_sdaps_x_dim,
999         \dim_use:N \l_sdaps_y_dim,

```

```

1000          \dim_use:N \l_sdaps_width_dim,
1001          \dim_use:N \l_sdaps_height_dim,
1002          \dim_use:N \l_sdaps_textbox_linewidth_dim,
1003          \tl_use:N \l_sdaps_var_tl,
1004      }
1005  }
1006 }
1007 \end{tikzpicture}
1008 \sdaps_if_rtl:T {\endL}
1009 }
1010
1011 % We are done here, but we need the correct depth, so skip around a bit
1012 \tex_penalty:D 10000
1013 \nointerlineskip
1014
1015 \skip_vertical:n{ - \l__sdaps_textbox_dp_dim}
1016
1017 \tex_penalty:D 10000
1018 \nointerlineskip
1019
1020 \hbox:n { \vrule depth \l__sdaps_textbox_dp_dim height Opt width Opt }
1021
1022 \int_gincr:N\g__sdaps_textbox_num_int
1023
1024 \group_end:
1025 \fi:
1026 \fi:
1027 }
1028 \cs_generate_variant:Nn \sdaps_textbox_vhstretch:nnn { Vnn }
1029
1030 \cs_new_protected_nopar:Nn \sdaps_textbox_vhstretch:nn
1031 {
1032   \sdaps_textbox_vhstretch:nnn { #1 } { #2 } { 1 }
1033 }
1034
1035
1036 \cs_new_protected_nopar:Nn \sdaps_textbox_hstretch:nnnn
1037 {
1038   \group_begin:
1039     \sdaps_if_rtl:T {\beginL}
1040
1041   \__sdaps_textbox_prepare:n { #1 }
1042
1043   \dim_set:Nn \l_tmpa_dim { #2 }
1044   \dim_set:Nn \l_tmpb_dim { #3 }
1045
1046   % Place a vrule to make space for the top/bottom padding
1047   \tex_vrule:D depth \dim_use:N \l_tmpa_dim height \dim_use:N \l_tmpb_dim width Opt \nobreak
1048   \pgf@markposition{textboxstart}\int_use:N\g__sdaps_textbox_num_int} \nobreak
1049

```

```

1050 \skip_horizontal:n { #4 + \stretch{#5} } \nobreak
1051
1052 % The textbox (rendered on the background)
1053 \begin{tikzpicture}[remember picture, overlay, shift=(current page.south west)]
1054   \pgf@sys@getposition{\pgfpageorigin}{\@sdaps@pageorigin}
1055   \pgf@sys@getposition{\textboxstart\int_use:N\g_sdaps_textbox_num_int}{\@sdaps@textboxpos}
1056   % The position here is the position of the baseline.
1057   % So move up by height (param 2) to get the correct vertical position.
1058   \pgfpointadd{\@sdaps@textboxpos}{%
1059     \pgfpointadd{\@sdaps@pageorigin}{\pgfpoint{0}{\dim_use:N \l_tmpb_dim}}
1060   }
1061   \pgfgetlastxy{\l_sdaps_x}{\l_sdaps_y}
1062   \dim_set:Nn \l_sdaps_x_dim {\l_sdaps_x}
1063   \dim_set:Nn \l_sdaps_y_dim {\l_sdaps_y}
1064
1065   \pgf@sys@getposition{\pgfpictureid}{\@sdaps@textboxendpos}
1066   % Calculate width and add the height to it
1067   \pgfpointadd{%
1068     \pgfpointdiff{\@sdaps@textboxpos}{\@sdaps@textboxendpos}
1069   }{\pgfpoint{0pt}{\dim_use:N \l_tmpa_dim + \dim_use:N \l_tmpb_dim}}
1070   \pgfgetlastxy{\l_sdaps_width}{\l_sdaps_height}
1071   \dim_set:Nn \l_sdaps_width_dim {\l_sdaps_width}
1072   \dim_set:Nn \l_sdaps_height_dim {\l_sdaps_height}
1073
1074   % Draw the rectangle
1075   \draw[line width=\l_sdaps_textbox_linewidth_dim, fill=\l_sdaps_textbox_fill_tl, draw=\l_sdaps_textbox_stroke_color]
1076
1077   \begin{scope}
1078     \seq_map_inline:Nn \g_sdaps_textbox_overlays_seq {##1}
1079   \end{scope}
1080
1081   \bool_if:NT \g_sdaps_write_enable_bool {
1082     \sdaps_context_get:nN {id} \l_sdaps_tma_t1
1083     \tl_if_eq:VnTF \l_sdaps_tma_t1 { \q_no_value } {
1084       \msg_warning:nn { sdapsbase } { no_qid }
1085     } {
1086       \sdaps_info_write_x:x {
1087         Box[\l_sdaps_tma_t1]=\l_sdaps_textbox_boxtype_t1,
1088         \exp_not:n{\int_use:N\g_sdaps_page_int},
1089         \dim_use:N \l_sdaps_x_dim,
1090         \dim_use:N \l_sdaps_y_dim,
1091         \dim_use:N \l_sdaps_width_dim,
1092         \dim_use:N \l_sdaps_height_dim,
1093         \dim_use:N \l_sdaps_textbox_linewidth_dim,
1094         \tl_use:N \l_sdaps_var_t1,
1095       }
1096     }
1097   }
1098 \end{tikzpicture}
1099

```

