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 LaTeX3 language internally, so we need to enable it.

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

And we need a number of other packages.

```
\ExplSyntaxOff

\input{sdapscode128}

\RequirePackage{qrcode}

\RequirePackage{tikz}
\usetikzlibrary{calc}
\usetikzlibrary{positioning}
\usetikzlibrary{decorations.pathmorphing}
\ExplSyntaxOn


% Define aliases for code128 functions, and generate variants
\cs_new_eq:NN \code_render:n \code
\cs_generate_variant:Nn \code_render:n { x, V }

% Also define an alias for the qrcode renderer
\cs_new_protected_nopar:Nn \qrcode_render:nn {
```

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

```
  \qrcode[#1] {#2}
}
\cs_generate_variant:Nn \qrcode_render:nn { nx, nV }


% Define a method to check for RTL languages, as the relevant commands
% may not always be defined.
\prg_new_conditional:Npnn \sdaps_if_rtl: { p, T, F, TF }
{
  \cs_if_exist:cTF { if@RTL } {
    \tl_use:c { if@RTL }
      \prg_return_true:
    \else
      \prg_return_false:
    \fi
  } {
    \prg_return_false:
  }
}

```

2.1 Context Handling

When creating complex questionnaires we need a mechnism 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.

```

\cs_generate_variant:Nn \tl_if_eq:nnTF { Vn }
\cs_generate_variant:Nn \tl_if_eq:nnT { Vn }
\cs_generate_variant:Nn \tl_if_eq:nnF { Vn }
\cs_generate_variant:Nn \int_if_odd:nTF { V }
\cs_generate_variant:Nn \int_if_odd:nF { V }
\cs_generate_variant:Nn \int_if_odd:nT { V }
\cs_generate_variant:Nn \tl_set:Nn { Nv }
\cs_generate_variant:Nn \msg_error:nnn { nnV }
\cs_generate_variant:Nn \exp_not:n { f }


\tl_new:N \l__sdaps_tmpa_tl
\tl_new:N \l__sdaps_tmpb_tl

\dim_new:N \l__sdaps_tmpa_dim %
\dim_new:N \l__sdaps_tmpb_dim %

\prop_new:N \g__sdaps_current_context_prop
\tl_new:N \g__sdaps_current_context_id_tl

```

```
\tl_new:N \g__sdaps_current_context_tl

\seq_new:N \g__sdaps_context_ids_seq
\seq_new:N \g__sdaps_contexts_seq

% Global metadata write enable variable managed by context
\bool_new:N \g_sdaps_write_enable_bool
\bool_gset_false:N \g_sdaps_write_enable_bool

\cs_new_protected_nopar:Nn \_sdaps_context_to_tl:N
{
  \tl_set:Nx #1 {_write_enable=\bool_if:NTF\g_sdaps_write_enable_bool{\c_true_bool}{\c_false_bo
  \prop_map_inline:Nn \g__sdaps_current_context_prop {
    % Could we remove some of the braces in the TL?
    \tl_if_eq:nnTF { \undefined } { ##2 } {
      \tl_put_right:Nn #1 {,{##1}}
    } {
      \tl_put_right:Nn #1 {,{##1}={##2}}
    }
  }
}


% Create new context using given identifier
\cs_new_protected_nopar:Nn \sdaps_context_begin:n
{
  % We need to serialize the current context and save it away.

  \group_begin:
    % Serialize the current context
    \_sdaps_context_to_tl:N \l__sdaps_tmpa_tl
    \tl_gset:NV \g__sdaps_current_context_tl \l__sdaps_tmpa_tl
    % Stuff it away in our sequence
    \seq_gput_left:NV \g__sdaps_contexts_seq \g__sdaps_current_context_tl
    \seq_gput_left:NV \g__sdaps_context_ids_seq \g__sdaps_current_context_id_tl

    % Clear the hooks
    \sdaps_context_put:nn { _context_hook_end } {}
    \sdaps_context_put:nn { _context_hook_post_end } {}

    \tl_gset:Nn \g__sdaps_current_context_id_tl { #1 }
  \group_end:
}

\msg_new:nnn { sdapsbase } { context_end_none_left } { There ~ is ~ no ~ context ~ to ~ end ~ l
\msg_new:nnn { sdapsbase } { context_end_broken } { The ~ current ~ context ~ with ~ id ~ #1 ~

\cs_new_protected_nopar:Nn \__sdaps_context_end:
{
  \seq_if_empty:NTF \g__sdaps_context_ids_seq {
    \msg_error:nn { sdapsbase } { context_end_none_left }
  } {

    \group_begin:
    \sdaps_context_get:nN { _context_hook_end } \l_tmpa_tl
    \tl_if_eq:VnF \l_tmpa_tl { \q_no_value } { \tl_use:N \l_tmpa_tl }

    % Grab post end hook
    \sdaps_context_get:nN { _context_hook_post_end } \l_tmpa_tl

    \_sdaps_context_clear:
    \seq_gpop_left:NN \g__sdaps_contexts_seq \g__sdaps_current_context_tl
    \seq_gpop_left:NN \g__sdaps_context_ids_seq \l__sdaps_tmpa_tl

    % Unpack context token list
    \sdaps_context_set:V \g__sdaps_current_context_tl

    \sdaps_context_get:nN { _write_enable } \l_tmpb_tl
    \bool_gset:Nn \g_sdaps_write_enable_bool { \l_tmpb_tl }
    \sdaps_context_remove:n { _write_enable }

    \tl_gclear:N \g__sdaps_current_context_tl
    \tl_gset:NV \g__sdaps_current_context_id_tl \l__sdaps_tmpa_tl

    \tl_if_eq:VnF \l_tmpa_tl { \q_no_value } { \tl_use:N \l_tmpa_tl }
    \group_end:
  }
}

\bool_new:N \l__sdaps_tmp_bool

\cs_new_protected_nopar:Nn \__sdaps_test_context_id:n
{
  \tl_if_eq:VnTF \g__sdaps_current_context_id_tl { #1 } {
    \bool_set:Nn \l__sdaps_tmp_bool \c_true_bool
  } {
    \bool_set:Nn \l__sdaps_tmp_bool \c_false_bool
  }
}

% Exit first context with passed in identifier
\cs_new_protected_nopar:Nn \sdaps_context_end:n
{
  \__sdaps_test_context_id:n { #1 }

  \bool_until_do:nn { \l__sdaps_tmp_bool } {
    \sdaps_context_end:

    \__sdaps_test_context_id:n { #1 }
  }
  \sdaps_context_end:
}

\cs_new_protected_nopar:Nn \__sdaps_context_end_local_scope:
{
  \__sdaps_test_context_id:n { sdaps_local_scope }

  \bool_until_do:nn { \l__sdaps_tmp_bool } {
    \__sdaps_context_end:

    \__sdaps_test_context_id:n { sdaps_local_scope }
  }
  \__sdaps_context_end:
}

% Exit current context
\cs_new_protected_nopar:Nn \sdaps_context_end:
{
  % Ensure the current context is not a local group
  \tl_if_eq:VnTF \g__sdaps_current_context_id_tl { sdaps_local_scope } {
    \msg_error:nnV { sdapsbase } { context_end_broken } \g__sdaps_current_context_id_tl
  } {}

  \__sdaps_context_end:
}

% Create new context using an empty name
\cs_new_protected_nopar:Nn \sdaps_context_begin:
{
  \sdaps_context_begin:n {}
}

\cs_new_protected_nopar:Nn \sdaps_context_begin_local:
{
  % Create a new context which will automatically be destroyed at the end of
  % the current TeX group.
  \sdaps_context_begin:n { sdaps_local_scope }
  \group_insert_after:N \__sdaps_context_end_local_scope:
}

\cs_new_protected_nopar:Nn \sdaps_context_put:n
{
  \sdaps_context_put:nn { #1 } { \undefined }
}

\cs_new_protected_nopar:Nn \sdaps_context_remove:n
{
  \prop_gremove:Nn \g__sdaps_current_context_prop { #1 }
}

