

ltx-talk – A class for typesetting presentations*

Joseph Wright[†]

Released 2025-09-18

Contents

I	ltx-talk – Overall set up	1
1	ltx-talk implementation	1
1.1	Set up	1
1.2	Additions for expl3	1
1.3	Extra variants	2
1.4	Scratch space	2
1.5	Option handling	3
1.6	Setting up	3
1.7	Math support	4
1.8	Font selection	4
1.9	Hyperlinks	5
1.10	Tagging	5
II	ltx-talk-color – Color definitions	6
1	ltx-talk-color implementation	6
1.1	Existing definitions	6
1.2	Document commands	6
1.3	Color definition	7
1.4	Semantic colors	7
III	ltx-talk-decode – Decoding overlay specs	8
1	ltx-talk-decode implementation	8
IV	ltx-talk-frame – The structure of frames	15

*This file describes v0.2.1, last revised 2025-09-18.

[†]E-mail: joseph@texdev.net

1	ltx-talk-frame implementation	15
1.1	Slides in frames	15
1.2	Counters	18
1.3	Frame options	19
1.4	Tagging for headers	19
1.5	Wallpaper	20
1.6	The <code>frame</code> environment	24
V	ltx-talk-frame – The structure of frames	27
1	ltx-talk-frame-structure implementation	27
1.1	Columns	27
1.2	Floats	29
VI	ltx-talk-mode – Modes	32
1	ltx-talk-mode implementation	32
VII	ltx-talk-overlay – Overlays	33
1	ltx-talk-overlay implementation	33
1.1	Utilities	33
1.2	Action commands and environments	33
1.3	Non-action commands and environments	37
1.4	Fixed-size areas	38
1.5	Adding overlays to existing commands	40
VIII	ltx-talk-required – “Required” definitions	43
1	ltx-talk-required implementation	43
1.1	Standard design settings	43
1.2	List support	44
IX	ltx-talk-structure – Structural commands	45
1	ltx-talk-structure implementation	45
1.1	Frame title	45
1.2	Sectioning	46
1.3	Table of contents	48
1.4	Block environments	50
1.5	Lists	51
1.6	Theorems, <i>etc.</i>	54
X	ltx-talk-title – Title pages	55

1	ltx-talk-title implementation	55
	Index	58

Part I

ltx-talk – Overall set up

1 ltx-talk implementation

Start the DocStrip guards.

```
1 <{*class>
    Identify the internal prefix.
2 <{@=talk>
```

1.1 Set up

Identify the package and give the over all version information.

```
3 \ProvidesExplClass {ltx-talk} {2025-09-18} {0.2.1}
4 {A class for typesetting presentations}
    Get the right type of message.
5 \prop_gput:Nnn \g_msg_module_name_prop { talk } { ltx-talk }
6 \prop_gput:Nnn \g_msg_module_type_prop { talk } { Class }
    Require the latest LATEX structures.
7 \IfFormatAtLeastF{2025-11-01}
8 {
9     \msg_new:nnnn { ltx-talk } { kernel-too-old }
10     { The~ltx-talk~class~requires~LaTeX~2025-11-01~or~later. }
11     {
12         You~have~tried~to~use~the~ltx-talk~class~with~a~LaTeX~kernel~release~
13         prior~to~2025-11-01;~the~required~functionality~is~missing. \\ \\
14         At~present,~you~may~need~to~use~the~development~release~of~LaTeX,~
15         invoked~using~"lualatex-dev"~or~"pdflatex-dev".
16     }
17     \msg_fatal:nn { ltx-talk } { kernel-too-old }
18 }
19 \NeedsDocumentMetadata
```

1.2 Additions for expl3

Like \vcoffin_set:Nnn, so should be an easy enough addition.

```
20 \cs_gset_protected:Npn \vbox_set_to_wd:Nnn #1#2#3
21 {
22     \tex_setbox:D #1 \tex_vbox:D
23     {
24         \tex_hsize:D \_box_dim_eval:n {#2}
25         \color_group_begin: #3 \par \color_group_end:
26     }
27     \box_dp:N #1 \_box_dim_eval:n {#2}
28 }
29 \cs_gset_protected:Npn \vbox_set_to_wd:Nnw #1#2
30 {
31     \cs_set_protected:Npn \_box_set_to_wd:
32     { \box_wd:N #1 \_box_dim_eval:n {#2} }
```

```

33 \tex_setbox:D #1 \tex_vbox:D
34 \c_group_begin_token
35 \tex_hsize:D \_box_dim_eval:n {#2}
36 \group_insert_after:N \_box_set_to_wd:
37 \color_group_begin:
38 }
    Some things from xbox that would be useful.
39 \cs_gset_protected:Npn \rule:nnn #1#2#3
40 {
41   \tex_vrule:D
42   height \dim_eval:n {#2} \exp_stop_f:
43   depth \dim_eval:n {#3} \exp_stop_f:
44   width \dim_eval:n {#1} \exp_stop_f:
45   \scan_stop:
46 }

```

1.3 Extra variants

```

47 \cs_generate_variant:Nn \clist_set:Nn { cv }
48 \cs_generate_variant:Nn \hook_gput_code:nnn { nne }
49 \exp_args_generate:n { nVv }
50 \cs_generate_variant:Nn \color_select:n { V }
51 \cs_generate_variant:Nn \dim_compare:nNnTF { v }
52 \cs_generate_variant:Nn \dim_compare_p:nNn { vNv }
53 \cs_generate_variant:Nn \dim_max:nn { v }
54 \cs_generate_variant:Nn \str_replace_all:Nnn { NnV }
55 \cs_generate_variant:Nn \text_purify:n { v }
56 \cs_generate_variant:Nn \vbox_to_ht:nn { v }

```

1.4 Scratch space

_talk_tmp:w For one-off processing.

```

57 \cs_new_protected:Npn \_talk_tmp:w { }

```

(End of definition for _talk_tmp:w.)

\l__talk_tmp_box

```

58 \box_new:N \l__talk_tmp_box

```

(End of definition for \l__talk_tmp_box.)

\l__talk_tmp_tl

```

59 \tl_new:N \l__talk_tmp_tl

```

(End of definition for \l__talk_tmp_tl.)

1.5 Option handling

```

\l__talk_aspect_ratio_str
\l__talk_fontsize_dim
\l__talk_frame_title_bool
\l__talk_mode_str
60 \keys_define:nn { talk }
61 {
62   aspect-ratio .str_set:N =
63     \l__talk_aspect_ratio_str ,
64   font-size .dim_set:N =
65     \l__talk_fontsize_dim ,
66   frame-title-arg .bool_set:N =
67     \l__talk_frame_title_bool ,
68   mode .choices:nn =
69     { handout , projector }
70     { \str_set:NV \l__talk_mode_str \l_keys_choice_tl }
71 }

```

(End of definition for `\l__talk_aspect_ratio_str` and others.)

Scope for options.

```

72 \keys_define:nn { talk }
73 {
74   aspect-ratio .usage:n = load ,
75   font-size .usage:n = load ,
76   frame-title-arg .usage:n = load ,
77   mode .usage:n = load
78 }

```

Initial values.

```

79 \keys_set:nn { talk }
80 {
81   aspect-ratio = 16:9 ,
82   font-size = 11pt ,
83   frame-title-arg = false ,
84   mode = projector
85 }

```

```

86 \ProcessKeyOptions [ talk ]

```

1.6 Setting up

Load the font size setup if available, otherwise fall back on scaling.

```

87 \file_if_exist_input:nF { size \dim_to_decimal:n \l__talk_fontsize_dim .clo }
88 {
89   \file_input:n { size10.clo }
90   \RequirePackage { relsize }
91   \hook_gput_code:nne { begindocument } { talk }
92     { \exp_not:N \relsize { \fp_eval:n { \l__talk_fontsize_dim / 10pt } } }
93 }

```

`\c__talk_paper_height_dim` `\c__talk_paper_width_dim` As geometry is being used to set the paper size with no previous value, we have to use the optional argument rather than waiting to apply `\geometry`.

```

94 \dim_const:Nn \c__talk_paper_height_dim { 100mm }
95 \use:e
96 {
97   \cs_set_protected:Npn \exp_not:N \__talk_tmp:w

```

```

98      #1 \tl_to_str:n { : } #2 \tl_to_str:n { : } #3 \exp_not:N \q_stop
99      {
100        \dim_const:Nn \exp_not:N \c__talk_paper_width_dim
101        {
102          \exp_not:N \fp_to_dim:n
103          { (#1 / #2) * \exp_not:N \c__talk_paper_height_dim }
104        }
105      }
106      \exp_not:N \__talk_tmp:w \l__talk_aspect_ratio_str
107      \tl_to_str:n { : } 100 \exp_not:N \q_stop
108    }
109  \use:e
110  {
111    \exp_not:N \RequirePackage
112    [
113      papersize =
114      {
115        \dim_use:N \c__talk_paper_width_dim ,
116        \dim_use:N \c__talk_paper_height_dim
117      } ,
118      tmargin    = 10mm ,
119      bmargin    =  8mm ,
120      lmargin    = 10mm ,
121      rmargin    = 10mm ,
122      headheight = 10mm ,
123      headsep    =  2mm ,
124      footskip   =  6mm
125    ]
126    { geometry }
127  }

(End of definition for \c__talk_paper_height_dim and \c__talk_paper_width_dim.)
Turn off justification
128 \raggedright

```

1.7 Math support

We always require `amsmath`: this is forced anyway by `unicode-math` for LuaTeX.

```

129 \RequirePackage { amsmath }

```

1.8 Font selection

The aim here is to minimize change from the standard font setup but at the same time provide a sans-serif default. Since `beamer` was released, better sans-serif math mode fonts have become available. For OpenType engines, requiring `unicode-math` is the most sensible approach. The New Computer Modern font provides a reasonable initial set of glyphs. It comes with a wrapper package, but that does various other things: if the user wants these, they can choose to load themselves. For 8-bit engines, switching the text font to be sans-serif is easy. For math mode, the `sansmathfonts` package does a good job: here, using the package rather than adjusting directly is the sensible option.

```

130 \sys_if_engine_opentype:TF
131 {

```

```

132 \RequirePackage { unicode-math }
133 \setsansfont { NewCMSans10-Regular.otf }
134 \setmathfont { NewCMSansMath-Regular.otf }
135 }
136 {
137 \RequirePackage { sansmathfonts }
138 \RequirePackage [ nomath ] { lmodern }
139 }
140 \cs_set_eq:NN \rmdefault \sfdefault

```

1.9 Hyperlinks

`\thepage` We define `\thepage` here: this is checked for by `hyperref` so has to come early.

```

141 \cs_new:Npn \thepage { \@arabic \c@page }

```

(End of definition for `\thepage`. This variable is documented on page ??.)

A requirement.

```

142 \RequirePackage { hyperref }
143 \hypersetup { hidelinks }

```

1.10 Tagging

We need to extend the standard tagging model to work with slides and so on.

```

144 \tagpdfsetup
145 {
146   role / user-NS = ltx-talk          ,
147   role / new-tag = frame / Sect      ,
148   role / new-tag = frametitle / H4
149 }
150 </class>

```


Part II

ltx-talk-color – Color definitions

1 ltx-talk-color implementation

Start the DocStrip guards.

```
1 < *class >
    Identify the internal prefix.
2 < @@=talk >
```

The aim here is to *test* how well l3color can support the range of color functions that are needed for a presentation. As such, this is very much experimental, but deliberately so. In particular, there is an important question about the need for global colors: used throughout beamer but otherwise not widely encountered. At the same time, there is a need to work with packages that expect color to be managed in a predictable way: pgf in particular makes use of xcolor internal as part of color management.