```

1100   \int_gincr:N\g_sdaps_textbox_num_int
1101
1102   \sdaps_if_rtl:T {\endL}
1103   \group_end:
1104 }
1105
1106 \cs_new_protected_nopar:Nn \__sdaps_textbox_prepare_coffin:
1107 {
1108   \dim_set:Nn \l__sdaps_textbox_ht_dim { \coffin_ht:N \l__sdaps_textbox_coffin }
1109   \dim_set:Nn \l__sdaps_textbox_dp_dim { \coffin_dp:N \l__sdaps_textbox_coffin }
1110   \dim_set:Nn \l__sdaps_textbox_wd_dim { \coffin_wd:N \l__sdaps_textbox_coffin }
1111
1112   \hcoffin_set:Nn \l_tmpa_coffin {
1113     \tex_vrule:D depth Opt height \dim_eval:n { \l__sdaps_textbox_ht_dim + \l__sdaps_textbox_pa
1114
1115   \pdfsavepos
1116
1117   \dim_set:Nn \l_sdaps_width_dim { \l__sdaps_textbox_wd_dim + 2\l__sdaps_textbox_pad_dim }
1118   \dim_set:Nn \l_sdaps_height_dim { \l__sdaps_textbox_ht_dim + \l__sdaps_textbox_dp_dim + 2\l_
1119
1120   % pdflast[xy]pos is the PDF position of the top left corner excluding the
1121   % origin
1122   \bool_if:NT \g_sdaps_write_enable_bool {
1123     \sdaps_context_get:nN {id} \l_sdaps_tma_t1
1124     \tl_if_eq:VnTF \l_sdaps_tma_t1 { \q_no_value } {
1125       \msg_warning:nn { sdapsbase } { no_qid }
1126     }
1127     \sdaps_info_write_x:x {
1128       Box[\l_sdaps_tma_t1]=\l_sdaps_textbox_boxtype_t1,
1129       \exp_not:n{\int_use:N\g_sdaps_page_int},
1130       \exp_not:n{\dim_eval:n { \hoffset + \the\pdflastxpos sp}},
1131       \exp_not:n{\dim_eval:n} { \exp_not:n { \voffset + \the\pdflastypos sp} + \dim_use:N \l_
1132       \dim_use:N \l_sdaps_width_dim,
1133       \dim_use:N \l_sdaps_height_dim,
1134       \dim_use:N \l_sdaps_textbox_lineWidth_dim,
1135       \tl_use:N \l_sdaps_var_t1,
1136     }
1137   }
1138 }
1139
1140 \dim_set:Nn \l_sdaps_x_dim { Opt }
1141 \dim_set:Nn \l_sdaps_y_dim { \l_sdaps_textbox_ht_dim + \l_sdaps_textbox_pad_dim }
1142
1143 % The textbox (rendered on the background)
1144 \begin{tikzpicture}[overlay]
1145   % Draw the rectangle
1146   \draw[line width=\l_sdaps_textbox_lineWidth_dim, fill=\l_sdaps_textbox_fill_t1, draw=\l_sdap
1147
1148 \begin{scope}
1149   \seq_map_inline:Nn \g_sdaps_textbox_overlays_seq {##1}%

```