```

```
\msg_new:nnn { sdapsbase } { context_key_broken } { Keys ~ may ~ not ~ contain ~ any ~ special
% Directly set a certain key
\cs_new_protected_nopar:Nn \sdaps_context_put:nn
{
  % TODO: How can I ensure that {} are not contained?
  % Though it would not be that bad actually.
  \tl_if_in:nnTF {#1} {,} {
    \msg_error:nnn { sdapsbase } { context_key_broken } {#1}
  } {
  }

  \tl_if_in:nnTF {#1} {=} {
    \msg_error:nnn { sdapsbase } { context_key_broken } {#1}
  } {
  }

  \prop_gput:Nnn \g__sdaps_current_context_prop { #1 } { #2 }
}
\cs_generate_variant:Nn \sdaps_context_put:nn { nV }

% Set a set of keys using comma separated list of key/value pairs
\cs_new_protected_nopar:Nn \sdaps_context_set:n
{
  \keyval_parse:NNn \sdaps_context_put:n \sdaps_context_put:nn { #1 }
}
\cs_generate_variant:Nn \sdaps_context_set:n {V}

\cs_new_protected_nopar:Nn \sdaps_context_get:nN
{
  \prop_get:NnN \g__sdaps_current_context_prop { #1 } #2
}

\cs_new_protected_nopar:Nn \sdaps_context_gget:nN
{
  \prop_get:NnN \g__sdaps_current_context_prop { #1 } \l_tmpa_tl
  \tl_gset_eq:NN #2 \l_tmpa_tl
}

\cs_new_protected_nopar:Nn \sdaps_context_append:nnn
{
  \sdaps_context_get:nN { #1 } \l__sdaps_tmpa_tl
  \tl_if_eq:VnTF \l__sdaps_tmpa_tl { \q_no_value } {
    \sdaps_context_put:nn { #1 } { #2 }
  } {
    \tl_put_right:Nn \l__sdaps_tmpa_tl { #3 }
    \tl_put_right:Nn \l__sdaps_tmpa_tl { #2 }
    \sdaps_context_put:nV { #1 } \l__sdaps_tmpa_tl
  }
}
\cs_generate_variant:Nn \sdaps_context_append:nnn { nVn }
```

```


\cs_new_protected_nopar:Nn \sdaps_context_append:nn
{
  \sdaps_context_append:nnn { #1 } { #2 } { , }
}

\cs_new_protected_nopar:Nn \sdaps_context_hook_end:n
{
  \sdaps_context_append:nnn { _context_hook_end } { #1 } { }
}

\cs_new_protected_nopar:Nn \sdaps_context_hook_post_end:n
{
  \sdaps_context_append:nnn { _context_hook_post_end } { #1 } { }
}

\cs_new_protected_nopar:Nn \sdaps_context_enable_writing:
{
  \bool_gset_true:N \g_sdaps_write_enable_bool
}

\cs_new_protected_nopar:Nn \sdaps_context_disable_writing:
{
  \bool_gset_false:N \g_sdaps_write_enable_bool
}

\cs_new_protected_nopar:Nn \_sdaps_context_clear:
{
  \prop_gclear:N \g__sdaps_current_context_prop
}

\cs_new:Nn \sdaps_context_map_function:N
{
  \prop_map_function:NN \g__sdaps_current_context_prop #1
}

\cs_new_protected_nopar:Nn \__sdaps_get:Nn
{
  \prop_get:NnN \g__sdaps_current_context_prop { #2 } #1
}

\cs_new_protected_nopar:Nn \__sdaps_get_empty:Nn
{
  \prop_get:NnN \g__sdaps_current_context_prop { #2 } #1
  \tl_if_eq:VnT #1 { \q_no_value } {
    \tl_set:Nn #1 {}
  }
}

```

```
\cs_new_protected_nopar:Nn \__sdaps_append_from_context:nN
{
  \prop_get:NnN \g__sdaps_current_context_prop { #1 } \l__sdaps_tmpb_tl
  \tl_if_eq:VnF \l__sdaps_tmpb_tl { \q_no_value } {
    \tl_put_right:Nn #2 {,}
    \tl_put_right:NV #2 {\l__sdaps_tmpb_tl}
  }
}
\cs_generate_variant:Nn \__sdaps_append_from_context:nN { VN }


```

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.

```

\seq_new:N \g__sdaps_object_id_seq
\int_new:N \g__sdaps_object_id_int
\int_gzero:N \g__sdaps_object_id_int

\cs_new_protected_nopar:Nn \_sdaps_qobject_end_hook:
{
  % Take the current implicit variable
  \sdaps_context_get:nN { _implicit_var } \l__sdaps_tmpa_tl

  % Prepend explicit variable name; we assume that either _implicit_var or _var
  % have a proper value.
  \sdaps_context_get:nN { _var } \l__sdaps_tmpb_tl
  \tl_if_eq:VnF \l__sdaps_tmpb_tl { \q_no_value } {
    \tl_if_eq:VnTF \l__sdaps_tmpa_tl { \q_no_value } {
      \tl_clear:N \l__sdaps_tmpa_tl
    } {
      \tl_put_left:Nn \l__sdaps_tmpa_tl { _ }
    }
    \tl_put_left:NV \l__sdaps_tmpa_tl \l__sdaps_tmpb_tl
  }

  \sdaps_context_get:nN { id } \l__sdaps_tmpb_tl

  \tl_if_eq:VnTF \l__sdaps_tmpb_tl { \q_no_value } {
    \msg_warning:nn { sdapsbase } { no_qid }
  } {
    \sdaps_info_write_x:x{
      Variable[\l__sdaps_tmpb_tl]=\l__sdaps_tmpa_tl
    }
  }
}

\cs_new_protected_nopar:Nn \_sdaps_qobject_post_hook:
{
  \seq_gpop_right:NN \g__sdaps_object_id_seq \l__sdaps_tmpa_tl
  \int_gset:NV \g__sdaps_object_id_int \l__sdaps_tmpa_tl
}

\cs_new_protected_nopar:Nn \sdaps_qobject_begin:nnn
{
  \int_gincr:N \g__sdaps_object_id_int
  \seq_gput_right:NV \g__sdaps_object_id_seq \g__sdaps_object_id_int
  \tl_set:Nx \l__sdaps_tmpa_tl { \int_use:N \g__sdaps_object_id_int }
  \int_gzero:N \g__sdaps_object_id_int

  \tl_set:Nx \l__sdaps_tmpb_tl { \seq_use:Nn \g__sdaps_object_id_seq {.} }

  \sdaps_context_begin:n {#1}
    \sdaps_context_put:nV {id} \l__sdaps_tmpb_tl
    \sdaps_info_write:x {QObject-#2=\tl_use:N\l__sdaps_tmpb_tl. ~ \exp_not:n{#3}}
    \sdaps_context_append:nVn { _implicit_var } \l__sdaps_tmpa_tl { _ }
    \sdaps_context_hook_end:n { \_sdaps_qobject_end_hook: }
    \sdaps_context_hook_post_end:n { \_sdaps_qobject_post_hook: }
}
\cs_generate_variant:Nn \sdaps_qobject_begin:nnn { nnV, nVV, nVn }


\cs_new_protected_nopar:Nn \sdaps_qobject_end:n
{
  % End the context in question, everything else is done from the close hook
  \sdaps_context_end:n {#1}
}

\cs_new_protected_nopar:Nn \sdaps_qobject_begin:nn
{
  \sdaps_qobject_begin:nnn { unnamed_qobject } { #1 } { #2 }
}

\cs_new_protected_nopar:Nn \sdaps_qobject_begin_local:nn
{
  % Empty local context which automatically closes the qobject
  \sdaps_context_begin_local:
    \sdaps_qobject_begin:nnn { unnamed_local_qobject } { #1 } { #2 }
}
```

```

\cs_new_protected_nopar:Nn \sdaps_qobject_end:
{
  \sdaps_qobject_end:n { unnamed_qobject }
}

\cs_new_protected_nopar:Nn \sdaps_qobject_append_var:n
{
  % If the given variable name starts with _ then include the implicitly
  % generated variable name.
  \tl_if_head_eq_charcode:nNTF { #1 } _ {
    \sdaps_context_get:nN { _implicit_var } \l__sdaps_tmpa_tl
    \tl_if_eq:VnF \l__sdaps_tmpa_tl { \q_no_value } {
      \sdaps_context_append:nVn { _var } \l__sdaps_tmpa_tl { _ }
    }

    \sdaps_context_append:nnn { _var } { #1 } { }
  } {
    \sdaps_context_append:nnn { _var } { #1 } { _ }
  }

  % We have a proper variable name now, delete the implicit one
  \sdaps_context_remove:n { _implicit_var }
}
\cs_generate_variant:Nn \sdaps_qobject_append_var:n { V }

\msg_new:nnn { sdapsbase } { no_qid } { Trying~to~output~metadata~but~no~question~ID~is~set~on~

\cs_new_protected_nopar:Nn \sdaps_answer:n
{
  \sdaps_context_get:nN { id } \l__sdaps_tmpa_tl

  \tl_if_eq:VnTF \l__sdaps_tmpa_tl { \q_no_value } {
    \msg_warning:nn { sdapsbase } { no_qid }
  } {
    \sdaps_info_write:x {
       Answer[\tl_use:N \l__sdaps_tmpa_tl]=\exp_not:n { #1 }
    }
  }
}
\cs_generate_variant:Nn \sdaps_answer:n { o }
\cs_generate_variant:Nn \sdaps_answer:n { f }
\cs_generate_variant:Nn \sdaps_answer:n { V }

\cs_new_protected_nopar:Nn \sdaps_range:nnn
{
  \sdaps_context_get:nN { id } \l__sdaps_tmpa_tl

  \tl_if_eq:VnTF \l__sdaps_tmpa_tl { \q_no_value } {
    \msg_warning:nn { sdapsbase } { no_qid }
  } {
    \sdaps_info_write:x {
       Range-#1[\tl_use:N \l__sdaps_tmpa_tl]=\int_eval:n{#2},\exp_not:n { #3 }
    }
  }
}
\cs_generate_variant:Nn \sdaps_range:nnn { nno }
\cs_generate_variant:Nn \sdaps_range:nnn { nnf }
\cs_generate_variant:Nn \sdaps_range:nnn { nnV }

\cs_generate_variant:Nn \tl_if_head_eq_charcode:nNT { VN }

\cs_new_protected_nopar:Nn \_sdaps_generate_var:nN
{
  \tl_set:Nn \l__sdaps_tmpa_tl { #1 }

  % Generate a variable name if there is none (prepended with _ prefix)
  \tl_if_empty:VT \l__sdaps_tmpa_tl {
    \tl_set:Nx \l__sdaps_tmpa_tl { _ \int_eval:n { \g__sdaps_object_id_int + 1 } }
  }

  % Prepend any implicitly generated variable names if prefixed by _
  \tl_if_head_eq_charcode:VNT \l__sdaps_tmpa_tl _ {
    \sdaps_context_get:nN { _implicit_var } \l__sdaps_tmpb_tl
    \tl_if_eq:VnTF \l__sdaps_tmpb_tl { \q_no_value } {
      \tl_remove_once:Nn \l__sdaps_tmpa_tl { _ }
    } {
      \tl_put_left:NV \l__sdaps_tmpa_tl \l__sdaps_tmpb_tl
    }
  }

  % Prepend explicit variable name
  \sdaps_context_get:nN { _var } \l__sdaps_tmpb_tl
  \tl_if_eq:VnF \l__sdaps_tmpb_tl { \q_no_value } {
    \tl_put_left:Nn \l__sdaps_tmpa_tl { _ }
    \tl_put_left:NV \l__sdaps_tmpa_tl \l__sdaps_tmpb_tl
  }

  \tl_set:NV #2 \l__sdaps_tmpa_tl
}

\cs_new_protected_nopar:Nn \_sdaps_box_inc_object_id:
{
  \bool_if:NT \g_sdaps_write_enable_bool {
    \int_gincr:N \g__sdaps_object_id_int
  }
}