Currently, colors defined using xcolor will be passed on to l3color provided \DocumentMetadata is active. As that is a requirement in any case for ltx-talk, some of the setup is relatively easy to do.

1.1 Existing definitions

```
3 \RequirePackage { xcolor }

\stdcolor Save the document commands.
\stdmathcolor 4 \NewCommandCopy \stdcolor \color
\stdtextcolor 5 \NewCommandCopy \stdmathcolor \mathcolor
6 \NewCommandCopy \stdtextcolor \textcolor
```

(End of definition for \stdcolor, \stdmathcolor, and \stdtextcolor. These functions are documented on page ??.)

1.2 Document commands

```
7 \cs_generate_variant:Nn \color_select:n { e }
8 \cs_generate_variant:Nn \color_select:nn { ne }
9 \cs_generate_variant:Nn \color_math:nn { e }
10 \cs_generate_variant:Nn \color_math:nnn { ne }

\color Add the overlay specification and use l3color.
\mathcolor
\textcolor
11 \RenewDocumentCommand \color { D <> { all } o m }
12 {
13     \__talk_if_overlay:nT {#1}
14     {
15         \IfNoValueTF {#2}
16         { \color_select:e {#3} }
17         { \color_select:ne {#2} {#3} }
18     }
19 }
20 \RenewDocumentCommand \mathcolor { D <> { all } o m +m }
21 {
22     \__talk_if_overlay:nT {#1}
```

```

23     {
24         \IfNoValueTF {#2}
25         { \color_math:en {#3} {#4} }
26         { \color_math:nen {#2} {#3} {#4} }
27     }
28 }
29 \RenewDocumentCommand \textcolor { D <> { all } o m +m }
30 {
31     \__talk_if_overlay:nT {#1}
32     {
33         \mode_leave_vertical:
34         \group_begin:
35         \IfNoValueTF {#2}
36         { \color_select:e {#3} }
37         { \color_select:ne {#2} {#3} }
38         #4
39         \group_end:
40     }
41 }

```

(End of definition for `\color`, `\mathcolor`, and `\textcolor`. These functions are documented on page ??.)

1.3 Color definition

`\DeclareColor` Provide a single interface here: as the data will be passed to `l3color` in any case, there is not too much to do.

```

42 \NewDocumentCommand \DeclareColor { m o m }
43 {
44     \IfNoValueTF {#2}
45     { \colorlet {#1} {#3} }
46     { \definecolor {#1} {#2} {#3} }
47 }

```

(End of definition for `\DeclareColor`. This function is documented on page ??.)

1.4 Semantic colors

Pick up the standard colors from beamer.

```

48 \DeclareColor { alert } [ RGB ] { 200 , 0 , 0 }
49 \DeclareColor { example } { green!50!black }
50 \DeclareColor { structure } [ rgb ] { 0.2 , 0.2 , 0.7 }
51 </class>

```

Part III

ltx-talk-decode – Decoding overlay specs

1 ltx-talk-decode implementation

Start the DocStrip guards.

```
1 <*class>
    Identify the internal prefix.
2 <@@=talk>
```

`\l__talk_decode_overlays_bool` The result from decoding: are we on the current slide. This may well be better handled by moving to a TF signature: to be explored.

```
3 \bool_new:N \l__talk_decode_overlays_bool
```

(End of definition for \l__talk_decode_overlays_bool.)

`\g__talk_pauses_int` The automatically-incremented value for the relative overlay value.

```
\c@pauses 4 \int_new:N \g__talk_pauses_int
\thepauses 5 \cs_new_eq:NN \c@pauses \g__talk_pauses_int
6 \cs_new:Npn \thepauses { \@arabic \g__talk_pauses_int }
```

(End of definition for \g__talk_pauses_int, \c@pauses, and \thepauses. These variables are documented on page ??.)

`\l__talk_decode_pure_bool` Tracks whether only mode specifications were given.

```
7 \bool_new:N \l__talk_decode_pure_bool
```

(End of definition for \l__talk_decode_pure_bool.)

`\l__talk_decode_step_bool` Tracks whether to step `\g__talk_pauses_int`.

```
8 \bool_new:N \l__talk_decode_step_bool
```

(End of definition for \l__talk_decode_step_bool.)

`\l__talk_decode_arg_str` For error usage.

```
9 \str_new:N \l__talk_decode_arg_str
```

(End of definition for \l__talk_decode_arg_str.)

`\l__talk_decode_overlays_clist` The decoded overlay specification: will have only absolute slide numbers present, potentially along with ranges.

`\l__talk_decode_overlays_str`

```
10 \clist_new:N \l__talk_decode_overlays_clist
11 \str_new:N \l__talk_decode_overlays_str
```

(End of definition for \l__talk_decode_overlays_clist and \l__talk_decode_overlays_str.)

`\l__talk_decode_action_str` The action which is active, if any.

```
12 \str_new:N \l__talk_decode_action_str
```

(End of definition for \l__talk_decode_action_str.)

`\l__talk_decode_actions_bool` For the actions versions of overlay tracking.

```

\l__talk_decode_actions_clist
\l__talk_decode_actions_str
13 \bool_new:N \l__talk_decode_actions_bool
14 \clist_new:N \l__talk_decode_actions_clist
15 \str_new:N \l__talk_decode_actions_str

(End of definition for \l__talk_decode_actions_bool, \l__talk_decode_actions_clist, and \l__-
talk_decode_actions_str.)

```

`__talk_decode_parse:n` First a simple check for an entirely blank argument: if that's the case, there is no additional overlay to consider. Then deal with any category code issues before looping over blocks divided by `|` tokens.

`__talk_decode_parse_aux:n`

`__talk_decode_parse:w`

```

16 \cs_new_protected:Npn \__talk_decode_parse:n #1
17 {
18   \str_clear:N \l__talk_decode_action_str
19   \bool_lazy_or:nnTF
20     { \tl_if_blank_p:n {#1} }
21     { \str_if_eq_p:nn {#1} { all } }
22     { \bool_set_true:N \l__talk_decode_overlays_bool }
23     {
24       \str_set:Nn \l__talk_decode_arg_str {#1}
25       \bool_set_false:N \l__talk_decode_actions_bool
26       \bool_set_false:N \l__talk_decode_overlays_bool
27       \bool_set_true:N \l__talk_decode_pure_bool
28       \str_clear:N \l__talk_decode_overlays_str
29       \str_clear:N \l__talk_decode_actions_str
30       \exp_args:No \__talk_decode_parse_aux:n { \l__talk_decode_arg_str }
31     }
32 }
33 \cs_new_protected:Npn \__talk_decode_parse_aux:n #1
34 { \__talk_decode_parse:w #1 | \q_recursion_tail | \q_recursion_stop }

```

The end-of-loop test here covers the case where the active mode is not mentioned at all in the specification.

```

35 \cs_new_protected:Npn \__talk_decode_parse:w #1 |
36 {
37   \quark_if_recursion_tail_stop_do:nn {#1}
38   {
39     \bool_lazy_and:nnT
40       { \str_if_empty_p:N \l__talk_decode_overlays_str }
41       { ! \l__talk_decode_pure_bool }
42       { \bool_set_true:N \l__talk_decode_overlays_bool }
43   }
44   \exp_args:Ne \__talk_decode_mode:n
45   { \tl_trim_spaces:n {#1} }
46   \__talk_decode_parse:w
47 }

```

(End of definition for `__talk_decode_parse:n`, `__talk_decode_parse_aux:n`, and `__talk_decode_parse:w`.)

`\c__talk_modes_clist` The possible modes: detokenized as that is applied up-front in decoding.

```

48 \clist_const:Ne \c__talk_modes_clist
49 {
50   \tl_to_str:n { handout } ,
51   \tl_to_str:n { projector }
52 }

```

(End of definition for \c__talk_modes_clist.)

Check if the mode is known and current. If we find an action but have no overlay details, they are filled in with a *.

```

\__talk_decode_mode:n
\__talk_decode_mode:w
\__talk_decode_mode_aux:n
53 \cs_new_protected:Npe \__talk_decode_mode:n #1
54 {
55   \clist_if_in:NnTF \exp_not:N \c__talk_modes_clist {#1}
56   {
57     \exp_not:N \str_if_eq:VnT
58     \exp_not:N \l__talk_mode_str {#1}
59     { \bool_set_true:N \exp_not:N \l__talk_decode_overlays_bool }
60   }
61   {
62     \exp_not:N \__talk_decode_mode:w #1 \tl_to_str:n { : : }
63     \exp_not:N \q_stop
64   }
65 }
66 \use:e
67 {
68   \cs_new_protected:Npe \exp_not:N \__talk_decode_mode:w
69   #1 \token_to_str:N :
70   #2 \token_to_str:N :
71   #3 \exp_not:N \q_stop
72 }
73 {
74   \exp_not:N \tl_if_blank:nTF {#2}
75   {
76     \exp_not:N \__talk_decode_mode:nn
77     { \tl_to_str:n { projector } } {#1}
78   }
79   { \exp_not:N \__talk_decode_mode:nn {#1} {#2} }
80 }
81 \cs_new_protected:Npn \__talk_decode_mode:nn #1#2
82 {
83   \str_if_eq:VnTF \l__talk_mode_str {#1}
84   {
85     \__talk_decode_action:n {#2}
86     \str_if_empty:NT \l__talk_decode_overlays_str
87     { \__talk_decode_overlays:nn { overlays } { * } }
88   }
89   {
90     \tl_if_blank:nF {#2}
91     { \bool_set_false:N \l__talk_decode_pure_bool }
92   }
93 }

```

(End of definition for __talk_decode_mode:n, __talk_decode_mode:w, and __talk_decode_mode_aux:n.)

Here, we have two valid possibilities: no action specification at all, or from the known list. If we don't find one of those outcomes, we can issue an error.

```

94 \cs_new_protected:Npe \__talk_decode_action:n #1
95 {
96   \exp_not:N \__talk_decode_action:w

```

```

97     #1 \tl_to_str:n { @ @ } \exp_not:N \q_stop
98   }
99   \use:e
100   {
101     \cs_new_protected:Npn \exp_not:N \__talk_decode_action:w
102     #1 \tl_to_str:n { @ @ } #2 \tl_to_str:n { @ @ } #3 \exp_not:N \q_stop
103   }
104   {
105     \tl_if_blank:nTF {#2}
106     { \__talk_decode_overlays:nn { overlays } {#1} }
107     {
108       \cs_if_exist:cTF { __talk_action_ #1 :N }
109       {
110         \bool_set_false:N \l__talk_decode_pure_bool
111         \str_set:Nn \l__talk_decode_action_str {#1}
112         \tl_if_blank:nF {#2}
113         { \__talk_decode_overlays:nn { actions } {#2} }
114       }
115       {
116         \msg_error:nnV { talk } { bad-action-spec }
117         \l__talk_decode_arg_str
118       }
119     }
120   }