```

1150      \end{scope}
1151  \end{tikzpicture}
1152  \skip_horizontal:n { \l_sdaps_textbox_pad_dim }
1153 }
1154
1155 \hcoffin_set:Nn \l_tmpb_coffin {
1156   \skip_horizontal:n { \l_sdaps_textbox_pad_dim }
1157   \tex_vrule:D depth \l_sdaps_textbox_pad_dim height Opt width Opt
1158 }
1159
1160 \coffin_join:NnnNnnnn \l_tmpa_coffin { r } { H } \l_sdaps_textbox_coffin { l } { H } { Opt }
1161 \coffin_join:NnnNnnnn \l_tmpa_coffin { r } { b } \l_tmpb_coffin { l } { t } { Opt } { Opt }
1162
1163 \coffin_set_eq:NN \l_sdaps_textbox_coffin \l_tmpa_coffin
1164
1165
1166 \int_gincr:N\g_sdaps_textbox_num_int
1167 }
1168
1169 \coffin_new:N \l_sdaps_textbox_coffin
1170 \cs_new_protected_nopar:Nn \sdaps_textbox_hbox:nnn
1171 {
1172   \group_begin:
1173
1174   \__sdaps_textbox_prepare:n { #1 }
1175
1176   \hcoffin_set:Nn \l_sdaps_textbox_coffin { \tl_trim_spaces:n { #3 } }
1177
1178   \dim_set:Nn \l_sdaps_textbox_pad_dim { #2 }
1179
1180   \__sdaps_textbox_prepare_coffin:
1181
1182   \coffin_typeset:Nnnnn \l_sdaps_textbox_coffin { l } { H } { Opt } { Opt }
1183
1184   \group_end:
1185 }
1186
1187 \cs_new_protected:Nn \sdaps_textbox_vbox:nnnn
1188 {
1189   \group_begin:
1190
1191   \__sdaps_textbox_prepare:n { #1 }
1192
1193   \dim_set:Nn \l_sdaps_textbox_pad_dim { #3 }
1194   \dim_set:Nn \l_sdaps_textbox_wd_dim { #2 - 2\l_sdaps_textbox_pad_dim }
1195
1196   \vcoffin_set:Nnn \l_sdaps_textbox_coffin { \l_sdaps_textbox_wd_dim } { \tl_trim_spaces:n
1197
1198   \__sdaps_textbox_prepare_coffin:
1199

```

```

1200      \coffin_typeset:Nnnnn \l_sdaps_textbox_coffin { 1 } { H } { Opt } { Opt }
1201
1202  \group_end:
1203 }
1204

```

2.7 Page Marking

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