```

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},
}}
```

```

\cs_generate_variant:Nn \keyval_parse:NNn { NNV }
\cs_generate_variant:Nn \int_gset:Nn { NV }

\seq_new:N \g__sdaps_checkbox_overlays_seq
\seq_new:N \g__sdaps_textbox_overlays_seq


\prop_new:N \g__sdaps_id_to_overrides_prop
\prop_new:N \g__sdaps_overrides_prop
\prop_new:N \g__sdaps_id_overrides_prop

\cs_new_protected_nopar:Nn \__sdaps_questionnaire_overrides_set:nn
{
  \str_if_eq_x:nnTF { #1 } { * } {
    \__sdaps_parse_overrides:n{ #2 }
  } {
    \prop_put:Nnn \g__sdaps_id_to_overrides_prop { #1 } { #2 }
  }
}

\cs_new_protected_nopar:Nn \sdaps_overrides_init:n
{
  \keyval_parse:NNn \use_none:n \__sdaps_questionnaire_overrides_set:nn { #1 }
}


\cs_new_protected_nopar:Nn \__sdaps_overrides_set:nn
{
  \prop_gput:Nnn \g__sdaps_overrides_prop { #1 } { #2 }
}

\cs_new_protected_nopar:Nn \__sdaps_id_overrides_set:nn
{
  \prop_gput:Nnn \g__sdaps_id_overrides_prop { #1 } { #2 }
}

```

```
\cs_new_protected_nopar:Nn \__sdaps_parse_overrides:n
{
  \prop_gclear:N \g__sdaps_overrides_prop
  \keyval_parse:NNn \use_none:n \__sdaps_overrides_set:nn { #1 }
}

\tl_new:N \l__sdaps_set_qid_tl
\cs_new_protected_nopar:Nn \sdaps_set_questionnaire_id:n
{
  \tl_gset:Nn \g__sdaps_questionnaire_id_tl { #1 }
  \prop_gclear:N \g__sdaps_id_overrides_prop
  \prop_get:NnNT \g__sdaps_id_to_overrides_prop { #1 } \l__sdaps_set_qid_tl {
    \keyval_parse:NNV \use_none:n \__sdaps_id_overrides_set:nn \l__sdaps_set_qid_tl
  }
}
\cs_generate_variant:Nn \sdaps_set_questionnaire_id:n { V }


\cs_new_protected_nopar:Nn \__sdaps_append_override_options:Nnn
{
  % Global definition
  % First generic for all items
  \prop_get:NnNT \g__sdaps_overrides_prop { * } \l__sdaps_tmpa_tl {
    \tl_put_right:Nn #1 {,}
    \tl_put_right:NV #1 \l__sdaps_tmpa_tl
  }
  \tl_if_empty:nF { #2 } {
    % Items with same variable name
    \prop_get:NnNT \g__sdaps_overrides_prop { #2 } \l__sdaps_tmpa_tl {
      \tl_put_right:Nn #1 {,}
      \tl_put_right:NV #1 \l__sdaps_tmpa_tl
    }
    \tl_if_empty:nF { #3 } {
      % Items with same variable name and value
      \prop_get:NnNT \g__sdaps_overrides_prop { #2&&#3 } \l__sdaps_tmpa_tl {
        \tl_put_right:Nn #1 {,}
        \tl_put_right:NV #1 \l__sdaps_tmpa_tl
      }
    }
  }


  % Local (questionnaire ID specific) definition
  % First generic for all items
  \prop_get:NnNT \g__sdaps_id_overrides_prop { * } \l__sdaps_tmpa_tl {
    \tl_put_right:Nn #1 {,}
    \tl_put_right:NV #1 \l__sdaps_tmpa_tl
  }
  \tl_if_empty:nF { #2 } {
    % Items with same variable name
    \prop_get:NnNT \g__sdaps_id_overrides_prop { #2 } \l__sdaps_tmpa_tl {
      \tl_put_right:Nn #1 {,}
      \tl_put_right:NV #1 \l__sdaps_tmpa_tl
    }
    \tl_if_empty:nF { #3 } {
      % Items with same variable name and value
      \prop_get:NnNT \g__sdaps_id_overrides_prop { #2&#3 } \l__sdaps_tmpa_tl {
        \tl_put_right:Nn #1 {,}
        \tl_put_right:NV #1 \l__sdaps_tmpa_tl
      }
    }
  }
}
\cs_generate_variant:Nn \__sdaps_append_override_options:Nnn  { NVn }

```

First we define constants and global variables for later use.

```
\dim_new:N \g_sdaps_linewidth_dim
\g_sdaps_linewidth_dim=1bp
\tl_new:N \g__sdaps_checkbox_last_info_tl

\int_new:N \g__sdaps_textbox_num_int
\int_set:Nn \g__sdaps_textbox_num_int 0

\tl_new:N \l_sdaps_var_tl
\dim_new:N \l_sdaps_x_dim
\dim_new:N \l_sdaps_y_dim
\dim_new:N \l_sdaps_width_dim
\dim_new:N \l_sdaps_height_dim

```

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.

```

\iow_new:N \g_sdaps_infofile_iow
\iow_open:Nn \g_sdaps_infofile_iow { \c_sys_jobname_str . sdaps }
\int_new:N \g__sdaps_infofile_line_int
\int_gset:Nn \g__sdaps_infofile_line_int { 0 }

\cs_new_protected_nopar:Nn \sdaps_info_write:n
{
  \bool_if:NT \g_sdaps_write_enable_bool {
    \int_gincr:N \g__sdaps_infofile_line_int
    \iow_shipout:Nx \g_sdaps_infofile_iow { [ \int_use:N \g__sdaps_infofile_line_int ] \exp_not
  }
}
\cs_generate_variant:Nn \sdaps_info_write:n { x }

\cs_new_protected_nopar:Nn \sdaps_info_write_x:n
```

```
{
  \bool_if:NT \g_sdaps_write_enable_bool {
    \int_gincr:N \g__sdaps_infofile_line_int
    \iow_shipout_x:Nx \g_sdaps_infofile_iow { [ \int_use:N \g__sdaps_infofile_line_int ] \exp_n
  }
}
\cs_generate_variant:Nn \sdaps_info_write_x:n { x }


```

Set some options at the beginning of the document.