```

(End of definition for __talk_decode_action:n and __talk_decode_action:w.)

```

\__talk_decode_overlays:nn
\__talk_decode_overlays:nN
  \@_decode_overlay_+:nw
\__talk_decode_overlay_.:nw
  \__talk_decode_overlay_aux:nN
  \__talk_decode_overlay_offset:nN
  \__talk_decode_overlay_offset:nN
121 \cs_new_protected:Npn \__talk_decode_overlays:nn #1#2
122   {
123     \bool_set_false:N \l__talk_decode_step_bool
124     \__talk_decode_overlays:nN {#1} #2 \q_recursion_tail \q_recursion_stop
125     \bool_if:NT \l__talk_decode_step_bool
126     { \int_gincr:N \g__talk_pauses_int }
127     \__talk_decode_check:n {#1}
128   }
129 \cs_new_protected:Npn \__talk_decode_overlays:nN #1#2
130   {
131     \quark_if_recursion_tail_stop:N #2
132     \cs_if_exist_use:cF { __talk_decode_overlay_ #2 :nw }
133     {
134       \str_put_right:cn { l__talk_decode_ #1 _str } {#2}
135       \__talk_decode_overlays:nN
136     }
137     {#1}
138   }
139 \cs_new_protected:cpn { __talk_decode_overlay_+:nw } #1
140   {
141     \bool_set_true:N \l__talk_decode_step_bool
142     \__talk_decode_overlay_aux:nN {#1} 1
143   }
144 \cs_new_protected:cpn { __talk_decode_overlay_.:nw } #1

```

The loop here needs to replace all + and . characters by the current automatic value, allowing for any offsets. This step also needs to track whether to increment the automatic value: true if a + is seen, false otherwise.

```
145 { \__talk_decode_overlay_aux:nNN {#1} 0 }
```

The look-ahead for an offset to a relative specification. If the end-of-loop is reached, the value still needs to be inserted: to share auxiliaries, that is done by using the same function as elsewhere, so the end-of-loop markers are re-inserted. Otherwise, there is a check to see if the next token is a (.

```
146 \cs_new_protected:Npn \__talk_decode_overlay_aux:nNN #1#2#3
147 {
148   \quark_if_recursion_tail_stop_do:Nn #3
149   {
150     \__talk_decode_overlay_offset:nNn {#1} #2 { 0 }
151     \q_recursion_tail \q_recursion_stop
152   }
153   \token_if_eq_meaning:NNTF #3 ( % )
154   { \__talk_decode_overlay_offset:nNn {#1} #2 { } }
155   { \__talk_decode_overlay_offset:nNn {#1} #2 { 0 } #3 }
156 }
```

For the end of an offset, any valid overlay specification must have a closing), so this time the end-of-loop case is an error. Otherwise simply collect up tokens until the closing) is found.

```
157 \cs_new_protected:Npn \__talk_decode_overlay_offset:nNnN #1#2#3#4
158 {
159   \quark_if_recursion_tail_stop_do:Nn #4
160   {
161     \msg_error:nnV { talk } { bad-action-spec }
162     \l__talk_decode_arg_str
163   } % (
164   \token_if_eq_meaning:NNTF #4 )
165   { \__talk_decode_overlay_offset:nNn {#1} #2 {#3} }
166   { \__talk_decode_overlay_offset:nNnN {#1} #2 {#3#4} }
167 }
```

Overlay values can never be negative: this is enforced here.

```
168 \cs_new_protected:Npn \__talk_decode_overlay_offset:nNn #1#2#3
169 {
170   \str_put_right:ce { l__talk_decode_ #1 _str }
171   { \int_max:nn { 0 } { #3 + \g__talk_pauses_int + #2 } }
172   \__talk_decode_overlays:nN {#1}
173 }
```

(End of definition for __talk_decode_overlays:nn and others. This function is documented on page ??.)

```
\__talk_decode_check:n
\__talk_decode_check:nw
  \__talk_decode_check_single:nn
  \__talk_decode_check_range:nnn
```

At this stage we have a fully “written out” overlay specification, and need to work out if the current slide is included. We need to look at each entry in the comma-separated list to sort this out. First we filter out the case of a *, then it’s a question of working out whether each entry is a single number or a range, and if the latter, whether it’s open at either the start or the end.

```
174 \cs_new_protected:Npn \__talk_decode_check:n #1
175 {
176   \clist_set:cv { l__talk_decode_ #1 _clist } { l__talk_decode_ #1 _str }
177   \clist_if_in:cnTF { l__talk_decode_ #1 _clist } { * }
178   { \bool_set_true:c { l__talk_decode_ #1 _bool } }
179   {
```

```

180     \clist_map_inline:cn { l__talk_decode_ #1 _clist }
181     { \__talk_decode_check:nw {#1} 0 ##1 - - \q_stop }
182   }
183 }

```

If #4 is empty, both of the “filler” - tokens were consumed: we have a single value. Otherwise there is a range: the setup above ensures that there will be a starting value in all cases due to the leading 0, but there may not be an end one.

```

184 \cs_new_protected:Npn \__talk_decode_check:nw #1#2 - #3 - #4 \q_stop
185 {
186   \tl_if_empty:nTF {#4}
187   { \__talk_decode_check_single:nn {#1} {#2} }
188   {
189     \tl_if_blank:nTF {#3}
190     { \__talk_decode_check_range:nnn {#1} {#2} { \c_max_int } }
191     { \__talk_decode_check_range:nnn {#1} {#2} {#3} }
192   }
193 }
194 \cs_set_protected:Npn \__talk_decode_check_single:nn #1#2
195 {
196   \int_compare:nNnTF \g__talk_slide_int = {#2}
197   {
198     \bool_set_true:c { l__talk_decode_ #1 _bool }
199     \clist_map_break:
200   }
201   {
202     \int_compare:nNnT {#2} > \g__talk_slide_int
203     { \bool_gset_true:N \g__talk_slide_continue_bool }
204   }
205 }

```

TODO: In the following we might want to add a check whether the range was given with #2 being smaller than #3, to be decided upon.

```

206 \cs_set_protected:Npn \__talk_decode_check_range:nnn #1#2#3
207 {
208   \int_compare:nNnF \g__talk_slide_int > {#3}
209   {
210     \int_compare:nNnTF \g__talk_slide_int < {#2}
211     { \bool_gset_true:N \g__talk_slide_continue_bool }
212     {
213       \bool_set_true:c { l__talk_decode_ #1 _bool }
214       \bool_lazy_and:nnT
215       { \int_compare_p:nNn \g__talk_slide_int < {#3} }
216       { \int_compare_p:nNn {#3} < \c_max_int }
217       { \bool_gset_true:N \g__talk_slide_continue_bool }
218       \clist_map_break:
219     }
220   }
221 }

```

(End of definition for __talk_decode_check:n and others.)

```

222 \msg_new:nnnn { talk } { bad-action-spec }
223 { Bad-overlay-specification~"#1". }
224 {
225   The~overlay~specification~given~doesn't~follow~the~pattern~described~in~

```



```
226     the~ltx-talk~manual:~it~has~been~ignored.  
227 }  
228 </class>
```

Part IV

ltx-talk-frame – The structure of frames

1 ltx-talk-frame implementation

Start the DocStrip guards.

```
1 <{*class}>
    Identify the internal prefix.
2 <{@@=talk}>
```

1.1 Slides in frames

Currently, each slide in a frame will produce a separate page in the output (unless the slide is suppressed entirely). Material is then hidden on some pages by using opacity. An alternative approach would be to use Optional Content Groups to have a similar effect on one page per frame. However, whilst that would be relatively clear for appear/disappear effects, it would be much less straight-forward for partial transparency, *etc.*, plus would depend more heavily on viewer support. At a future stage we may wish to revisit this.

`\g__talk_slide_continue_bool` Tracks whether the frame continues after the current slide.

```
3 \bool_new:N \g__talk_slide_continue_bool
```

(End of definition for `\g__talk_slide_continue_bool`.)

`\l__talk_slide_box`

```
4 \box_new:N \l__talk_slide_box
```

(End of definition for `\l__talk_slide_box`.)

`\g__talk_slide_int`

The slide number inside the current frame: needed to know which overlays are active.

`\c@slide`

We also provide L^AT_EX counter-style access.

`\theslide`

```
5 \int_new:N \g__talk_slide_int
6 \cs_new_eq:NN \c@slide \g__talk_slide_int
7 \cs_new:Npn \theslide { \@arabic \c@slide }
```

(End of definition for `\g__talk_slide_int`, `\c@slide`, and `\theslide`. These variables are documented on page ??.)

Required to know which is the last slide in a frame for tagging.

```
8 \property_new:nnnn { slides } { now } { 1 } { \int_use:N \g__talk_slide_int }
```

`__talk_slide:nn`
`__talk_slide_aux:n`

Each slide is parsed inside simple set up, the only complexity being if we are handling fragile frames. There, all `\obeyedline` in the grabbed tokens need to be turned back into `^M` before rescanning: this ensures that any verbatim grabbing in the frame still works. The strange business with setting the continuation boolean is needed as otherwise we get an infinite loop if there is an overlay specification for the frame itself. Tagging should not of itself force slide continuation, so the global boolean is reset for the tagged slide.

```
9 \cs_new_protected:Npn \__talk_slide:nn #1#2
10 {
```

```

11 \group_begin:
12   \tl_set:N\l__talk_tmp_tl
13   {
14     \property_ref:ee { frame . \int_use:N \g__talk_frame_int }
15     { slides }
16   }
17   \str_if_eq:VnTF \l__talk_frame_tagging_str { n }
18   { \str_set:N\l__talk_frame_tagging_str \l__talk_tmp_tl }
19   {
20     \str_replace_all:NnV \l__talk_frame_tagging_str { ,n }
21     \l__talk_tmp_tl
22     \str_replace_all:NnV \l__talk_frame_tagging_str { ,~n }
23     \l__talk_tmp_tl
24   }
25   \int_gzero:N \g__talk_slide_int
26   \RenewCommandCopy \frame \__talk_latex_frame:n
27   \bool_do_while:Nn \g__talk_slide_continue_bool
28   {
29     \int_gincr:N \g__talk_slide_int
30     \bool_gset_false:N \g__talk_slide_continue_bool
31     \__talk_if_overlay:nT {#1}
32     {
33       \__talk_slide_begin:
34       \__talk_if_overlay:VTF \l__talk_frame_tagging_str
35       {
36         \bool_gset_false:N \g__talk_slide_continue_bool
37         \__talk_frame_tag:n
38       }
39       {
40         \bool_gset_false:N \g__talk_slide_continue_bool
41         \__talk_frame_notag:n
42       }
43       {
44         \bool_if:N\l__talk_frame_verb_bool
45         { \__talk_slide_aux:n }
46         { \use:n }
47         {#2}
48       }
49       \__talk_slide_end:
50     }
51   }
52   \property_record:ee { frame . \int_use:N \g__talk_frame_int }
53   { slides }
54 \group_end:
55 }
56 \cs_new_protected:Npn \__talk_slide_aux:n #1
57 {
58   \group_begin:
59   \cs_set:Npn \obeyedline { ^^J }
60   \use:e
61   {
62     \group_end:
63     \tl_retokenize:n {#1}
64   }

```

65 }

(End of definition for `__talk_slide:nn` and `__talk_slide_aux:n`.)

The very last frame will not be recorded by the above, so we have to add to the hook at the very end of the run.

```
66 \AddToHook { enddocument / afterlastpage }
67 {
68   \property_record:ee { frame . \int_use:N \g__talk_frame_int }
69   { slides }
70 }
```

`\g__talk_frame_struct_int` The tagging structure number for the slide: needed by the content placed outside of the current box (for example the frame title).

```
71 \int_new:N \g__talk_frame_struct_int
```

(End of definition for `\g__talk_frame_struct_int`.)