```

1205
1206 \int_new:N \g_sdaps_page_int
1207 \int_set:Nn \g_sdaps_page_int { 0 }
1208
1209 % Disabling recognition disables all drawings related to automatic recognition
1210 \bool_new:N \g_sdaps_recognition_bool
1211 \bool_new:N \g_sdaps_draft_bool
1212 \bool_new:N \g_sdaps_twoside_bool
1213 \bool_new:N \g_sdaps_print_questionnaire_id_bool
1214
1215 \bool_set:Nn \g_sdaps_recognition_bool \c_true_bool
1216 \bool_set:Nn \g_sdaps_draft_bool \c_true_bool
1217 \bool_set:Nn \g_sdaps_twoside_bool \c_false_bool
1218 \bool_set:Nn \g_sdaps_print_questionnaire_id_bool \c_false_bool
1219
1220
1221
1222 \tl_new:N \g_sdaps_style_tl
1223 \tl_new:N \g_sdaps_checkmode_tl
1224 \dim_new:N \g_sdaps_edge_left_margin_dim
1225 \dim_new:N \g_sdaps_edge_right_margin_dim
1226 \dim_new:N \g_sdaps_edge_top_margin_dim
1227 \dim_new:N \g_sdaps_edge_bottom_margin_dim
1228 \dim_new:N \g_sdaps_edge_marker_linewidth_dim
1229 \dim_new:N \g_sdaps_edge_marker_length_dim
1230
1231 \dim_new:N \g_sdaps_classic_boxpad_dim
1232 \dim_new:N \g_sdaps_classic_boxsize_dim
1233
1234
1235 \tl_gset:Nn \g_sdaps_style_tl { code128 }
1236 \tl_gset:Nn \g_sdaps_twoside_barcode_tl { both }
1237 \tl_gset:Nn \g_sdaps_checkmode_tl { checkcorrect }
1238 \dim_gset:Nn \g_sdaps_edge_left_margin_dim { 10mm }
1239 \dim_gset:Nn \g_sdaps_edge_right_margin_dim { 10mm }
1240 \dim_gset:Nn \g_sdaps_edge_top_margin_dim { 12mm }
1241 \dim_gset:Nn \g_sdaps_edge_bottom_margin_dim { 12mm }
1242 \dim_gset:Nn \g_sdaps_edge_marker_linewidth_dim { 1bp }
1243 \dim_gset:Nn \g_sdaps_edge_marker_length_dim { 20mm }
1244

```

```

1245 \dim_gset:Nn \g_sdaps_classic_boxpad_dim { 3mm }
1246 \dim_gset:Nn \g_sdaps_classic_boxsize_dim { 3.5mm }
1247
1248
1249 \tl_new:N \g_sdaps_questionnaire_id_tl
1250 \tl_gset:Nn \g_sdaps_questionnaire_id_tl { }
1251
1252 \tl_new:N \g_sdaps_questionnaire_id_label_tl
1253 \tl_gset:Nn \g_sdaps_questionnaire_id_label_tl { }
1254
1255
1256 \tl_new:N \g_sdaps_survey_id_tl
1257 \tl_gset:Nn \g_sdaps_survey_id_tl { 32498923 }
1258
1259 \tl_new:N \g_sdaps_global_id_tl
1260 \tl_gset:Nn \g_sdaps_global_id_tl { }
1261
1262 \tl_new:N \g_sdaps_global_id_label_tl
1263 \tl_gset:Nn \g_sdaps_global_id_label_tl { }
1264
1265
1266
1267
1268 % Settings for code128 barcodes (used in code128 style, who would have thought?)
1269 \dim_new:N \c_sdaps_barcode_height_dim
1270 \dim_gset:Nn \c_sdaps_barcode_height_dim {6.5mm}
1271 % This is the same as the default
1272 \dim_new:N \c_sdaps_barcode_bar_width_dim
1273 \dim_gset:Nn \c_sdaps_barcode_bar_width_dim {0.33mm}
1274 % Same as default. Barwidth is decreased for printing by this value.
1275 \dim_new:N \c_sdaps_barcode_bcorr_dim
1276 \dim_gset:Nn \c_sdaps_barcode_bcorr_dim {0.020mm}
1277
1278 % The padding on the left/right of a barcode. This is the distance that the
1279 % barcodes will be printed from the cornermarks
1280 % Choosen to be the same as the barcode height. Note that the Code-128
1281 % standard requires a quiet zone of max(10*modulesize, ~6.4mm).
1282 \dim_new:N \c_sdaps_barcode_hpad_dim
1283 \dim_gset:Nn \c_sdaps_barcode_hpad_dim {6.5mm}
1284
1285 % This needs to be smaller than 6.5mm because otherwise the content is too close.
1286 % We set it so that it forms a golden ratio 6.5mm*(sqrt(5/4)-0.5). This means
1287 % we have about 4.5mm padding to the content.
1288 \dim_new:N \c_sdaps_barcode_vpad_dim
1289 \dim_gset:Nn \c_sdaps_barcode_vpad_dim {4.02mm}
1290
1291
1292 \cs_new_protected_nopar:Nn \sdaps_draw_codes:
1293 {
1294   \group_begin:

```