```
%\AtBeginDocument{}
```

2.4 Definition for keyword parameters

```



\dim_new:N \l_sdaps_checkbox_linewidth_dim
\dim_new:N \l_sdaps_checkbox_width_dim
\dim_new:N \l_sdaps_checkbox_height_dim
\tl_new:N \l_sdaps_checkbox_form_tl
\tl_new:N \l_sdaps_checkbox_fill_tl
\tl_new:N \l_sdaps_checkbox_draw_tl
\tl_new:N \l_sdaps_checkbox_var_tl
\tl_new:N \l_sdaps_checkbox_value_tl
\bool_new:N \l_sdaps_checkbox_draw_check_bool

\tl_set:Nn \l_sdaps_checkbox_form_tl { box }

\tl_new:N \l_sdaps_parse_unknown_tl


% Internal overlays
\tl_new:N \l_sdaps_overlay_centered_text_tl
\tl_new:N \l_sdaps_overlay_minipage_text_tl
\tl_new:N \l_sdaps_overlay_minipage_pos_tl
\dim_new:N \l_sdaps_overlay_minipage_pad_dim

% Note that width/height is the *outside* width/height
\keys_define:nn { sdaps / checkbox }
{
  linewidth    .dim_set:N   = \l_sdaps_checkbox_linewidth_dim,
  linewidth    .initial:n   = 1bp,
  width        .dim_set:N   = \l_sdaps_checkbox_width_dim,
  width        .initial:n   = 3.5mm,
  height       .dim_set:N   = \l_sdaps_checkbox_height_dim,
  height       .initial:n   = 3.5mm,
  form         .choices:nn  = { box, ellipse } { \tl_set:Nx \l_sdaps_checkbox_form_tl { \l_keys
  value        .tl_set:N    = \l_sdaps_checkbox_value_tl,

  fill         .tl_set:N    = \l_sdaps_checkbox_fill_tl,
  fill         .initial:n   = { white },

  draw         .tl_set:N    = \l_sdaps_checkbox_draw_tl,
  draw         .initial:n   = { . },

  draw_check   .bool_set:N  = \l_sdaps_checkbox_draw_check_bool,
  draw_check   .default:n   = true,
  draw_check   .initial:n   = false,

  % Simple node overlay
  centered_text  .tl_set:N    = \l_sdaps_overlay_centered_text_tl,
  centered_text  .initial:n   = {},

  % minipage overlay
  text         .tl_set:N    = \l_sdaps_overlay_minipage_text_tl,
  text         .initial:n   = {},
  text_align   .tl_set:N    = \l_sdaps_overlay_minipage_pos_tl,
  text_align   .initial:n   = {c},
  text_padding .dim_set:N   = \l_sdaps_overlay_minipage_pad_dim,
  text_padding .initial:n   = {2bp},

  ellipse    .meta:n  = { form=ellipse },
  box        .meta:n  = { form=box },
}
```

2.5 Checkboxes

```

\cs_new_protected_nopar:Nn \__sdaps_checkbox_internal:nn
{
  \mbox{
    \sdaps_if_rtl:T {\beginL}
    \pdfsavepos

     % Position of page and baseline offset
    \dim_set:Nn \l_sdaps_x_dim { \hoffset }
    \dim_set:Nn \l_sdaps_y_dim { \voffset + \l_sdaps_checkbox_height_dim - \dim_eval:n { 0.5\l_

    % Size
    \dim_set:Nn \l_sdaps_width_dim { \l_sdaps_checkbox_width_dim }
    \dim_set:Nn \l_sdaps_height_dim { \l_sdaps_checkbox_height_dim }

    \bool_if:NT \g_sdaps_write_enable_bool {
      % pdflast[xy]pos is the PDF position of the baseline at the start of the box
      % excluding the page origin offset.
      \sdaps_context_get:nN {id} \l__sdaps_tmpa_tl
      \tl_if_eq:VnTF \l__sdaps_tmpa_tl { \q_no_value } {
        \msg_warning:nn { sdapsbase } { no_qid }
      } {
        \sdaps_info_write_x:x{
          Box[\l__sdaps_tmpa_tl]=Checkbox,
          \exp_not:n{\int_use:N\g_sdaps_page_int},
          \exp_not:n{\dim_eval:n} { \exp_not:f {\dim_use:N \l_sdaps_x_dim + \the\pdflastxpos sp
          \exp_not:n{\dim_eval:n} { \exp_not:f {\dim_use:N \l_sdaps_y_dim + \the\pdflastypos sp
          \dim_use:N \l_sdaps_width_dim,
          \dim_use:N \l_sdaps_height_dim,
          \tl_to_str:N\l_sdaps_checkbox_form_tl,
          \dim_use:N \l_sdaps_checkbox_linewidth_dim,
          #1,\tl_if_empty:nTF { #2 } { \int_use:N \g__sdaps_object_id_int } { #2 }
        }
      }
    }


    \tikz[baseline={0.5\l_sdaps_checkbox_height_dim-0.8ex}]{%
      \tl_if_eq:VnT \l_sdaps_checkbox_form_tl { box } {
        \draw[line~width=\l_sdaps_checkbox_linewidth_dim,fill=\l_sdaps_checkbox_fill_tl,draw=\l
      }
      \tl_if_eq:VnT \l_sdaps_checkbox_form_tl { ellipse } {
        \draw[line~width=\l_sdaps_checkbox_linewidth_dim,fill=\l_sdaps_checkbox_fill_tl,draw=\l
      }

      % For the overlay we actually position the nodes relative to the checkbox
      % and not absolute on the page.
      \dim_set:Nn \l_sdaps_x_dim { 0pt }
      \dim_set:Nn \l_sdaps_y_dim { \l_sdaps_checkbox_height_dim }

      % Use overlay so that nothing happens if a node is larger than the checkbox
      \begin{scope}[overlay]
        \seq_map_inline:Nn \g__sdaps_checkbox_overlays_seq {##1}
      \end{scope}
    }
    \sdaps_if_rtl:T {\endL}
  }
}
\cs_generate_variant:Nn \__sdaps_checkbox_internal:nn { Vn }

\sdaps_context_set:n { checkboxtype=multichoice }
\cs_new_protected_nopar:Nn \sdaps_checkbox_set_type:n
{
  \sdaps_context_set:n { checkboxtype={#1} }
}
\cs_generate_variant:Nn \sdaps_checkbox_set_type:n { V }

\cs_new_protected_nopar:Nn \sdaps_checkbox:nn
{
  \group_begin:%

```

```
    \_sdaps_generate_var:nN { #1 } \l_sdaps_var_tl

    \sdaps_context_get:nN { checkboxtype } \l__sdaps_tmpa_tl
    \tl_if_eq:VnTF \l__sdaps_tmpa_tl { multichoice } {
      \tl_set:Nn \l_sdaps_parse_unknown_tl { box }
    } {
      \tl_set:Nn \l_sdaps_parse_unknown_tl { ellipse }
    }

    \__sdaps_append_from_context:nN { * } \l_sdaps_parse_unknown_tl
    \__sdaps_append_from_context:VN \l__sdaps_tmpa_tl \l_sdaps_parse_unknown_tl
    \__sdaps_append_override_options:NVn \l_sdaps_parse_unknown_tl \l_sdaps_var_tl { #2 }

    \keys_set_known:nVN { sdaps / checkbox } \l_sdaps_parse_unknown_tl \l_sdaps_parse_unknown_t

    \_sdaps_box_inc_object_id:

    \__sdaps_checkbox_internal:Vn \l_sdaps_var_tl { #2 }
  \group_end:%
  \ignorespaces
}
\cs_generate_variant:Nn \sdaps_checkbox:nn { Vn, VV, nV }


\cs_new_protected_nopar:Nn \sdaps_overlay_check:
{
  \bool_if:NT \l_sdaps_checkbox_draw_check_bool {
    \begin{scope}[decoration={random~steps,segment~length=4pt,amplitude=1pt}]
      \draw[line~width=\l_sdaps_checkbox_linewidth_dim, decorate] ($(0, 0) - (2pt,2pt)$) -- (0.
      \draw[line~width=\l_sdaps_checkbox_linewidth_dim, decorate] ($(0, \l_sdaps_checkbox_heigh
    \end{scope}
  }
}
\seq_put_left:Nn \g__sdaps_checkbox_overlays_seq \sdaps_overlay_check:


\cs_new_protected_nopar:Nn \sdaps_overlay_centered:
{
  \tl_if_empty:NF \l_sdaps_overlay_centered_text_tl {
    \node[anchor=center,inner~sep=0pt,outer~sep=0pt] at ($(\l_sdaps_x_dim, \l_sdaps_y_dim) + 0.
      \l_sdaps_overlay_centered_text_tl
    };
  }
}
\seq_put_left:Nn \g__sdaps_checkbox_overlays_seq \sdaps_overlay_centered:
\seq_put_left:Nn \g__sdaps_textbox_overlays_seq \sdaps_overlay_centered:


\cs_new_protected_nopar:Nn \sdaps_overlay_minipage:
{
  \tl_if_empty:NF \l_sdaps_overlay_minipage_text_tl {
    \node[anchor=center,inner~sep=0pt,outer~sep=0pt] at ($(\l_sdaps_x_dim, \l_sdaps_y_dim) + 0.
      \dim_set:Nn \l_sdaps_width_dim { \l_sdaps_width_dim - 2\l_sdaps_overlay_minipage_pad_dim
      \dim_set:Nn \l_sdaps_height_dim { \l_sdaps_height_dim - 2\l_sdaps_overlay_minipage_pad_di

      \begin{minipage}[t][\l_sdaps_height_dim][\l_sdaps_overlay_minipage_pos_tl]{\l_sdaps_width
        % Hm, is this sane?
        \tex_let:D \textheight\l_sdaps_height_dim
        \l_sdaps_overlay_minipage_text_tl
      \end{minipage}
    };
  }
}
\seq_put_left:Nn \g__sdaps_checkbox_overlays_seq \sdaps_overlay_minipage:
\seq_put_left:Nn \g__sdaps_textbox_overlays_seq \sdaps_overlay_minipage:
```

2.6 Textboxes

```

\sdaps_context_set:n { textboxtype=textbox }
\cs_new_protected_nopar:Nn \sdaps_textbox_set_type:n
{
  \sdaps_context_set:n { textboxtype={#1} }
}
\cs_generate_variant:Nn \sdaps_textbox_set_type:n { V }

\dim_new:N  \l_sdaps_textbox_linewidth_dim
\tl_new:N   \l_sdaps_textbox_var_tl
\tl_new:N   \l_sdaps_textbox_fill_tl
\tl_new:N   \l_sdaps_textbox_draw_tl
\tl_new:N   \l__sdaps_textbox_boxtype_tl


\keys_define:nn { sdaps / textbox }
{
  linewidth    .dim_set:N   = \l_sdaps_textbox_linewidth_dim,
  linewidth    .initial:n   = 1bp,

  fill         .tl_set:N    = \l_sdaps_textbox_fill_tl,
  fill         .initial:n   = { white },

  draw         .tl_set:N    = \l_sdaps_textbox_draw_tl,
  draw         .initial:n   = { . },

  % Simple node overlay
  centered_text  .tl_set:N    = \l_sdaps_overlay_centered_text_tl,
  centered_text  .initial:n   = {},

  % minipage overlay
  text         .tl_set:N    = \l_sdaps_overlay_minipage_text_tl,
  text         .initial:n   = {},
  text_align   .tl_set:N    = \l_sdaps_overlay_minipage_pos_tl,
  text_align   .initial:n   = {c},
  text_padding .dim_set:N   = \l_sdaps_overlay_minipage_pad_dim,
  text_padding .initial:n   = {2bp},
}




\dim_new:N \l__sdaps_textbox_dp_dim
\dim_new:N \l__sdaps_textbox_ht_dim
\dim_new:N \l__sdaps_textbox_wd_dim
\dim_new:N \l__sdaps_textbox_pad_dim


\msg_new:nnn { sdapsbase } { textbox_wrong_mode } { Impossible~to~layout~a~#1~textbox~in~#2~mod

\cs_new_protected_nopar:Nn \__sdaps_textbox_prepare:n
{
  \tl_set:Nn \l_sdaps_parse_unknown_tl {}

  \_sdaps_generate_var:nN { #1 } \l_sdaps_var_tl

  \sdaps_context_get:nN { textboxtype } \l__sdaps_tmpa_tl
  \tl_if_eq:VnTF \l__sdaps_tmpa_tl { codebox } {
    \tl_set:Nn \l__sdaps_textbox_boxtype_tl { Codebox }
  } {
    \tl_set:Nn \l__sdaps_textbox_boxtype_tl { Textbox }
  }

  \__sdaps_append_from_context:nN { * } \l_sdaps_parse_unknown_tl
  \__sdaps_append_from_context:VN \l__sdaps_tmpa_tl \l_sdaps_parse_unknown_tl
  \__sdaps_append_override_options:NVn \l_sdaps_parse_unknown_tl \l_sdaps_var_tl { }

  \keys_set_known:nVN { sdaps / textbox } \l_sdaps_parse_unknown_tl \l_sdaps_parse_unknown_tl

  \_sdaps_box_inc_object_id:
}

\cs_new_protected_nopar:Nn \sdaps_textbox_vhstretch:nnn
{
```

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.

```
  \if_mode_math:
    \msg_error:nnnn { sdapsbase } { textbox_wrong_mode } { vhstretch } { math }
  \else:
  \tex_par:D
  \if_mode_horizontal:
    \msg_error:nnnn { sdapsbase } { textbox_wrong_mode } { vhstretch } { horizontal }
  \else:
  % Go into vertical mode by ending the current paragraph and insert
  % \cs{widowpenalty} so that we don't get an unwelcome page break. It is
  % assumed that any explanation for the textbox will be on top.
  \tex_penalty:D \tex_widowpenalty:D

  \group_begin:%

    \__sdaps_textbox_prepare:n { #1 }

```

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.

```
    % TODO: Make this configurable
    \dim_set:Nn \l__sdaps_textbox_dp_dim { 0.5ex }
    \dim_set:Nn \l__sdaps_textbox_ht_dim { 1.7ex }

    \vbox:n {
      \sdaps_if_rtl:T {\beginL}
      \leftskip=0pt
      \rightskip=0pt plus 1fill
      \noindent
      \tex_vrule:D height \l__sdaps_textbox_ht_dim depth 0pt width 0pt
      \pgfsys@markposition{textboxtop\int_use:N\g__sdaps_textbox_num_int}
      \sdaps_if_rtl:T {\endL}
    }

    \tex_penalty:D 10000
    \nointerlineskip

    \tex_cleaders:D\tex_hbox:D{}\skip_vertical:n{#2 - \l__sdaps_textbox_ht_dim}

    \tex_penalty:D 10000
    \nointerlineskip
```

```

  \tex_cleaders:D\tex_hbox:D{}\skip_vertical:n{\stretch{#3}}

  \tex_penalty:D 10000
  \nointerlineskip

  \vbox:n {
    \sdaps_if_rtl:T {\beginL}
    \noindent
    \leftskip=0pt plus 1fill
    \rightskip=0pt
    \begin{tikzpicture}[remember~picture,overlay,shift=(current~page.south~west)]
      \pgfsys@getposition{pgfpageorigin}{\@sdaps@pageorigin}
      \pgfsys@getposition{textboxtop\int_use:N\g__sdaps_textbox_num_int}{\@sdaps@textboxtoppo
      \pgfpointadd{\@sdaps@textboxtoppos}{
        \pgfpointadd{\@sdaps@pageorigin}{\pgfpoint{0}{\l__sdaps_textbox_ht_dim}}
      }
      \pgfgetlastxy{\l__sdaps_x}{\l__sdaps_y}
      \dim_set:Nn \l_sdaps_x_dim {\l__sdaps_x}
      \dim_set:Nn \l_sdaps_y_dim {\l__sdaps_y}

      \pgfsys@getposition{\pgfpictureid}{\@sdaps@textboxbottompos}
      \pgfpointdiff{\@sdaps@textboxtoppos}{\@sdaps@textboxbottompos}

      % Is there a more elegant way to multiply the height with -1?
      \pgfgetlastxy{\l__sdaps_width}{\l__sdaps_height}
      % Add the minimum height (i.e. depth below the last baseline) to the
      % overall height.
      \pgfpoint{\l__sdaps_width}{-\l__sdaps_height + \l__sdaps_textbox_ht_dim}
      \pgfgetlastxy{\l__sdaps_width}{\l__sdaps_height}
      \dim_set:Nn \l_sdaps_width_dim {\l__sdaps_width}
      \dim_set:Nn \l_sdaps_height_dim {\l__sdaps_height}

      % Draw the rectangle
      \draw[line~width=\l_sdaps_textbox_linewidth_dim,fill=\l_sdaps_textbox_fill_tl,draw=\l_s

      \begin{scope}
        \seq_map_inline:Nn \g__sdaps_textbox_overlays_seq {##1}
      \end{scope}

      \bool_if:NT \g_sdaps_write_enable_bool {
        \sdaps_context_get:nN {id} \l__sdaps_tmpa_tl
        \tl_if_eq:VnTF \l__sdaps_tmpa_tl { \q_no_value } {
          \msg_warning:nn { sdapsbase } { no_qid }
        } {
          \sdaps_info_write_x:x {
            Box[\l__sdaps_tmpa_tl]=\l__sdaps_textbox_boxtype_tl,
            \exp_not:n{\int_use:N\g_sdaps_page_int},
            \dim_use:N \l_sdaps_x_dim,
            \dim_use:N \l_sdaps_y_dim,
            \dim_use:N \l_sdaps_width_dim,
            \dim_use:N \l_sdaps_height_dim,
            \dim_use:N \l_sdaps_textbox_linewidth_dim,
            \tl_use:N \l_sdaps_var_tl,
          }
        }
      }
    \end{tikzpicture}
    \sdaps_if_rtl:T {\endL}
  }