`__talk_slide_begin:`
`__talk_slide_end:`

```
72 \cs_new_protected:Npn \__talk_slide_begin:
73 {
74   \int_gzero:N \g__talk_pauses_int
75   \tl_gclear:N \g__talk_frame_title_tl
76   \tl_gclear:N \g__talk_frame_subtitle_tl
77   \__talk_cnt_save:
78   \vbox_set:Nw \l__talk_slide_box
79   \tl_gclear:N \g__talk_onslide_tl
80 }
81 \cs_new_protected:Npn \__talk_slide_end:
82 {
83   \tl_use:N \g__talk_onslide_tl
84   \vbox_set_end:
85   \bool_if:NT \g__talk_slide_continue_bool
86     { \__talk_cnt_restore: }
87   \vbox_to_ht:nn { \textheight }
88   {
89     \use:c { __talk_slide_align_ \l__talk_frame_alignment_tl :n }
90     { \vbox_unpack_drop:N \l__talk_slide_box }
91   }
92   \clearpage
93 }
```

(End of definition for `__talk_slide_begin:` and `__talk_slide_end:.`)

`__talk_slide_align_bottom:n` A pretty standard abstraction: we make sure there are always two skips.

```
\__talk_slide_align_center:n 94 \cs_new_protected:Npn \__talk_slide_align_bottom:n #1
\__talk_slide_align_stretch:n 95 {
\__talk_slide_align_top:n     96   \skip_vertical:n { Opt~plus~1fil }
97   #1
98   \skip_vertical:n { Opt }
99 }
100 \cs_new_protected:Npn \__talk_slide_align_center:n #1
101 {
102   \skip_vertical:n { Opt~plus~0.5fil }
103   #1
```

```

104     \skip_vertical:n { Opt~plus~0.5fil }
105   }
106 \cs_new_protected:Npn \__talk_slide_align_stretch:n #1
107 {
108     \skip_vertical:n { Opt }
109     #1
110     \skip_vertical:n { Opt }
111   }
112 \cs_new_protected:Npn \__talk_slide_align_top:n #1
113 {
114     \skip_vertical:n { Opt }
115     #1
116     \skip_vertical:n { Opt~plus~1fil }
117   }

```

(End of definition for __talk_slide_align_bottom:n and others.)

1.2 Counters

\l__talk_cnt_reset_seq As \stepcounter, etc., will increment at each overlay, there is a need to save and reset. The list will be finalized at the end of the preamble, so the data storage is created at that point. The starting point is counters created before the class is loaded (other than those for lists, which reset “naturally”). Other cases are handled by adding to \newcounter.

```

118 \seq_new:N \l__talk_cnt_reset_seq
119 \seq_set_from_clist:Nn \l__talk_cnt_reset_seq
120 {
121     equation      ,
122     footnote      ,
123     mpfootnote    ,
124     parentequation
125 }
126 \seq_map_inline:Nn \l__talk_cnt_reset_seq
127 {
128     \int_new:c { g__talk_saved_ #1 _int }
129     \int_gset_eq:cc { g__talk_saved_ #1 _int } { c@ #1 }
130 }

```

(End of definition for \l__talk_cnt_reset_seq.)

__talk_cnt_save: A simple save-and-restore pair.

__talk_cnt_restore:

```

131 \cs_new_protected:Npn \__talk_cnt_save:
132 {
133     \seq_map_inline:Nn \l__talk_cnt_reset_seq
134     { \int_gset_eq:cc { g__talk_saved_ ##1 _int } { c@ ##1 } }
135   }
136 \cs_new_protected:Npn \__talk_cnt_restore:
137 {
138     \seq_map_inline:Nn \l__talk_cnt_reset_seq
139     { \int_gset_eq:cc { c@ ##1 } { g__talk_saved_ ##1 _int } }
140   }

```

(End of definition for __talk_cnt_save: and __talk_cnt_restore:.)

```

\@definecounter Track all counters for resetting.
\std@definecounter
141 \cs_new_eq:NN \std@definecounter \@definecounter
142 \cs_gset_protected:Npn \@definecounter #1
143 {
144   \std@definecounter {#1}
145   \int_new:c { g__talk_saved_ #1 _int }
146   \seq_gput_right:Nn \l__talk_cnt_reset_seq {#1}
147 }

```

(End of definition for \@definecounter and \std@definecounter. These functions are documented on page ??.)

1.3 Frame options

```

\l__talk_frame_alignment_tl
148 \tl_new:N \l__talk_frame_alignment_tl

(End of definition for \l__talk_frame_alignment_tl.)

```

```

\l__talk_action_spec_str
\l__talk_frame_tagging_str
149 \keys_define:nn { talk / frame }
150 {
151   action-spec .str_set:N
152     = \l__talk_action_spec_str ,
153   tag-slides .str_set:N
154     = \l__talk_frame_tagging_str ,
155   vertical-alignment .choices:nn =
156     { bottom , center , stretch , top }
157     {
158       \tl_set_eq:NN \l__talk_frame_alignment_tl
159       \l_keys_value_tl
160     }
161 }
162 \keys_set:nn { talk / frame }
163 {
164   action-spec = ,
165   tag-slides = n ,
166   vertical-alignment = center
167 }

(End of definition for \l__talk_action_spec_str and \l__talk_frame_tagging_str.)

```

1.4 Tagging for headers

```

\__talk_header_tag_begin:n Generalized control for inserting material into the header area (which is otherwise outside
\__talk_header_tag_begin:e of tagging).
\__talk_header_tag_end:
168 \cs_new_protected:Npn \__talk_header_tag_begin:n #1
169 {
170   \tag_resume:n { header }
171   \tag_mc_end:
172   \tag_struct_begin:n {#1}
173   \tag_mc_begin:n { }
174 }
175 \cs_generate_variant:Nn \__talk_header_tag_begin:n { e }

```

```

176 \cs_new_protected:Npn \__talk_header_tag_end:
177 {
178   \tag_mc_end:
179   \tag_struct_end:
180   \tag_mc_begin:n { artifact }
181   \tag_suspend:n { header }
182 }

```

(End of definition for __talk_header_tag_begin:n and __talk_header_tag_end:.)

1.5 Wallpaper

```

\l__talk_footelem_left_skip
\l__talk_footelem_right_skip
\l__talk_footelem_color_tl
\l__talk_footelem_font_tl
183 \NewTemplateType { footer-element } { 1 }
184 \DeclareTemplateInterface { footer-element } { talk } { 1 }
185 {
186   color      : tokenlist ,
187   font       : tokenlist = ,
188   left-skip  : length = 0em ,
189   right-skip : length = 0em
190 }
191 \DeclareTemplateCode { footer-element } { talk } { 1 }
192 {
193   color      = \l__talk_footelem_color_tl ,
194   font       = \l__talk_footelem_font_tl ,
195   left-skip  = \l__talk_footelem_left_skip ,
196   right-skip = \l__talk_footelem_right_skip
197 }
198 {
199   \tl_if_empty:nF {#1}
200   {
201     \hspace { \l__talk_footelem_left_skip }
202     \group_begin:
203       \tl_if_empty:NF \l__talk_footelem_color_tl
204       { \color_select:V \l__talk_footelem_color_tl }
205       \l__talk_footelem_font_tl
206       #1
207     \group_end:
208     \hspace { \l__talk_footelem_right_skip }
209   }
210 }
211 \DeclareInstance { footer-element } { date } { talk } { }
212 \DeclareInstance { footer-element } { author } { talk } { }
213 \DeclareInstance { footer-element } { title } { talk } { }
214 \DeclareInstance { footer-element } { institute } { talk } { }
215 \DeclareInstance { footer-element } { framenumbers } { talk } { }

```

(End of definition for \l__talk_footelem_left_skip and others.)

```

\l__talk_header_bg_tl
\l__talk_header_fg_tl
\l__talk_header_font_tl
\l__talk_header_ht_dim
\l__talk_header_left_skip
\l__talk_header_frametitle_bool
\l__talk_header_right_skip

```

Templates for the header area. The background always covers the full width, but the text area may be narrower. The setup here aims to avoid repeated kerns but also dealing with complex conditionals, hence we always move to the edge of the paper first then adjust as required.

```

216 \NewTemplateType { header } { 0 }

```

```

217 \DeclareTemplateInterface { header } { talk } { 0 }
218 {
219     background-color : tokenlist,
220     color             : tokenlist = structure ,
221     font              : tokenlist = \normalfont ,
222     height            : length = \Gm@tmargin + \headsep ,
223     left-hspace       : skip = \Gm@lmargin ,
224     print-frame-title : boolean = true ,
225     right-hspace      : skip = \Gm@rmargin
226 }
227 \DeclareTemplateCode { header } { talk } { 0 }
228 {
229     background-color = \l__talk_header_bg_tl ,
230     color            = \l__talk_header_fg_tl ,
231     font             = \l__talk_header_font_tl ,
232     height           = \l__talk_header_ht_dim ,
233     left-hspace      = \l__talk_header_left_skip ,
234     print-frame-title = \l__talk_header_frametitle_bool ,
235     right-hspace     = \l__talk_header_right_skip
236 }
237 {
238     \noindent
239     \__talk_wallpaper_hruler:Nnn
240     \l__talk_header_bg_tl
241     { \l__talk_header_ht_dim - \headsep }
242     { \headsep }
243     \skip_horizontal:n { \l__talk_header_left_skip }
244     \group_begin:
245     \tl_if_empty:NF \l__talk_header_fg_tl
246     { \color_select:V \l__talk_header_fg_tl }
247     \l__talk_header_font_tl
248     \bool_if:NT \l__talk_header_frametitle_bool
249     {
250         \ExpandArgs { nnV }
251         \UseInstance { frametitle } { header }
252         \g__talk_frame_title_tl
253     }
254     \group_end:
255 }
256 \DeclareInstance { header } { std } { talk } { }
257 \AddToHook { begindocument }
258 {
259     \DeclareInstanceCopy { header } { wallpaper } { std }
260     \EditInstance { header } { wallpaper }
261     { print-frame-title = false }
262 }

```

(End of definition for \l__talk_header_bg_tl and others.)

\l__talk_footer_bg_tl Templates for the footer area. Again the margins are handled in stages: here we do have
\l__talk_footer_fg_tl a box for the content so the right skip is used, and we avoid an overfull box by including
\l__talk_footer_font_tl consideration of the right margin of the page layout.
\l__talk_footer_order_clist

```

263 \NewTemplateType { footer } { 0 }
264 \DeclareTemplateInterface { footer } { talk } { 0 }

```

\l__talk_footer_sep_tl
\l__talk_footer_left_skip
\l__talk_footer_right_skip


```

265 {
266     background-color : tokenlist ,
267     color             : tokenlist ,
268     element-order     : commalist ,
269     font              : tokenlist = \tiny ,
270     left-skip         : length = \Gm@lmargin ,
271     right-skip        : length = \Gm@rmargin ,
272     separator         : tokenlist = \hfil
273 }
274 \DeclareTemplateCode { footer } { talk } { 0 }
275 {
276     background-color = \l__talk_footer_bg_tl ,
277     color            = \l__talk_footer_fg_tl ,
278     element-order    = \l__talk_footer_order_clist ,
279     separator        = \l__talk_footer_sep_tl ,
280     font             = \l__talk_footer_font_tl ,
281     left-skip        = \l__talk_footer_left_skip ,
282     right-skip       = \l__talk_footer_right_skip
283 }
284 {
285     \noindent
286     \__talk_wallpaper_hrule:Nnn
287     \l__talk_footer_bg_tl
288     { \footskip }
289     { \Gm@bmargin - \footskip }
290     \skip_horizontal:n { \l__talk_footer_left_skip }
291     \vbox_set_to_wd:Nnn \l__talk_tmp_box
292     {
293         \paperwidth
294         - \l__talk_footer_left_skip
295         - \l__talk_footer_right_skip
296     }
297     {
298         \tl_if_empty:NF \l__talk_footer_fg_tl
299         { \color_select:V \l__talk_footer_fg_tl }
300         \l__talk_footer_font_tl
301         \clist_pop:NNT \l__talk_footer_order_clist \l__talk_tmp_tl
302         {
303             \ExpandArgs { nVv } \UseInstance { footer-element } \l__talk_tmp_tl
304             { @ \l__talk_tmp_tl }
305             \clist_map_inline:Nn \l__talk_footer_order_clist
306             {
307                 \l__talk_footer_sep_tl
308                 \ExpandArgs { nnv }
309                 \UseInstance { footer-element } {##1} { @ ##1 }
310             }
311         }
312         \hfil
313     }
314     \box_use_drop:N \l__talk_tmp_box
315     \skip_horizontal:n { \l__talk_footer_right_skip - \Gm@rmargin }
316 }
317 \DeclareInstance { footer } { std } { talk } { }
318 \AddToHook { begindocument }

```

```

319 {
320   \DeclareInstanceCopy { footer } { wallpaper } { std }
321   \EditInstance { footer } { wallpaper }
322     { element-order = }
323 }

```

(End of definition for \l__talk_footer_bg_tl and others.)