```

1295 \begin{scope}[line~width=\g_sdaps_edge_marker_linewidth_dim, shift={(current~page.south~west)}]
1296   \str_if_eq:VnT \g_sdaps_style_tl { qr } {
1297     \tl_if_empty:VF \g_sdaps_questionnaire_id_tl {
1298       \begin{scope}[shift={($(\g_sdaps_edge_left_margin_dim, \g_sdaps_edge_bottom_margin_dim)}]
1299         \node[barcode] [anchor=south~west,outer~sep=0,inner~sep=0] {
1300           \qrcode_render:nV {nolink,version=2,level=H,padding,height=10mm} \g_sdaps_questionnaire_id_tl
1301         };
1302       \end{scope}
1303     }
1304   }
1305   % We unconditionally print this barcode, it is required for the recognition
1306   % process.
1307   \begin{scope}[shift={($(\paperwidth, 0) + (-\g_sdaps_edge_right_margin_dim, \g_sdaps_edge_right_margin_dim)}]
1308     \pgfmathsetbasenumberlength{4}
1309     \pgfmathdectobase{\paddedpage}{\int_use:N\g_sdaps_page_int}{10}% Yes, padding to 4 characters
1310     \node[barcode] [anchor=south~east,outer~sep=0,inner~sep=0] {
1311       \qrcode_render:nx {nolink,version=2,level=H,padding,height=10mm} {\tl_use:N\g_sdaps_questionnaire_id_tl}
1312     };
1313   \end{scope}
1314
1315   \tl_if_empty:VF \g_sdaps_global_id_tl {
1316     \begin{scope}[shift={($(\paperwidth/2-\g_sdaps_edge_right_margin_dim/2+\g_sdaps_edge_right_margin_dim, \g_sdaps_edge_right_margin_dim)}]
1317       \node[barcode] [anchor=south,outer~sep=0,inner~sep=0] {
1318         \qrcode_render:nV {nolink,version=2,level=H,padding,height=10mm} \g_sdaps_global_id_tl
1319       };
1320     \end{scope}
1321   }
1322 }
1323
1324 \str_if_eq:VnT \g_sdaps_style_tl { code128 } {
1325   \tl_if_empty:VF \g_sdaps_questionnaire_id_tl {
1326     % TODO: do not hardcode the font!
1327     \ttfamily\footnotesize
1328
1329   \begin{scope}[shift={($(\g_sdaps_edge_left_margin_dim, \g_sdaps_edge_bottom_margin_dim)}]
1330     \node[barcode] [anchor=south~west,outer~sep=0,inner~sep=0] {
1331       \X = \c_sdaps_barcode_bar_width_dim
1332       \bcorr = \c_sdaps_barcode_bcorr_dim
1333       \barheight = \c_sdaps_barcode_height_dim
1334       \code_render:V \g_sdaps_questionnaire_id_tl
1335     };
1336     \node[below=1mm~of~barcode,distance=0,anchor=north,outer~sep=0,inner~sep=0] {
1337       \tl_if_empty:VTF \g_sdaps_questionnaire_id_label_tl {
1338         \tl_use:N \g_sdaps_questionnaire_id_label_tl
1339       } {
1340         \tl_use:N \g_sdaps_questionnaire_id_label_tl
1341       }
1342     };
1343   \end{scope}
1344 }

```