  % We are done here, but we need the correct depth, so skip around a bit
  \tex_penalty:D 10000
  \nointerlineskip

  \skip_vertical:n{ - \l__sdaps_textbox_dp_dim}

  \tex_penalty:D 10000
  \nointerlineskip

  \hbox:n { \vrule depth \l__sdaps_textbox_dp_dim height 0pt width 0pt }

  \int_gincr:N\g__sdaps_textbox_num_int

 \group_end:
 \fi:
 \fi:
}
\cs_generate_variant:Nn \sdaps_textbox_vhstretch:nnn { Vnn }

\cs_new_protected_nopar:Nn \sdaps_textbox_vhstretch:nn
{
 \sdaps_textbox_vhstretch:nnn { #1 } { #2 } { 1 }
}


\cs_new_protected_nopar:Nn \sdaps_textbox_hstretch:nnnnn
{
 \group_begin:
  \sdaps_if_rtl:T {\beginL}

  \__sdaps_textbox_prepare:n { #1 }

  \dim_set:Nn \l_tmpa_dim { #2 }
  \dim_set:Nn \l_tmpb_dim { #3 }

  % Place a vrule to make space for the top/bottom padding
  \tex_vrule:D depth \dim_use:N \l_tmpa_dim height \dim_use:N \l_tmpb_dim width 0pt \nobreak
  \pgfsys@markposition{textboxstart\int_use:N\g__sdaps_textbox_num_int} \nobreak

```

```
\skip_horizontal:n { #4 + \stretch{#5} } \nobreak

% The textbox (rendered on the background)
\begin{tikzpicture}[remember~picture,overlay,shift=(current~page.south~west)]
  \pgfsys@getposition{pgfpageorigin}{\@sdaps@pageorigin}
  \pgfsys@getposition{textboxstart\int_use:N\g__sdaps_textbox_num_int}{\@sdaps@textboxpos}
  % The position here is the position of the baseline.
  % So move up by height (param 2) to get the correct vertical position.
  \pgfpointadd{\@sdaps@textboxpos}{
    \pgfpointadd{\@sdaps@pageorigin}{\pgfpoint{0}{ \dim_use:N \l_tmpb_dim}}
  }
  \pgfgetlastxy{\l__sdaps_x}{\l__sdaps_y}
  \dim_set:Nn \l_sdaps_x_dim {\l__sdaps_x}
  \dim_set:Nn \l_sdaps_y_dim {\l__sdaps_y}

  \pgfsys@getposition{\pgfpictureid}{\@sdaps@textboxendpos}
  % Calculate width and add the height to it
  \pgfpointadd{
    \pgfpointdiff{\@sdaps@textboxpos}{\@sdaps@textboxendpos}
  }{\pgfpoint{0pt}{\dim_use:N \l_tmpa_dim +  \dim_use:N \l_tmpb_dim}}
  \pgfgetlastxy{\l__sdaps_width}{\l__sdaps_height}
  \dim_set:Nn \l_sdaps_width_dim {\l__sdaps_width}
  \dim_set:Nn \l_sdaps_height_dim {\l__sdaps_height}

  % Draw the rectangle
  \draw[line~width=\l_sdaps_textbox_linewidth_dim,fill=\l_sdaps_textbox_fill_tl,draw=\l_sda

  \begin{scope}
    \seq_map_inline:Nn \g__sdaps_textbox_overlays_seq {##1}
  \end{scope}

  \bool_if:NT \g_sdaps_write_enable_bool {
    \sdaps_context_get:nN {id} \l__sdaps_tmpa_tl
    \tl_if_eq:VnTF \l__sdaps_tmpa_tl { \q_no_value } {
      \msg_warning:nn { sdapsbase } { no_qid }
    } {
      \sdaps_info_write_x:x {
        Box[\l__sdaps_tmpa_tl]=\l__sdaps_textbox_boxtype_tl,
        \exp_not:n{\int_use:N\g_sdaps_page_int},
        \dim_use:N \l_sdaps_x_dim,
        \dim_use:N \l_sdaps_y_dim,
        \dim_use:N \l_sdaps_width_dim,
        \dim_use:N \l_sdaps_height_dim,
        \dim_use:N \l_sdaps_textbox_linewidth_dim,
        \tl_use:N \l_sdaps_var_tl,
      }
    }
  }
\end{tikzpicture}

```

```
    \int_gincr:N\g__sdaps_textbox_num_int

    \sdaps_if_rtl:T {\endL}
  \group_end:
}

\cs_new_protected_nopar:Nn \__sdaps_textbox_prepare_coffin:
{
  \dim_set:Nn \l__sdaps_textbox_ht_dim { \coffin_ht:N \l__sdaps_textbox_coffin }
  \dim_set:Nn \l__sdaps_textbox_dp_dim { \coffin_dp:N \l__sdaps_textbox_coffin }
  \dim_set:Nn \l__sdaps_textbox_wd_dim { \coffin_wd:N \l__sdaps_textbox_coffin }

  \hcoffin_set:Nn \l_tmpa_coffin {
    \tex_vrule:D depth 0pt height \dim_eval:n { \l__sdaps_textbox_ht_dim + \l__sdaps_textbox_pa

    \pdfsavepos

    \dim_set:Nn \l_sdaps_width_dim {\l__sdaps_textbox_wd_dim + 2\l__sdaps_textbox_pad_dim}
    \dim_set:Nn \l_sdaps_height_dim {\l__sdaps_textbox_ht_dim + \l__sdaps_textbox_dp_dim + 2\l_

    % pdflast[xy]pos is the PDF position of the top left corner excluding the
    % origin
    \bool_if:NT \g_sdaps_write_enable_bool {
      \sdaps_context_get:nN {id} \l__sdaps_tmpa_tl
      \tl_if_eq:VnTF \l__sdaps_tmpa_tl { \q_no_value } {
        \msg_warning:nn { sdapsbase } { no_qid }
      } {
        \sdaps_info_write_x:x {
          Box[\l__sdaps_tmpa_tl]=\l__sdaps_textbox_boxtype_tl,
          \exp_not:n{\int_use:N\g_sdaps_page_int},
          \exp_not:n{\dim_eval:n {\hoffset + \the\pdflastxpos sp}},
          \exp_not:n{\dim_eval:n} { \exp_not:n {\voffset + \the\pdflastypos sp} + \dim_use:N \l
          \dim_use:N \l_sdaps_width_dim,
          \dim_use:N \l_sdaps_height_dim,
          \dim_use:N \l_sdaps_textbox_linewidth_dim,
          \tl_use:N \l_sdaps_var_tl,
        }
      }
    }

    \dim_set:Nn \l_sdaps_x_dim { 0pt }
    \dim_set:Nn \l_sdaps_y_dim { \l__sdaps_textbox_ht_dim + \l__sdaps_textbox_pad_dim }

    % The textbox (rendered on the background)
    \begin{tikzpicture}[overlay]
      % Draw the rectangle
      \draw[line~width=\l_sdaps_textbox_linewidth_dim,fill=\l_sdaps_textbox_fill_tl,draw=\l_sda

      \begin{scope}
          \seq_map_inline:Nn \g__sdaps_textbox_overlays_seq {##1}%
      \end{scope}
    \end{tikzpicture}
    \skip_horizontal:n { \l__sdaps_textbox_pad_dim }
  }

  \hcoffin_set:Nn \l_tmpb_coffin {
    \skip_horizontal:n { \l__sdaps_textbox_pad_dim }
    \tex_vrule:D depth \l__sdaps_textbox_pad_dim height 0pt width 0pt
  }

  \coffin_join:NnnNnnnn \l_tmpa_coffin { r } { H } \l__sdaps_textbox_coffin { l } { H } { 0pt }
  \coffin_join:NnnNnnnn \l_tmpa_coffin { r } { b } \l_tmpb_coffin { l } { t } { 0pt } { 0pt }

  \coffin_set_eq:NN \l__sdaps_textbox_coffin \l_tmpa_coffin


  \int_gincr:N\g__sdaps_textbox_num_int
}

\coffin_new:N \l__sdaps_textbox_coffin
\cs_new_protected_nopar:Nn \sdaps_textbox_hbox:nnn
{
  \group_begin:

    \__sdaps_textbox_prepare:n { #1 }

    \hcoffin_set:Nn \l__sdaps_textbox_coffin { \tl_trim_spaces:n { #3 } }

    \dim_set:Nn \l__sdaps_textbox_pad_dim { #2 }

    \__sdaps_textbox_prepare_coffin:

    \coffin_typeset:Nnnnn \l__sdaps_textbox_coffin { l } { H } { 0pt } { 0pt }

  \group_end:
}

\cs_new_protected:Nn \sdaps_textbox_vbox:nnnn
{
  \group_begin:

    \__sdaps_textbox_prepare:n { #1 }

    \dim_set:Nn \l__sdaps_textbox_pad_dim { #3 }
    \dim_set:Nn \l__sdaps_textbox_wd_dim { #2 - 2\l__sdaps_textbox_pad_dim }

    \vcoffin_set:Nnn \l__sdaps_textbox_coffin { \l__sdaps_textbox_wd_dim } { \tl_trim_spaces:n

    \__sdaps_textbox_prepare_coffin:

```

```
    \coffin_typeset:Nnnnn \l__sdaps_textbox_coffin { l } { H } { 0pt } { 0pt }

  \group_end:
}

```

2.7 Page Marking

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

```

\int_new:N \g_sdaps_page_int
\int_set:Nn \g_sdaps_page_int { 0 }

% Disabling recognition disables all drawings related to automatic recognition
\bool_new:N \g_sdaps_recognition_bool
\bool_new:N \g_sdaps_draft_bool
\bool_new:N \g_sdaps_twoside_bool
\bool_new:N \g_sdaps_print_questionnaire_id_bool

\bool_set:Nn \g_sdaps_recognition_bool \c_true_bool
\bool_set:Nn \g_sdaps_draft_bool \c_true_bool
\bool_set:Nn \g_sdaps_twoside_bool \c_false_bool
\bool_set:Nn \g_sdaps_print_questionnaire_id_bool \c_false_bool



\tl_new:N \g_sdaps_style_tl
\tl_new:N \g_sdaps_checkmode_tl
\dim_new:N \g_sdaps_edge_left_margin_dim
\dim_new:N \g_sdaps_edge_right_margin_dim
\dim_new:N \g_sdaps_edge_top_margin_dim
\dim_new:N \g_sdaps_edge_bottom_margin_dim
\dim_new:N \g_sdaps_edge_marker_linewidth_dim
\dim_new:N \g_sdaps_edge_marker_length_dim

\dim_new:N \g_sdaps_classic_boxpad_dim
\dim_new:N \g_sdaps_classic_boxsize_dim


\tl_gset:Nn \g_sdaps_style_tl { code128 }
\tl_gset:Nn \g_sdaps_twoside_barcode_tl { both }
\tl_gset:Nn \g_sdaps_checkmode_tl { checkcorrect }
\dim_gset:Nn \g_sdaps_edge_left_margin_dim { 10mm }
\dim_gset:Nn \g_sdaps_edge_right_margin_dim { 10mm }
\dim_gset:Nn \g_sdaps_edge_top_margin_dim { 12mm }
\dim_gset:Nn \g_sdaps_edge_bottom_margin_dim { 12mm }
\dim_gset:Nn \g_sdaps_edge_marker_linewidth_dim { 1bp }
\dim_gset:Nn \g_sdaps_edge_marker_length_dim { 20mm }

```

```
\dim_gset:Nn \g_sdaps_classic_boxpad_dim { 3mm }
\dim_gset:Nn \g_sdaps_classic_boxsize_dim { 3.5mm }


\tl_new:N \g__sdaps_questionnaire_id_tl
\tl_gset:Nn \g__sdaps_questionnaire_id_tl { }

\tl_new:N \g_sdaps_questionnaire_id_label_tl
\tl_gset:Nn \g_sdaps_questionnaire_id_label_tl { }


\tl_new:N \g_sdaps_survey_id_tl
\tl_gset:Nn \g_sdaps_survey_id_tl { 32498923 }

\tl_new:N \g_sdaps_global_id_tl
\tl_gset:Nn \g_sdaps_global_id_tl { }

\tl_new:N \g_sdaps_global_id_label_tl
\tl_gset:Nn \g_sdaps_global_id_label_tl { }




% Settings for code128 barcodes (used in code128 style, who would have thought?)
\dim_new:N \c_sdaps_barcode_height_dim
\dim_gset:Nn \c_sdaps_barcode_height_dim {6.5mm}
% This is the same as the default
\dim_new:N \c_sdaps_barcode_bar_width_dim
\dim_gset:Nn \c_sdaps_barcode_bar_width_dim {0.33mm}
% Same as default. Barwidth is decreased for printing by this value.
\dim_new:N \c_sdaps_barcode_bcorr_dim
\dim_gset:Nn \c_sdaps_barcode_bcorr_dim {0.020mm}

% The padding on the left/right of a barcode. This is the distance that the
% barcodes will be printed from the cornermarks
% Choosen to be the same as the barcode height. Note that the Code-128
% standard requires a quiet zone of max(10*modulesize, ~6.4mm).
\dim_new:N \c_sdaps_barcode_hpad_dim
\dim_gset:Nn \c_sdaps_barcode_hpad_dim {6.5mm}

% This needs to be smaller than 6.5mm because otherwise the content is too close.
% We set it so that it forms a golden ratio 6.5mm*(sqrt(5/4)-0.5). This means
% we have about 4.5mm padding to the content.
\dim_new:N \c_sdaps_barcode_vpad_dim
\dim_gset:Nn \c_sdaps_barcode_vpad_dim {4.02mm}


\cs_new_protected_nopar:Nn \sdaps_draw_codes:
{
  \group_begin:
```

```
  \begin{scope}[line~width=\g_sdaps_edge_marker_linewidth_dim, shift={(current~page.south~west
    \str_if_eq:VnT \g_sdaps_style_tl { qr } {
      \tl_if_empty:VF \g__sdaps_questionnaire_id_tl {
        \begin{scope}[shift={($(\g_sdaps_edge_left_margin_dim, \g_sdaps_edge_bottom_margin_di
          \node(barcode)[anchor=south~west,outer~sep=0,inner~sep=0]{
            \qrcode_render:nV {nolink,version=2,level=H,padding,height=10mm} \g__sdaps_questi
          };
        \end{scope}
      }

      % We unconditionally print this barcode, it is required for the recognition
      % process.
      \begin{scope}[shift={($(\paperwidth, 0) + (-\g_sdaps_edge_right_margin_dim, \g_sdaps_ed
        \pgfmathsetbasenumberlength{4}
        \pgfmathdectobase{\paddedpage}{\int_use:N\g_sdaps_page_int}{10}% Yes, padding to 4 ch
        \node(barcode)[anchor=south~east,outer~sep=0,inner~sep=0]{
          \qrcode_render:nx {nolink,version=2,level=H,padding,height=10mm} {\tl_use:N\g_sdaps
        };
      \end{scope}

      \tl_if_empty:VF \g_sdaps_global_id_tl {
        \begin{scope}[shift={($(\paperwidth/2-\g_sdaps_edge_right_margin_dim/2+\g_sdaps_edge_
          \node(barcode)[anchor=south,outer~sep=0,inner~sep=0]{
            \qrcode_render:nV {nolink,version=2,level=H,padding,height=10mm} \g_sdaps_global_
          };
        \end{scope}
      }
    }

    \str_if_eq:VnT \g_sdaps_style_tl { code128 } {
      \tl_if_empty:VF \g__sdaps_questionnaire_id_tl {
        % TODO: do not hardcode the font!
        \ttfamily\footnotesize

        \begin{scope}[shift={($(\g_sdaps_edge_left_margin_dim, \g_sdaps_edge_bottom_margin_di
          \node(barcode)[anchor=south~west,outer~sep=0,inner~sep=0]{
            \X = \c_sdaps_barcode_bar_width_dim
            \bcorr = \c_sdaps_barcode_bcorr_dim
            \barheight = \c_sdaps_barcode_height_dim
            \code_render:V \g__sdaps_questionnaire_id_tl
          };
          \node[below=1mm~of~barcode,distance=0,anchor=north,outer~sep=0,inner~sep=0]{
            \tl_if_empty:VTF \g_sdaps_questionnaire_id_label_tl {
              \tl_use:N \g__sdaps_questionnaire_id_tl
            } {
              \tl_use:N \g_sdaps_questionnaire_id_label_tl
            }
          };
        \end{scope}
      }

      % We unconditionally print this barcode, it is required for the recognition
      % process.
      \begin{scope}[shift={($(\paperwidth, 0) + (-\g_sdaps_edge_right_margin_dim, \g_sdaps_ed
        \pgfmathsetbasenumberlength{4}
        \pgfmathdectobase{\paddedpage}{\int_use:N\g_sdaps_page_int}{10}% Yes, padding to 4 ch
        \node(barcode)[anchor=south~east,outer~sep=0,inner~sep=0]{
          \X = \c_sdaps_barcode_bar_width_dim
          \bcorr = \c_sdaps_barcode_bcorr_dim
          \barheight = \c_sdaps_barcode_height_dim
          \code_render:x {\tl_use:N\g_sdaps_survey_id_tl \paddedpage}
        };
        \node[below=1mm~of~barcode,distance=0,anchor=north,outer~sep=0,inner~sep=0]{
          % TODO: do not hardcode the font!
          \ttfamily\footnotesize