__talk_wallpaper_hrule:Nnn A simple abstraction for the top and bottom rules on the page.

```

324 \cs_new_protected:Npn \__talk_wallpaper_hrule:Nnn #1#2#3
325 {
326   \skip_horizontal:n { -\Gm@lmargin }
327   \tl_if_empty:NF #1
328   {
329     \group_begin:
330       \color_select:V #1
331       \rule:nnn { \paperwidth } {#2} {#3}
332       \skip_horizontal:n { -\paperwidth }
333     \group_end:
334   }
335 }

```

(End of definition for __talk_wallpaper_hrule:Nnn.)

\ps@plain Install a standard header and footer template, and redefine the plain one as this will be used for frames without “wallpaper” which still need core links, *etc.* We also provide a version that only shows the visual elements: this is deliberately using the same settings as the main templates.

```

336 \cs_set_nopar:Npn \ps@plain
337 {
338   \cs_set_nopar:Npn \@oddhead
339   {
340     \__talk_section_tagged:
341     \hfil
342   }
343   \cs_set_nopar:Npn \@oddfoot { }
344   \cs_set_eq:NN \@evenhead \@oddhead
345   \cs_set_eq:NN \@evenfoot \@oddfoot
346 }
347 \cs_set_nopar:Npn \ps@wallpaper
348 {
349   \cs_set_nopar:Npn \@oddhead
350   {
351     \__talk_section_tagged:
352     \UseInstance { header } { wallpaper }
353     \hfil
354   }
355   \cs_set_nopar:Npn \@oddfoot
356   {
357     \UseInstance { footer } { wallpaper }
358     \hfil
359   }
360   \cs_set_eq:NN \@evenhead \@oddhead
361   \cs_set_eq:NN \@evenfoot \@oddfoot

```

```

362 }
363 \cs_new_nopar:Npn \ps@talk
364 {
365   \cs_set_nopar:Npn \@oddhead
366   {
367     \__talk_section_tagged:
368     \UseInstance { header } { std }
369     \hfil
370   }
371   \cs_set_nopar:Npn \@oddfoot { \UseInstance { footer } { std } }
372   \cs_set_eq:NN \@evenhead \@oddhead
373   \cs_set_eq:NN \@evenfoot \@oddfoot
374 }
375 \pagestyle { talk }

```

(End of definition for \ps@plain, \ps@wallpaper, and \ps@talk. These functions are documented on page ??.)

1.6 The frame environment

`\l__talk_frame_bool` To track whether we are inside a frame or not.

```

376 \bool_new:N \l__talk_frame_bool

```

(End of definition for \l__talk_frame_bool.)

`\g__talk_frame_tag_bool` To track when a frame is being tagged: mainly needed for the header (and as a result global).

```

377 \bool_new:N \g__talk_frame_tag_bool

```

(End of definition for \g__talk_frame_tag_bool.)

`\l__talk_frame_verb_bool` Indicates that material was collected verbatim (and thus needs rescanning).

```

378 \bool_new:N \l__talk_frame_verb_bool

```

(End of definition for \l__talk_frame_verb_bool.)

`\g__talk_frame_int` The overall frame number, including L^AT_EX counter-like access.

```

\c@frame 379 \int_new:N \g__talk_frame_int
\theframe 380 \cs_new_eq:NN \c@frame \g__talk_frame_int
\@framenum 381 \cs_new:Npn \theframe { \@arabic \c@frame }
382 \cs_new:Npn \@framenum { \arabic { frame } }

```

(End of definition for \g__talk_frame_int and others. These variables are documented on page ??.)

The total frames can be handled using the kernel properties.

```

383 \property_new:nnnn { totalframes } { shipout } { -1 }
384 { \int_use:N \g__talk_frame_int }
385 \AddToHook { enddocument / afterlastpage }
386 { \property_record:nn { lastpage } { totalframes } }

```

`__talk_latex_frame:n` As we will need to re-define \frame but have it available inside frames, a copy is made here.

```

387 \NewCommandCopy \__talk_latex_frame:n \frame

```

(End of definition for __talk_latex_frame:n.)

`__talk_frame_process:nn` Here, the frame content is received as the argument.

```

388 \cs_new_protected:Npn \__talk_frame_process:nn #1#2
389 {
390   \int_gincr:N \g__talk_frame_int
391   \bool_set_true:N \l__talk_frame_bool
392   \__talk_slide:nn {#1} {#2}
393 }

```

(End of definition for __talk_frame_process:nn.)

`__talk_frame_tag:n` Wraps some content in tagging for a frame: we may have multiple of these in one logical frame, but that is non-standard.

```

394 \cs_new_protected:Npn \__talk_frame_tag:n #1
395 {
396   \tag_struct_begin:n { tag = frame }
397   \int_gset:Nn \g__talk_frame_struct_int { \tag_get:n { struct_num } }
398   \bool_gset_true:N \g__talk_frame_tag_bool
399   #1
400   \tag_struct_end:
401 }

```

(End of definition for __talk_frame_tag:n.)

`__talk_frame_notag:n` The alternative: turn off tagging and suppress the function that would tag the frame title.

```

402 \cs_new_protected:Npn \__talk_frame_notag:n #1
403 {
404   \tag_mc_begin:n { artifact }
405   \tag_suspend:n { frame }
406   \bool_gset_false:N \g__talk_frame_tag_bool
407   #1
408   \par
409   \tag_resume:n { frame }
410   \tag_mc_end:
411 }

```

(End of definition for __talk_frame_notag:n.)

frame The definition for the **frame** and **frame*** environments: the exact interface at both the document and code levels is still open.

```

412 \bool_if:NTF \l__talk_frame_title_bool
413 {
414   \RenewDocumentEnvironment { frame }
415     { D <> { all } = { action-spec } 0 { } +m +b }
416     {
417       \keys_set:nn { talk / frame } {#2}
418       \bool_set_false:N \l__talk_frame_verb_bool
419       \__talk_frame_process:nn {#1} { \frametitle {#3} #4 }
420     }
421   { }
422   \NewDocumentEnvironment { frame* }
423     { D <> { all } = { action-spec } 0 { } +m c }
424     {
425       \keys_set:nn { talk / frame } {#2}

```

```

426     \bool_set_true:N \l__talk_frame_verb_bool
427     \tl_gset:Nn \g__talk_frame_title_tl {#3}
428     \exp_args:Nne \__talk_frame_process:nn {#1}
429     { \tl_to_str:n { \frametitle } \exp_not:n { {#3} #4 } }
430   }
431   { }
432 }
433 {
434   \RenewDocumentEnvironment { frame }
435   { !D <> { all } = { action-spec } !0 { } +b }
436   {
437     \keys_set:nn { talk / frame } {#2}
438     \bool_set_false:N \l__talk_frame_verb_bool
439     \__talk_frame_process:nn {#1} {#3}
440   }
441   { }
442   \NewDocumentEnvironment { frame* }
443   { !D <> { all } = { action-spec } !0 { } c }
444   {
445     \keys_set:nn { talk / frame } {#2}
446     \bool_set_true:N \l__talk_frame_verb_bool
447     \__talk_frame_process:nn {#1} {#3}
448   }
449   { }
450 }

```

(End of definition for frame and frame. These functions are documented on page ??.)*

451 \langle /class \rangle

Part V

ltx-talk-frame – The structure of frames

1 ltx-talk-frame-structure implementation

Start the DocStrip guards.

```
1 <{*class}>
    Identify the internal prefix.
2 <{@@=talk}>
```

1.1 Columns

```
3 \keys_define:nn { talk }
4   { columns .inherit:n = talk / column }
```

`\l__talk_columns_wd_tl` We store the requested width for columns in a `tl` as this means that the key value will make sense even if it depends on the current `\textwidth`.

```
5 \keys_define:nn { talk / columns }
6   { width .tl_set:N = \l__talk_columns_wd_tl }
7 \keys_set:nn { talk / columns }
8   { width = \textwidth }
```

(End of definition for `\l__talk_columns_wd_tl`.)

`columns (env.)` Columns are block-like environments so we start and end with a `\par` to ensure correct tagging.

```
9 \NewDocumentEnvironment { columns } { D <> { all } 0 { } }
10 {
11   \__talk_action_begin:n {#1}
12   \par
13   \keys_set:nn { talk / columns } {#2}
14   \hbox_set_to_wd:Nnw \l__talk_tmp_box { \l__talk_columns_wd_tl }
15   \dim_set:Nn \textwidth { \l__talk_columns_wd_tl }
16   \dim_set_eq:NN \columnwidth \textwidth
17   \hfil
18   \ignorespaces
19 }
20 {
21   \unskip
22   \hfil
23   \hbox_set_end:
24   \box_use_drop:N \l__talk_tmp_box
25   \par
26   \__talk_action_end:
27 }
```

`\l__talk_column_alignment_tl`

```

28 \keys_define:nn { talk / column }
29 {
30   b .meta:n =
31     { vertical-alignment = bottom } ,
32   b .value_forbidden:n = true ,
33   c .meta:n =
34     { vertical-alignment = center } ,
35   c .value_forbidden:n = true ,
36   t .meta:n =
37     { vertical-alignment = top } ,
38   t .value_forbidden:n = true ,
39   vertical-alignment .choices:nn =
40     { bottom , center , top }
41     {
42       \tl_set_eq:NN \l__talk_column_alignment_tl
43       \l_keys_value_tl
44     }
45 }
46 \keys_set:nn { talk / column }
47 {
48   vertical-alignment = center
49 }

```

(End of definition for \l__talk_column_alignment_tl.)

`__talk_column_align_bottom:n`
`__talk_column_align_center:n`
`__talk_column_align_top:n`

Based on ideas in the highly experimental `xbox`.

```

50 \cs_new_protected:Npn \__talk_column_align_bottom:n #1
51   { \vbox:n {#1} }
52 \cs_new_protected:Npn \__talk_column_align_center:n #1
53   {
54     \vbox:n
55     {
56       \hbox:n
57       {
58         \box_move_down:nn
59         {
60           0.5 \box_ht:N \l__talk_tmp_box
61           - \tex_fontdimen:D 22 ~ \tex_textfont:D 2 ~
62         }
63         { \vbox:n {#1} }
64       }
65     }
66   }
67 \cs_new_protected:Npn \__talk_column_align_top:n #1
68   { \vbox_top:n {#1} }

```

(End of definition for __talk_column_align_bottom:n, __talk_column_align_center:n, and __talk_column_align_top:n.)