```

1345
1346     % We unconditionally print this barcode, it is required for the recognition
1347     % process.
1348     \begin{scope}[shift={($(\paperwidth, 0) + (-\g_sdaps_edge_right_margin_dim, \g_sdaps_ed-
1349     \pgfmathsetbasenumberlength{4}
1350     \pgfmathdecoctobase{\paddedpage}{\int_use:\N\g_sdaps_page_int}{10}}% Yes, padding to 4 ch
1351     \node[barcode] [anchor=south~east,outer~sep=0,inner~sep=0]{
1352         \X = \c_sdaps_barcode_bar_width_dim
1353         \bcorr = \c_sdaps_barcode_bcorr_dim
1354         \barheight = \c_sdaps_barcode_height_dim
1355         \code_render:x {\tl_use:\N\g_sdaps_survey_id_tl \paddedpage}
1356     };
1357     \node[below=1mm~of~barcode,distance=0,anchor=north,outer~sep=0,inner~sep=0]{
1358         % TODO: do not hardcode the font!
1359         \ttfamily\footnotesize
1360             \tl_use:\N\g_sdaps_survey_id_tl\,,\paddedpage
1361         };
1362     \end{scope}
1363
1364
1365     \tl_if_empty:VF \g_sdaps_global_id_tl {
1366         \begin{scope}[shift={($(\paperwidth/2-\g_sdaps_edge_right_margin_dim/2+\g_sdaps_ed-
1367             \node[barcode] [anchor=south,outer~sep=0,inner~sep=0]{
1368                 \X = \c_sdaps_barcode_bar_width_dim
1369                 \bcorr = \c_sdaps_barcode_bcorr_dim
1370                 \barheight = \c_sdaps_barcode_height_dim
1371                 \code_render:V \g_sdaps_global_id_tl
1372             };
1373             \node[below=1mm~of~barcode,distance=0,anchor=north,outer~sep=0,inner~sep=0]{
1374                 % TODO: do not hardcode the font!
1375                 \ttfamily\footnotesize
1376                     \tl_if_empty:VTF \g_sdaps_global_id_label_tl {
1377                         \tl_use:N \g_sdaps_global_id_tl
1378                     } {
1379                         \tl_use:N \g_sdaps_global_id_label_tl
1380                     }
1381                 };
1382             \end{scope}
1383         };
1384     }
1385
1386     \end{scope}%
1387     \group_end:
1388 }
1389
1390 \msg_new:nnn { sdapsbase } { classic_too_many_pages } { You~cannot~have~more~than~six~pages~wit-
1391 \msg_new:nnn { sdapsbase } { odd_page_count } { You~have~an~odd~number~of~pages,~this~does~not~
1392
1393 % This needs to be called once for each page!
1394 \cs_new_protected_nopar:Nn \sdaps_page_end: {

```