          \tl_use:N\g_sdaps_survey_id_tl\,\paddedpage
        };
      \end{scope}

      \tl_if_empty:VF \g_sdaps_global_id_tl {
        \begin{scope}[shift={($(\paperwidth/2-\g_sdaps_edge_right_margin_dim/2+\g_sdaps_edge_
          \node(barcode)[anchor=south,outer~sep=0,inner~sep=0]{
            \X = \c_sdaps_barcode_bar_width_dim
            \bcorr = \c_sdaps_barcode_bcorr_dim
            \barheight = \c_sdaps_barcode_height_dim
            \code_render:V \g_sdaps_global_id_tl
          };
          \node[below=1mm~of~barcode,distance=0,anchor=north,outer~sep=0,inner~sep=0]{
            % TODO: do not hardcode the font!
            \ttfamily\footnotesize

            \tl_if_empty:VTF \g_sdaps_global_id_label_tl {
              \tl_use:N \g_sdaps_global_id_tl
            } {
              \tl_use:N \g_sdaps_global_id_label_tl
            }
          };
        \end{scope}
      }
    }
  \end{scope}%
 \group_end:
}

\msg_new:nnn { sdapsbase } { classic_too_many_pages } { You~cannot~have~more~than~six~pages~wit
\msg_new:nnn { sdapsbase } { odd_page_count } { You~have~an~odd~number~of~pages,~this~does~not~

% This needs to be called once for each page!
\cs_new_protected_nopar:Nn \sdaps_page_end: {
```

```
\bool_if:NTF \g_sdaps_recognition_bool {
  \group_begin:

  \int_gincr:N\g_sdaps_page_int%
  \sdaps_info_write:x{Pages=\int_use:N\g_sdaps_page_int}

  \normalfont%

  \begin{tikzpicture}[remember~picture,overlay]%

    %-----------------------------------------------------------------------
    % corner marks and corner boxes in classic mode
    %-----------------------------------------------------------------------
    \begin{scope}[line~width=\g_sdaps_edge_marker_linewidth_dim, line~join=miter]
      \begin{scope}[shift={($(current~page.north~west) + (\g_sdaps_edge_left_margin_dim, -\g_
        \draw (0,-\g_sdaps_edge_marker_length_dim) -- (0, 0) -- (\g_sdaps_edge_marker_length_

        \str_if_eq:VnT \g_sdaps_style_tl { classic } {
          % TODO: Is the filled/non-filled mapping still correct?
          \int_case:nnTF \g_sdaps_page_int {
            { 1 } { \draw }
            { 2 } { \fill }
            { 3 } { \fill }
            { 4 } { \fill }
            { 5 } { \fill }
            { 6 } { \draw }
          } {
            (\g_sdaps_classic_boxpad_dim, -\g_sdaps_classic_boxpad_dim) rectangle +(\g_sdaps_
          } {
            % Error out (first time)
            \msg_error:nn { sdapsbase } { classic_too_many_pages }
          }
        }
      \end{scope}
      \begin{scope}[shift={($(current~page.north~east) + (-\g_sdaps_edge_right_margin_dim, -\
        \draw (0,-\g_sdaps_edge_marker_length_dim) -- (0,0) -- (-\g_sdaps_edge_marker_length_

        \str_if_eq:VnT \g_sdaps_style_tl { classic } {
          % TODO: Is the filled/non-filled mapping still correct?
          \int_case:nnT \g_sdaps_page_int {
            { 1 } { \fill }
            { 2 } { \fill }
            { 3 } { \draw }
            { 4 } { \draw }
            { 5 } { \draw }
            { 6 } { \draw }
          } {
            (-\g_sdaps_classic_boxpad_dim, -\g_sdaps_classic_boxpad_dim) rectangle +(-\g_sdap
          }
        }
      \end{scope}
      \begin{scope}[shift={($(current~page.south~west) + (\g_sdaps_edge_left_margin_dim, \g_s
        \draw (0,\g_sdaps_edge_marker_length_dim) -- (0, 0) -- (\g_sdaps_edge_marker_length_d

        \str_if_eq:VnT \g_sdaps_style_tl { classic } {
          % TODO: Is the filled/non-filled mapping still correct?
          \int_case:nnT \g_sdaps_page_int {
            { 1 } { \fill }
            { 2 } { \draw }
            { 3 } { \fill }
            { 4 } { \fill }
            { 5 } { \draw }
            { 6 } { \draw }
          } {
            (\g_sdaps_classic_boxpad_dim, \g_sdaps_classic_boxpad_dim) rectangle +(\g_sdaps_c
          }
        }
      \end{scope}
      \begin{scope}[shift={($(current~page.south~east) + (-\g_sdaps_edge_right_margin_dim, \g
        \draw (0,\g_sdaps_edge_marker_length_dim) -- (0mm, 0mm) -- (-\g_sdaps_edge_marker_len

        \str_if_eq:VnT \g_sdaps_style_tl { classic } {
          % TODO: Is the filled/non-filled mapping still correct?
          \int_case:nnT \g_sdaps_page_int {
            { 1 } { \fill }
            { 2 } { \draw }
            { 3 } { \fill }
            { 4 } { \draw }
            { 5 } { \draw }
            { 6 } { \fill }
          } {
            (-\g_sdaps_classic_boxpad_dim, \g_sdaps_classic_boxpad_dim) rectangle +(-\g_sdaps
          }
        }
      \end{scope}
    \end{scope}

    %-----------------------------------------------------------------------
    % barcodes/qr codes
    %-----------------------------------------------------------------------
    \bool_if:NTF \g__sdaps_last_page_bool {

      \int_compare:nNnTF { \g_sdaps_page_int } = { 1 } {
        % if we have a one page document, just always stamp the last
        % page. This means that SDAPS can fall back to simplex mode
        % automatically.
        \sdaps_draw_codes:
      } {

        \bool_if:NTF \g_sdaps_twoside_bool {
          \int_if_odd:VT \g_sdaps_page_int {
            % This should no happen; draw a barcode though to keep things
            % somewhat sane.
            \msg_warning:nn { sdapsbase } { odd_page_count }
            \sdaps_draw_codes:
          } {
            % Even page (i.e. back) has a barcode except in "front" mode
            \str_if_eq:VnF \g_sdaps_twoside_barcode_tl { front } {
              \sdaps_draw_codes:
            }
          }
        } {
          \sdaps_draw_codes:
        }
      }

      % TODO: Check whether the page count is as expected?
    } {
      \bool_if:NTF \g_sdaps_twoside_bool {
        \str_if_eq:VnTF \g_sdaps_twoside_barcode_tl { both } {
          \sdaps_draw_codes:
        } {
          \str_if_eq:VnTF \g_sdaps_twoside_barcode_tl { front } {
            \int_if_odd:VT \g_sdaps_page_int {
              \sdaps_draw_codes:
            }
          } { % And back is the only thing left
            \int_if_odd:VF \g_sdaps_page_int {
              \sdaps_draw_codes:
            }
          }
        }
      } {
        \sdaps_draw_codes:
      }
    }

    %-----------------------------------------------------------------------
    % watermark for non final mode
    %-----------------------------------------------------------------------
    \bool_if:NT \g_sdaps_draft_bool {
      \node [rotate=60,scale=10,text~opacity=0.2,color=red]
         at (current~page.center) {\textsc{draft}};
    }
  \end{tikzpicture}

  \group_end:
} {
  \int_gincr:N\g_sdaps_page_int
}
```

```
}


```

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.

```

\bool_new:N \g__sdaps_last_page_bool

\cs_new_protected_nopar:Nn \sdaps_begin: {
  \bool_gset:Nn \g__sdaps_last_page_bool \c_false_bool

  % TODO: We really want to make sure nobody modifies the values after \sdaps_begin:
  \sdaps_info_write:x{Duplex=\bool_if:NTF \g_sdaps_twoside_bool {true} {false}}
  \sdaps_info_write:x{PrintQuestionnaireId=\bool_if:NTF \g_sdaps_print_questionnaire_id_bool {1
  \sdaps_info_write:x{
    PageSize=\the\paperwidth, \the\paperheight
  }
  \sdaps_info_write:x{Style=\g_sdaps_style_tl}
  \sdaps_info_write:x{CheckMode=\g_sdaps_checkmode_tl}
  \sdaps_info_write:x{GlobalID=\g_sdaps_global_id_tl}
  \sdaps_info_write:x{GlobalIDLabel=\g_sdaps_global_id_label_tl}

  \int_gset:Nn \g_sdaps_page_int { 0 }
}

\cs_new_protected_nopar:Nn \sdaps_end: {
  % Note that using \sdaps_info_write_x:n may not work in some cases.
  % For this reason we write the out the Pages counter after each page,
  % which is fine to do.
  % Note that this means that the below "hack" to make onesided documents
  % work even in twoside (duplex) mode may not always work either.

  \bool_gset:Nn \g__sdaps_last_page_bool \c_true_bool
}


%
```

Change History

v0.1

General: Initial version 1