`column (env.)` A cut-down version of a `minipage`: we want to be clear on the semantic meaning.

```

69 \NewDocumentEnvironment { column } { D <> { all } 0 { } m }
70 {
71   \__talk_action_begin:n {#1}

```

```

72   \par
73   \keys_set:nn { talk / column } {#2}
74   \vbox_set_to_wd:Nnw \l__talk_tmp_box {#3}
75     \dim_set:Nn \textwidth {#3}
76     \dim_set_eq:NN \columnwidth \textwidth
77     \@parboxrestore
78     \leavevmode
79     \raggedright
80     \ignorespaces
81   }

```

The `\@ignore` here means that any spaces after `\end{column}` are suppressed by a `\ignorespaces` inserted by the kernel.

```

82   {
83     \vbox_set_end:
84     \use:c { __talk_column_align_ \l__talk_column_alignment_tl :n }
85       { \vbox_unpack_drop:N \l__talk_tmp_box }
86     \hfil
87     \par
88     \__talk_action_end:
89     \@ignoretrue
90   }

```

1.2 Floats

Well really “not floats at all” but the idea is clear.

`\l__talk_float_alignment_tl` We only worry about horizontal alignment here.

```

91 \tl_new:N \l__talk_float_alignment_tl

```

(End of definition for `\l__talk_float_alignment_tl`.)

A bit similar to the current approach to lists: we need a template at the start but a common function at the end. The `float-placement` key is at present just there to allow mopping up of any argument that is given by accident, hence maps to a temporary variable.

```

92 \NewTemplateType { floatenv } { 2 }
93 \DeclareTemplateInterface { floatenv } { talk } { 2 }
94   {
95     float-placement : tokenlist ,
96     horizontal-alignment : choice { left , center , right } = left
97   }
98 \DeclareTemplateCode { floatenv } { talk } { 2 }
99   {
100     float-placement = \l__talk_tmp_tl ,
101     horizontal-alignment =
102       {
103         left = \tl_set:Nn \l__talk_float_alignment_tl { flushleft } ,
104         center = \tl_set:Nn \l__talk_float_alignment_tl { center } ,
105         right = \tl_set:Nn \l__talk_float_alignment_tl { flushright }
106       }
107   }
108   {
109     \SetTemplateKeys { floatenv } { talk } {#1}
110     \begin { minipage } { \columnwidth }

```



```

111     \begin { \l__talk_float_alignment_tl }
112     \cs_set_nopar:Npn \@capytype {#2}
113   }
114 \DeclareInstance { floatenv } { std } { talk } { horizontal-alignment = left }

\endfloatenv And the common end function.
115 \cs_new_protected:Npn \endfloatenv
116 {
117     \end { \l__talk_float_alignment_tl }
118     \end { minipage }
119 }

```

(End of definition for \endfloatenv. This function is documented on page ??.)

figure (env.) Unlike beamer, we allow for overlays for the environments as a whole.

```

table (env.) 120 \clist_map_inline:nn { figure , table }
121 {
122     \NewDocumentEnvironment {#1} { D <> { all } = { float-placement } 0 { } }
123     {
124         \__talk_action_begin:n {##1}
125         \UseInstance { floatenv } { std } {##2} {#1}
126     }
127     {
128         \endfloatenv
129         \__talk_action_end:
130     }

```

```

\c@figure The standard variables needed to make captions work (nothing for list of floats, as at
\thefigure present those are not offered).
\c@table 131 \newcounter {#1}
\thetable 132 \tl_new:c { #1 name }
\figurename 133 \tl_set:ce { #1 name } { \text_titlecase_first:n {#1} }
\tableename 134 \tl_new:c { fnum@ #1 }
\fnum@figure 135 \tl_set:ce { fnum@ #1 }
\fnum@table 136 { \exp_not:c { #1 name } \exp_not:N \nobreakspace \exp_not:c { the #1 } }
137 }

```

(End of definition for \c@figure and others. These variables are documented on page ??.)

The spacing values needed for the standard function.

```

138 \newlength \abovecaptionskip
139 \newlength \belowcaptionskip
140 \setlength \abovecaptionskip { 7pt }
141 \setlength \belowcaptionskip { 7pt }

```

\@caption This is a copy of the kernel version of the function, but with writing to the list of whatever file removed. It is very likely this needs to be reworked as a template, but that will likely come from the kernel.

```

142 \cs_set_protected:Npn \@caption #1 [ #2 ] #3
143 {
144     \par
145     \begingroup
146         \@parboxrestore
147         \if@minipage \@setminipage \fi
148         \normalsize

```

```

149     \@makecaption { \csname fnum@ #1 \endcsname } { \ignorespaces #3 }
150     \par
151     \endgroup
152 }

(End of definition for \@caption. This function is documented on page ??.)

153 </class>

```

Part VI

ltx-talk-mode — Modes

1 ltx-talk-mode implementation

Start the DocStrip guards.

```
1 <*class>
    Identify the internal prefix.
2 <@@=talk>
```

`__talk_mode:nT` A simplified version of `\mode`: only deal with the argument form, only check the entire overlay spec as a string.

```
3 \prg_new_protected_conditional:Npnn \__talk_mode:n #1 { T }
4 {
5     \bool_lazy_or:nnTF
6     { \str_if_eq_p:nn {#1} { all } }
7     { \str_if_eq_p:Vn \l__talk_mode_str {#1} }
8     \prg_return_true:
9     \prg_return_false:
10 }
```

(End of definition for __talk_mode:nT.)

`\mode`

```
11 \NewDocumentCommand \mode { D <> { all } +m }
12 { \__talk_mode:nT {#1} {#2} }
```

(End of definition for \mode. This function is documented on page ??.)

```
13 </class>
```

Part VII

ltx-talk-overlay — Overlays

1 ltx-talk-overlay implementation

Start the DocStrip guards.

```
1 <*class>
    Identify the internal prefix.
2 <@@=talk>
```

1.1 Utilities

```
__talk_if_overlay:nTF
__talk_if_overlay:VTF
__talk_overlay_arg:n
3 \prg_new_protected_conditional:Npnn __talk_if_overlay:n #1 { T , F , TF }
4 {
5     __talk_decode_parse:n {#1}
6     \bool_if:NTF \l__talk_decode_overlays_bool
7     \prg_return_true:
8     \prg_return_false:
9 }
10 \prg_generate_conditional_variant:Nnn __talk_if_overlay:n { V } { T , F , TF }
```

A macro processor variant of the check that always results in an N-type bool.

```
11 \cs_new_protected:Npn __talk_overlay_arg:n #1
12 {
13     __talk_if_overlay:nTF {#1}
14     { \cs_set:Npn \ProcessedArgument { \c_true_bool } }
15     { \cs_set:Npn \ProcessedArgument { \c_false_bool } }
16 }
```

(End of definition for __talk_if_overlay:nTF and __talk_overlay_arg:n.)

1.2 Action commands and environments

Commands that can be used as actions all have a common form (with one exception). The common internal structure is used to enable them to be used as actions by looking for the name `__talk_action_⟨name⟩:N`. This is set up such that the inactive versions insert a whatsit equal to that which would be present if they were active: that's needed for spacing.

```
__talk_action_:N The fallback action.
17 \cs_new_protected:Npn __talk_action_:N #1 { }
```

(End of definition for __talk_action_:N.)

```
__talk_action_alert:N At present a color selection.
18 \cs_new_protected:Npn __talk_action_alert:N #1
19 {
20     \bool_if:NTF #1
21     { \color_select:n { alert } }
22     { \color_select:n { . } }
23 }
```

(End of definition for `__talk_action_alert:N`.)

`__talk_action_invisible:N`
`__talk_action_visible:N`

Simply hide unconditionally.

```

24 \cs_new_protected:Npn \__talk_action_invisible:N #1
25 {
26   \bool_if:NTF #1
27     { \opacity_select:n { 0 } }
28     { \opacity_select:n { 1 } }
29 }
30 \cs_new_protected:Npn \__talk_action_visible:N #1
31 {
32   \bool_if:NTF #1
33     { \opacity_select:n { 1 } }
34     { \opacity_select:n { 0 } }
35 }

```

(End of definition for `__talk_action_invisible:N` and `__talk_action_visible:N`.)

`__talk_action_only_begin:N`
`__talk_action_only_end:N`

Here, we simply throw away the content we do not want: this is done by typesetting in a disposable box.

```

36 \cs_new_protected:Npn \__talk_action_only:N #1
37 {
38   \bool_if:NF #1
39     { \vbox_set:Nw \l__talk_tmp_box }
40 }
41 \cs_new_protected:Npn \__talk_action_only_end:N #1
42 {
43   \bool_if:NF #1
44     { \vbox_set_end: }
45 }

```

(End of definition for `__talk_action_only_begin:N` and `__talk_action_only_end:N`.)

`\l__talk_uncover_hidden_fp`

Currently just an on-off, but that will change.

```

46 \NewTemplateType { hidden } { 0 }
47 \DeclareTemplateInterface { hidden } { talk } { 0 }
48   { opacity : real = 0 }
49 \DeclareTemplateCode { hidden } { talk } { 0 }
50   { opacity = \l__talk_uncover_hidden_fp }
51   { \opacity_select:n { \l__talk_uncover_hidden_fp } }
52 \DeclareInstance { hidden } { std } { talk } { }

```

(End of definition for `\l__talk_uncover_hidden_fp`.)

`__talk_action_uncover:N`

Use the template

```

53 \cs_new_protected:Npn \__talk_action_uncover:N #1
54 {
55   \bool_if:NTF #1
56     { \opacity_select:n { 1 } }
57     { \UseInstance { hidden } { std } }
58 }

```

(End of definition for `__talk_action_uncover:N`.)

`\only` Commands and environments where the payload applies when the material is not active
`\invisible` on the slide.

```
\uncover
59 \clist_map_inline:nn { only , invisible , uncover }
60 {
61   \ExpandArgs { cne } \NewDocumentCommand {#1}
62     { > { \__talk_overlay_arg:n } D <> { all } +m }
63     {
64       \group_begin:
65       \exp_not:c { __talk_action_ #1 :N } ##1
66       ##2
67       \cs_if_exist:cT { __talk_action_ #1 _end:N }
68       { \exp_not:c { __talk_action_ #1 _end:N } ##1 }
69       \group_end:
70     }
```

(End of definition for \only, \invisible, and \uncover. These functions are documented on page ??.)

```
onlyenv (env.)
invisibleenv (env.)
71 \ExpandArgs { nnee } \NewDocumentEnvironment { #1 env }
uncoverenv (env.)
72 { > { \__talk_overlay_arg:n } D <> { all } }
73 { \exp_not:c { __talk_action_ #1 :N } ##1 }
74 {
75   \cs_if_exist:cT { __talk_action_ #1 _end:N }
76   { \exp_not:c { __talk_action_ #1 _end:N } ##1 }
77 }
78 }
```

`\alert` And those where the action applies when we are on the slide.

```
\visible
79 \clist_map_inline:nn { alert , visible }
80 {
81   \ExpandArgs { cne } \NewDocumentCommand {#1}
82     { > { \__talk_overlay_arg:n } D <> { all } +m }
83     {
84       \group_begin:
85       \exp_not:c { __talk_action_ #1 :N } ##1
86       ##2
87       \group_end:
88     }
```

(End of definition for \alert and \visible. These functions are documented on page ??.)

```
alertenv (env.)
visibleenv (env.)
89 \ExpandArgs { nnee } \NewDocumentEnvironment { #1 env }
90 { > { \__talk_overlay_arg:n } D <> { all } }
91 { \exp_not:c { __talk_action_ #1 :N } ##1 }
92 { }
93 }
```

`\only` This code needs to be done manually as for the command version the content must be entirely discarded. That can't work for the environment version, which has to deal with for example single items in a list (and so cannot be collected up verbatim and must use a box).

```
94 \RenewDocumentCommand \only { D <> { all } +m }
95 {
```

```

96     \_talk_if_overlay:nT {#1}
97     {#2}
98 }