```

1395 \bool_if:NTF \g_sdaps_recognition_bool {
1396   \group_begin:
1397
1398   \int_gincr:N\g_sdaps_page_int%
1399   \sdaps_info_write:x{Pages=\int_use:N\g_sdaps_page_int}
1400
1401   \normalfont%
1402
1403   \begin{tikzpicture}[remember picture,overlay]%
1404
1405   %-----
1406   % corner marks and corner boxes in classic mode
1407   %-----
1408   \begin{scope}[line~width=\g_sdaps_edge_marker_linewidth_dim, line~join=miter]
1409     \begin{scope}[shift={($(current~page.north~west) + (\g_sdaps_edge_left_margin_dim, -\g_sdaps_edge_marker_length_dim) -- (0, 0) -- (\g_sdaps_edge_marker_length_dim, -\g_sdaps_edge_marker_length_dim) rectangle +(0, 0)}]
1410       \draw (0,-\g_sdaps_edge_marker_length_dim) -- (0, 0) -- (\g_sdaps_edge_marker_length_dim, -\g_sdaps_edge_marker_length_dim) rectangle +(0, 0);
1411
1412       \str_if_eq:VnT \g_sdaps_style_tl { classic } {
1413         % TODO: Is the filled/non-filled mapping still correct?
1414         \int_case:nnTF \g_sdaps_page_int {
1415           { 1 } { \draw }
1416           { 2 } { \fill }
1417           { 3 } { \fill }
1418           { 4 } { \fill }
1419           { 5 } { \fill }
1420           { 6 } { \draw }
1421         } {
1422           (\g_sdaps_classic_boxpad_dim, -\g_sdaps_classic_boxpad_dim) rectangle +(\g_sdaps_classic_boxpad_dim, -\g_sdaps_classic_boxpad_dim);
1423         }
1424         % Error out (first time)
1425         \msg_error:nn { sdapsbase } { classic_too_many_pages }
1426       }
1427     }
1428   \end{scope}
1429   \begin{scope}[shift={($(current~page.north~east) + (-\g_sdaps_edge_right_margin_dim, -\g_sdaps_edge_marker_length_dim) -- (0,0) -- (-\g_sdaps_edge_marker_length_dim, -\g_sdaps_edge_marker_length_dim) rectangle +(0, 0)}]
1430     \draw (0,-\g_sdaps_edge_marker_length_dim) -- (0,0) -- (-\g_sdaps_edge_marker_length_dim, -\g_sdaps_edge_marker_length_dim) rectangle +(0, 0);
1431
1432     \str_if_eq:VnT \g_sdaps_style_tl { classic } {
1433       % TODO: Is the filled/non-filled mapping still correct?
1434       \int_case:nnTF \g_sdaps_page_int {
1435         { 1 } { \fill }
1436         { 2 } { \fill }
1437         { 3 } { \draw }
1438         { 4 } { \draw }
1439         { 5 } { \draw }
1440         { 6 } { \draw }
1441       } {
1442         (-\g_sdaps_classic_boxpad_dim, -\g_sdaps_classic_boxpad_dim) rectangle +(-\g_sdaps_classic_boxpad_dim, -\g_sdaps_classic_boxpad_dim);
1443       }
1444     }

```

```

1445 \end{scope}
1446 \begin{scope}[shift={($(current~page.south~west) + (\g_sdaps_edge_left_margin_dim, \g_sdaps_page_int)}]
1447   \draw (0,\g_sdaps_edge_marker_length_dim) -- (0, 0) -- (\g_sdaps_edge_marker_length_dim, \g_sdaps_page_int);
1448 
1449 \str_if_eq:VnT \g_sdaps_style_tl { classic } {
1450   % TODO: Is the filled/non-filled mapping still correct?
1451   \int_case:nnT \g_sdaps_page_int {
1452     { 1 } { \fill }
1453     { 2 } { \draw }
1454     { 3 } { \fill }
1455     { 4 } { \fill }
1456     { 5 } { \draw }
1457     { 6 } { \draw }
1458   } {
1459     (\g_sdaps_classic_boxpad_dim, \g_sdaps_classic_boxpad_dim) rectangle +(\g_sdaps_classic_boxpad_dim, \g_sdaps_classic_boxpad_dim);
1460   }
1461 }
1462 \end{scope}
1463 \begin{scope}[shift={($(current~page.south~east) + (-\g_sdaps_edge_right_margin_dim, \g_sdaps_page_int)}]
1464   \draw (0,\g_sdaps_edge_marker_length_dim) -- (0mm, 0mm) -- (-\g_sdaps_edge_marker_length_dim, \g_sdaps_page_int);
1465 
1466 \str_if_eq:VnT \g_sdaps_style_tl { classic } {
1467   % TODO: Is the filled/non-filled mapping still correct?
1468   \int_case:nnT \g_sdaps_page_int {
1469     { 1 } { \fill }
1470     { 2 } { \draw }
1471     { 3 } { \fill }
1472     { 4 } { \draw }
1473     { 5 } { \draw }
1474     { 6 } { \fill }
1475   } {
1476     (-\g_sdaps_classic_boxpad_dim, \g_sdaps_classic_boxpad_dim) rectangle +(-\g_sdaps_classic_boxpad_dim, \g_sdaps_classic_boxpad_dim);
1477   }
1478 }
1479 \end{scope}
1480 \end{scope}
1481 
1482 %-----
1483 % barcodes/qr codes
1484 %-----
1485 \bool_if:NTF \g_sdaps_last_page_bool {
1486 
1487   \int_compare:nNnTF { \g_sdaps_page_int } = { 1 } {
1488     % if we have a one page document, just always stamp the last
1489     % page. This means that SDAPS can fall back to simplex mode
1490     % automatically.
1491     \sdaps_draw_codes:
1492   }
1493 
1494   \bool_if:NTF \g_sdaps_twoside_bool {

```