```

(End of definition for \only. This function is documented on page ??.)

```

\l__talk_saved_overlays_bool
\l__talk_saved_action_str
\l__talk_saved_actions_bool
99 \bool_new:N \l__talk_saved_overlays_bool
100 \str_new:N \l__talk_saved_action_str
101 \bool_new:N \l__talk_saved_actions_bool

```

(End of definition for \l__talk_saved_overlays_bool, \l__talk_saved_action_str, and \l__talk_saved_actions_bool.)

actionenv

As we need data on not just overlays but also actions at the end of the environment, this has to be done manually. To allow working with environments but also items, the code needs to save data for the end function. The group is needed for cases where we are not in a L^AT_EX environment group.

```

\__talk_action_begin:n
\__talk_action_end:
102 \NewDocumentCommand \action { D <> { all } +m }
103 {
104     \group_begin:
105     \__talk_action_begin:n {#1}
106     #2
107     \__talk_action_end:
108     \group_end:
109 }
110 \NewDocumentEnvironment { actionenv } { D <> { all } }
111 { \__talk_action_begin:n {#1} }
112 { \__talk_action_end: }
113 \cs_new_protected:Npn \__talk_action_begin:n #1
114 {
115     \group_begin:
116     \__talk_decode_parse:n {#1}
117     \bool_set_eq:NN \l__talk_saved_overlays_bool
118     \l__talk_decode_overlays_bool
119     \str_set_eq:NN \l__talk_saved_action_str
120     \l__talk_decode_action_str
121     \bool_set_eq:NN \l__talk_saved_actions_bool
122     \l__talk_decode_actions_bool
123     \bool_if:NTF \l__talk_decode_overlays_bool
124     {
125         \use:c { __talk_action_ \l__talk_decode_action_str :N }
126         \l__talk_decode_actions_bool
127     }
128     { \UseInstance { hidden } { std } }
129 }
130 \cs_new_protected:Npn \__talk_action_end:
131 {
132     \bool_if:NT \l__talk_saved_overlays_bool
133     {
134         \cs_if_exist_use:cF
135         { __talk_action_ \l__talk_saved_action_str _end:N }
136         { \use_none:n }
137         \l__talk_saved_actions_bool

```

```

138     }
139     \group_end:
140 }

```

(End of definition for \action, _talk_action_begin:n, and _talk_action_end:. This function is documented on page ??.)

1.3 Non-action commands and environments

This section contains commands and environments that do *not* need to be made available as actions.

\alt Simple wrappers around the internal switch.

```

141 \NewDocumentCommand \alt { D <> { all } +m +m }
142 {
143   \_talk_if_overlay:nTF {#1}
144   {#2}
145   {#3}
146 }

```

(End of definition for \alt. This function is documented on page ??.)

\onslide Simply make transparent: we will likely need to save the original opacity level. To allow us to apply independent of group level, a little work is needed.

```

\__talk_onslide:n
\__talk_onslide_reset:
147 \NewDocumentCommand \onslide { D <> { all } }
148 { \__talk_onslide:n {#1} }
149 \cs_new_protected:Npn \__talk_onslide:n #1
150 {
151   \tl_use:N \g__talk_onslide_tl
152   \_talk_if_overlay:nTF {#1}
153   { \__talk_onslide_reset: }
154   {
155     \opacity_select:n { 0 }
156     \tl_gset:Nn \g__talk_onslide_escape_tl
157     {
158       \opacity_select:n { 0 }
159       \group_insert_after:N \g__talk_onslide_escape_tl
160     }
161     \group_insert_after:N \g__talk_onslide_escape_tl
162     \tl_gset:Nn \g__talk_onslide_tl
163     {
164       \tl_gclear:N \g__talk_onslide_tl
165       \tl_gclear:N \g__talk_onslide_escape_tl
166       \__talk_onslide_reset:
167     }
168   }
169 }
170 \cs_new_protected:Npn \__talk_onslide_reset: { \opacity_select:n { 1 } }

```

(End of definition for \onslide, _talk_onslide:n, and _talk_onslide_reset:. This function is documented on page ??.)

```

\g__talk_onslide_tl
\g__talk_onslide_escape_tl
171 \tl_new:N \g__talk_onslide_tl
172 \tl_new:N \g__talk_onslide_escape_tl

```


(End of definition for `\g__talk_onslide_tl` and `\g__talk_onslide_escape_tl`.)

`\temporal` A tricky one: to separate the not-on-current-slide cases, the flag to continue is used.

```

173 \NewDocumentCommand \temporal { D <> { all } +m +m +m }
174 {
175   \__talk_if_overlay:nTF {#1}
176     {#3}
177     {
178       \bool_if:NTF \g__talk_slide_continue_bool
179         {#4}
180         {#2}
181     }
182 }

```

(End of definition for `\temporal`. This function is documented on page ??.)

`\pause` A thin wrapper.

```

183 \NewDocumentCommand \pause { o }
184 {
185   \IfNoValueTF {#1}
186     { \int_gincr:N \g__talk_pauses_int }
187     { \int_gset:Nn \g__talk_pauses_int {#1} }
188   \exp_args:Ne \__talk_onslide:n { \int_eval:n { \g__talk_pauses_int + 1 } - }
189 }

```

(End of definition for `\pause`. This function is documented on page ??.)

1.4 Fixed-size areas

`__talk_overprint_begin:n` A common auxiliary for overprinting, which starts off much the same for both `overlayarea` and `overprint`.

```

190 \cs_new_protected:Npn \__talk_overprint_begin:n #1
191 {
192   \par
193   \vbox_set_to_wd:Nnw \l__talk_tmp_box {#1}
194   \raggedright
195   \ignorespaces
196 }

```

(End of definition for `__talk_overprint_begin:n`.)

`overlayarea (env.)` An initial approach: quite similar to a column.

```

197 \NewDocumentEnvironment { overlayarea } { m m }
198 { \__talk_overprint_begin:n {#1} }
199 {
200   \vbox_set_end:
201   \vbox_to_ht:nn {#2}
202   {
203     \box_use_drop:N \l__talk_tmp_box
204     \vfil
205   }
206   \par
207 }

```

`\l__talk_overprint_int` Track the overprints on a slide: as the slide forms a group, we do not need to worry about resetting.

```
208 \int_new:N \l__talk_overprint_int
```

(End of definition for `\l__talk_overprint_int`.)

`__talk_frame_overprint:` To refer to the current overprint environment within the document: needed in the `.aux` so avoids using non-letters.

```
209 \cs_new:Npn \__talk_frame_overprint:
```

```
210 {
```

```
211   \int_to_Roman:n \g__talk_frame_int
```

```
212   \int_to_roman:n \l__talk_overprint_int
```

```
213 }
```

(End of definition for `__talk_frame_overprint:`.)

`__talk_overprint_int` For overprinting, in contrast to `beamer` we use a two-pass approach to save the size at the end of the run: this means you can use `\only` for example in overprinting.

```
214 \NewDocumentEnvironment { overprint } { 0 { \textwidth } }
```

```
215 { \__talk_overprint_begin:n {#1} }
```

```
216 {
```

```
217   \vbox_set_end:
```

```
218   \int_incr:N \l__talk_overprint_int
```

```
219   \__talk_overprint_save_ht:
```

```
220   \cs_if_exist:cTF
```

```
221     { overprint@ \__talk_frame_overprint: } 
```

```
222     {
```

```
223       \dim_compare:vNnTF
```

```
224         { overprint@ \__talk_frame_overprint: } 
```

```
225         > { \box_ht:N \l__talk_tmp_box } 
```

```
226         {
```

```
227           \vbox_to_ht:vn
```

```
228             { overprint@ \__talk_frame_overprint: } 
```

```
229             {
```

```
230               \box_use_drop:N \l__talk_tmp_box
```

```
231               \vfil
```

```
232             }
```

```
233           }
```

```
234           { \box_use_drop:N \l__talk_tmp_box } 
```

```
235         }
```

```
236         { \box_use_drop:N \l__talk_tmp_box } 
```

```
237     \par
```

```
238 }
```

As there is no clear end-point for overprinting, we need to be careful to keep the current width separate from the saved one. The rest is then about saving to the `.aux` file and helping out the user.

```
239 \cs_new_protected:Npn \__talk_overprint_save_ht:
```

```
240 {
```

```
241   \tl_if_exist:cF { g__talk_overprint_ \__talk_frame_overprint: _tl } 
```

```
242   {
```

```
243     \tl_new:c { g__talk_overprint_ \__talk_frame_overprint: _tl } 
```

```
244     \tl_gset:cn { g__talk_overprint_ \__talk_frame_overprint: _tl } 
```

```
245     { Opt } 
```

```

246     }
247     \tl_gset:ce { g__talk_overprint_ \__talk_frame_overprint: _tl }
248     {
249         \dim_max:vn { g__talk_overprint_ \__talk_frame_overprint: _tl }
250         { \box_ht:N \l__talk_tmp_box }
251     }
252     \legacy_if:nT { @files }
253     {
254         \iow_now:Ne \@auxout
255         {
256             \gdef \exp_not:c { overprint@ \__talk_frame_overprint: }
257             {
258                 \exp_not:v { g__talk_overprint_ \__talk_frame_overprint: _tl }
259             }
260         }
261     }
262     \hook_gput_code:nne { enddocument / afterlastpage } { talk }
263     { \__talk_overprint_check_ht:n { \__talk_frame_overprint: } }
264 }
265 \cs_new_protected:Npn \__talk_overprint_check_ht:n #1
266 {
267     \bool_lazy_and:nnF
268     { \exp_not:N \cs_if_exist_p:c { overprint@ #1 } }
269     {
270         \dim_compare_p:vNv { overprint@ #1 } = { g__talk_overprint_ #1 _tl }
271     }
272     {
273         \msg_warning:nn { talk } { overprint-ht }
274         \cs_gset_protected:Npn \__talk_overprint_check_ht:n ##1 { }
275     }
276 }
277 \msg_new:nnn { talk } { overprint-ht }
278 {
279     Overprint~area~height~has~changed:\\
280     rerun~LaTeX.
281 }

```

(End of definition for __talk_overprint_save_ht: and __talk_overprint_check_ht:n.)

1.5 Adding overlays to existing commands

Make the standard text commands overlay-aware. To keep the spacing unchanged when the command is not active, we use the same approach as the kernel does for inserting the right grouping.

```

\textnormal 282 \tl_map_inline:nn
\textrm      283 {
\textsc      284     \textbf
\textsf      285     \textit
\textsl      286     \textmd
\texttt      287     \textnormal
\textup      288     \textrm
\emph        289     \textsc
\stdtextbf   290     \textsf
\stdtextit   291     \textsl
\stdtextmd
\stdtextnormal
\stdtextrm
\stdtextsc
\stdtextsf
\stdtextsl
\stdtexttt
\stdtextup
\stdemph
\__talk_textcmd equiv:n

```

```

292 \texttt
293 \textup
294 \emph
295 }
296 {
297 \ExpandArgs { c } \NewCommandCopy { std \cs_to_str:N #1 } #1
298 \ExpandArgs { Nne } \RenewDocumentCommand #1
299 { D <> { all } +m }
300 {
301 \exp_not:N \__talk_if_overlay:nTF {##1}
302 { \exp_not:c { std \cs_to_str:N #1 } }
303 { \exp_not:N \__talk_textcmd_equiv:n }
304 {##2}
305 }
306 }
307 \cs_new_protected:Npn \__talk_textcmd_equiv:n #1
308 {
309 \mode_if_math:TF
310 { { \mbox {#1} } }
311 {
312 \mode_leave_vertical:
313 {#1}
314 }
315 }

```

(End of definition for `\textbf` and others. These functions are documented on page ??.)