```

1495     \int_if_odd:VT \g_sdaps_page_int {
1496         % This should no happen; draw a barcode though to keep things
1497         % somewhat sane.
1498         \msg_warning:nn { sdapsbase } { odd_page_count }
1499         \sdaps_draw_codes:
1500     } {
1501         % Even page (i.e. back) has a barcode except in "front" mode
1502         \str_if_eq:VnF \g_sdaps_twoside_barcode_tl { front } {
1503             \sdaps_draw_codes:
1504         }
1505     }
1506 }
1507     \sdaps_draw_codes:
1508 }
1509 }
1510
1511     % TODO: Check whether the page count is as expected?
1512 }
1513     \bool_if:NTF \g_sdaps_twoside_bool {
1514         \str_if_eq:VnTF \g_sdaps_twoside_barcode_tl { both } {
1515             \sdaps_draw_codes:
1516         } {
1517             \str_if_eq:VnTF \g_sdaps_twoside_barcode_tl { front } {
1518                 \int_if_odd:VT \g_sdaps_page_int {
1519                     \sdaps_draw_codes:
1520                 }
1521                 % And back is the only thing left
1522                 \int_if_odd:VF \g_sdaps_page_int {
1523                     \sdaps_draw_codes:
1524                 }
1525             }
1526         }
1527     }
1528     \sdaps_draw_codes:
1529 }
1530 }
1531
1532 %-----
1533 % watermark for non final mode
1534 %-----
1535 \bool_if:NT \g_sdaps_draft_bool {
1536     \node [rotate=60,scale=10,text~opacity=0.2,color=red]
1537         at (current~page.center) {\textsc{draft}};
1538     }
1539 \end{tikzpicture}
1540
1541     \group_end:
1542 }
1543     \int_gincr:N\g_sdaps_page_int
1544 }

```

```
1545 }
1546
1547
```

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.

```
1548
1549 \bool_new:N \g__sdaps_last_page_bool
1550
1551 \cs_new_protected_nopar:Nn \sdaps_begin: {
1552   \bool_gset:Nn \g__sdaps_last_page_bool \c_false_bool
1553
1554   % TODO: We really want to make sure nobody modifies the values after \sdaps_begin:
1555   \sdaps_info_write:x{Duplex=\bool_if:NTF \g_sdaps_twoside_bool {true} {false}}
1556   \sdaps_info_write:x{PrintQuestionnaireId=\bool_if:NTF \g_sdaps_print_questionnaire_id_bool {1}
1557   \sdaps_info_write:x{
1558     PageSize=\the\paperwidth, \the\paperheight
1559   }
1560   \sdaps_info_write:x{Style=\g_sdaps_style_t1}
1561   \sdaps_info_write:x{CheckMode=\g_sdaps_checkmode_t1}
1562   \sdaps_info_write:x{GlobalID=\g_sdaps_global_id_t1}
1563   \sdaps_info_write:x{GlobalIDLabel=\g_sdaps_global_id_label_t1}
1564
1565   \int_gset:Nn \g_sdaps_page_int { 0 }
1566 }
1567
1568 \cs_new_protected_nopar:Nn \sdaps_end: {
1569   % Note that using \sdaps_info_write_x:n may not work in some cases.
1570   % For this reason we write the out the Pages counter after each page,
1571   % which is fine to do.
1572   % Note that this means that the below "hack" to make onesided documents
1573   % work even in twoside (duplex) mode may not always work either.
1574
1575   \bool_gset:Nn \g__sdaps_last_page_bool \c_true_bool
1576 }
1577
1578
1579 %
```

Change History

v0.1
General: Initial version 1