`\includegraphics` Just wrap up the args and forward if appropriate. The star is #1 here as that matches the documented behavior of starred commands generally.

`\stdincludegraphics`

```

316 \RequirePackage { graphicx }
317 \NewCommandCopy \stdincludegraphics \includegraphics
318 \RenewDocumentCommand \includegraphics { s D <> { all } o o m }
319 {
320 \__talk_if_overlay:nT {#2}
321 {
322 \use:e
323 {
324 \exp_not:N \stdincludegraphics
325 \IfBooleanT #1 { * }
326 \IfNoValueF {#3} { [ \exp_not:n { {#3} } ] }
327 \IfNoValueF {#4} { [ \exp_not:n { {#4} } ] }
328 }
329 {#5}
330 }
331 }

```

(End of definition for `\includegraphics` and `\stdincludegraphics`. These functions are documented on page ??.)

`\label` Here, we can't wrap the existing command up as we need the space hack, so it has to be declared from scratch. There is also a non-standard overlay default. At present, no special tricks as seen in `beamer`.

`__talk_label:n`

```

332 \RenewDocumentCommand \label { D <> { 1 } m }
333 {

```

```

334     \@bsphack
335     \_talk_if_overlay:nT {#1}
336     { \_talk_label:n {#2} }
337     \@esphack
338   }
339   \cs_new_protected:Npn \_talk_label:n #1
340   {
341     \begingroup
342       \UseHookWithArguments { label } { 1 } {#1}
343       \protected@write \@auxout { }
344       {
345         \string \newlabel {#1}
346         {
347           { \@currentlabel }
348           { \thepage }
349           { \@currentlabelname }
350           { \@currentHref }
351           { \@kernel@reserved@label@data }
352         }
353       }
354     \endgroup
355   }

```

(End of definition for \label and _talk_label:n. This function is documented on page ??.)

```

356 </class>

```

Part VIII

ltx-talk-required – “Required” definitions

1 ltx-talk-required implementation

Start the DocStrip guards.

```
1 <*class>
    Identify the internal prefix.
2 <@@=talk>
```

Here we collect up things that are more-or-less required to create a useful class but are not defined by the L^AT_EX kernel for historical reasons. They are therefore largely copies from `article.cls` and contain “classical” definitions so that they follow the expectations of third-party code.

`\today` This is the definition as done in the standard classes.

```
3 \cs_new_nopar:Npn \today
4 {
5     \ifcase \month \or
6         January \or
7         February \or
8         March \or
9         April \or
10        May \or
11        June \or
12        July \or
13        August \or
14        September \or
15        October \or
16        November \or
17        December
18    \fi
19    \space
20    \number \day ,
21    \space
22    \number \year
23 }
```

(End of definition for \today. This function is documented on page ??.)

1.1 Standard design settings

```
24 \setcounter { tocdepth } { 3 }
25 \setlength \arraycolsep { 5pt }
26 \setlength \tabcolsep { 6pt }
27 \setlength \arrayrulewidth { 0.4pt }
28 \setlength \doublerulesep { 2pt }
29 \setlength \tabbingsep { \labelsep }
30 \skip \@mpfootins = \skip \footins
```

```

31 \setlength \fboxsep { 3pt }
32 \setlength \fboxrule { 0.4pt }

```

1.2 List support

```

33 \setlength \labelsep { 0.5em }
34 \cs_new:Npn \labelenumi { \theenumi . }
35 \cs_new:Npn \labelenumii { ( \theenumii ) }
36 \cs_new:Npn \labelenumiii { \theenumiii . }
37 \cs_new:Npn \labelenumiv { \theenumiv . }
38 \cs_new:Npn \labelitemi { \labelitemfont \textbullet }
39 \cs_new:Npn \labelitemii { \labelitemfont \bfseries \textendash }
40 \cs_new:Npn \labelitemiii { \labelitemfont \textasteriskcentered }
41 \cs_new:Npn \labelitemiv { \labelitemfont \textperiodcentered }
42 \cs_new:Npn \labelitemfont { \normalfont }

43 \setlength \leftmargini { 2em }
44 \setlength \leftmarginii { 2em }
45 \setlength \leftmarginiii { 2em }
46 \setlength \labelsep { 0.5em }
47 \setlength \labelwidth { \leftmargini }
48 \addtolength \labelwidth { -\labelsep }
49 \cs_gset_nopar:Npn \@listi
50 {
51   \leftmargin \leftmargini
52   \topsep 3pt plus 2pt minus 2.5pt
53   \parsep 0pt
54   \itemsep 3pt plus 2pt minus 3pt
55 }
56 \cs_gset_eq:NN \@listI \@listi
57 \cs_gset_nopar:Npn \@listii
58 {
59   \leftmargin \leftmarginii
60   \topsep 2pt plus 1pt minus 2pt
61   \parsep 0pt plus 1pt
62   \itemsep \parsep
63 }
64 \cs_gset_nopar:Npn \@listiii
65 {
66   \leftmargin \leftmarginiii
67   \topsep 2pt plus 1pt minus 2pt
68   \parsep 0pt plus 1pt
69   \itemsep \parsep
70 }
71 \setlength \partopsep { 0pt }
72 \endclass

```

Part IX

ltx-talk-structure – Structural commands

1 ltx-talk-structure implementation

Start the DocStrip guards.

```
1 <*class>
    Identify the internal prefix.
2 <@@=talk>
```

1.1 Frame title

```
\g__talk_frame_title_tl
\g__talk_frame_subtitle_tl
3 \tl_new:N \g__talk_frame_title_tl
4 \tl_new:N \g__talk_frame_subtitle_tl

(End of definition for \g__talk_frame_title_tl and \g__talk_frame_subtitle_tl.)
```

```
\frametitle Just data storage: at the present no use of the optional argument.
5 \NewDocumentCommand \frametitle { D <> { all } 0 {#3} m }
6 {
7   \__talk_if_overlay:nT {#1}
8   { \tl_gset:Nn \g__talk_frame_title_tl {#3} }
9 }
10 \NewDocumentCommand \framesubtitle { D <> { all } 0 {#3} m }
11 {
12   \__talk_if_overlay:nT {#1}
13   { \tl_gset:Nn \g__talk_frame_subtitle_tl {#3} }
14 }
```

(End of definition for \frametitle. This function is documented on page ??.)

```
\__talk_frame_title:n
\__talk_frame_title_tagged:n
Inserting the frame title requires we deal with tagging as well as appearance: if there is
a title, we need to tag just this part of the header.
15 \NewTemplateType { frametitle } { 1 }
16 \DeclareTemplateInterface { frametitle } { talk } { 1 }
17 {
18   after-vspace : skip = \bigskipamount ,
19   before-vspace : skip = 0em ,
20   color        : tokenlist = ,
21   font         : tokenlist = \Large \bfseries
22 }
23 \DeclareTemplateCode { frametitle } { talk } { 1 }
24 {
25   after-vspace = \l__talk_frametitle_after_skip ,
26   before-vspace = \l__talk_frametitle_before_skip ,
27   color        = \l__talk_frametitle_color_tl ,
28   font         = \l__talk_frametitle_font_tl
29 }
```



```

30 {
31   \noindent
32   \vspace { \l__talk_frametitle_before_skip }
33   \group_begin:
34     \tl_if_empty:NF \l__talk_frametitle_color_tl
35     { \color_select:V \l__talk_frametitle_color_tl }
36     \l__talk_frametitle_font_tl
37     \tl_if_blank:NF {#1}
38     { \__talk_frame_title:n {#1} }
39     \par
40   \group_end:
41   \vspace { \l__talk_frametitle_after_skip }
42 }
43 \DeclareInstance { frametitle } { header } { talk } { }
44 \cs_new_protected:Npn \__talk_frame_title:n #1
45 {
46   \bool_if:NTF \g__talk_frame_tag_bool
47   { \__talk_frame_title_tagged:n }
48   { \use:n }
49   {#1}
50 }
51 \cs_new_protected:Npn \__talk_frame_title_tagged:n #1
52 {
53   \__talk_header_tag_begin:e
54   {
55     firstkid = true ,
56     parent   = \int_use:N \g__talk_frame_struct_int ,
57     tag      = frametitle ,
58     title    = { \text_purify:n { \g__talk_frame_title_tl } } ,
59   }
60   \group_begin:
61     \tagpdfparaOff
62     #1
63   \group_end:
64   \__talk_header_tag_end:
65 }

```

(End of definition for `__talk_frame_title:n` and `__talk_frame_title_tagged:n`.)

1.2 Sectioning

```

\l__talk_section_tl  Two versions of the data store: one set locally (but at the top level) for general use, one
\g__talk_section_tl  set (and more importantly cleared) globally to allow insertion in the header area just
\l__talk_subsection_tl once per name.
\g__talk_subsection_tl
\l__talk_subsubsection_tl 66 \tl_new:N \l__talk_section_tl
\g__talk_subsubsection_tl 67 \tl_new:N \g__talk_section_tl
\l__talk_subsubsection_tl 68 \tl_new:N \l__talk_subsection_tl
\g__talk_subsubsection_tl 69 \tl_new:N \g__talk_subsubsection_tl
\l__talk_subsubsection_tl 70 \tl_new:N \l__talk_subsubsection_tl
\g__talk_subsubsection_tl 71 \tl_new:N \g__talk_subsubsection_tl

```

(End of definition for `\l__talk_section_tl` and others.)

<pre> \section \subsection \subsubsection \thesection \thesubsection \thesubsubsection </pre>	<p>Here, we need full L^AT_EX counters, so create them using the appropriate mechanism: that also means we can sort out counter dependency and the appearance (using the same setup</p>
---	---

as in article). As (subsub)section numbers never increment inside frames, we remove these counters from the general tracker.

```

72 \newcounter { section }
73 \newcounter { subsection } [ section ]
74 \newcounter { subsubsection } [ subsection ]
75 \seq_gremove_all:Nn \l__talk_cnt_reset_seq { section }
76 \seq_gremove_all:Nn \l__talk_cnt_reset_seq { subsection }
77 \seq_gremove_all:Nn \l__talk_cnt_reset_seq { subsubsection }
78 \cs_gset:Npn \thesection { \@arabic \c@section }
79 \cs_gset:Npn \thesubsection { \thesection . \@arabic \c@subsection }
80 \cs_gset:Npn \thesubsubsection { \thesubsection . \@arabic \c@subsubsection }

```

(End of definition for \section and others. These functions are documented on page ??.)

\section \subsection \subsubsection \insertsection \insertsubsection \insertsubsubsection	The sectioning commands all have essentially the same form: we therefore create using a generator with the necessary conditionals in place. As we do not typeset sections at this stage, the code is quite different from article. This also means that the bookmark links need to point forward to the next slide: if that doesn't appear, the bookmarks will be out. Using the general scratch sequence here should be OK: t really is a one-off setting. We need a sequence to allow indexed mapping to avoid any extra setup for the depth value.
--	---

```

81 \seq_set_from_clist:Nn \l_tmpa_seq
82   { section , subsection , subsubsection }
83 \seq_map_indexed_inline:Nn \l_tmpa_seq
84   {
85     \use:e
86     {
87       \NewDocumentCommand \exp_not:c { insert #2 } { { }
88         {
89           \exp_not:N \tl_use:N
90           \exp_not:c { l__talk_ #2 _tl }
91         }
92       \NewDocumentCommand \exp_not:c {#2}
93       { s D <> { all } 0 {##4} m }
94       {
95         \exp_not:N \refstepcounter {#2}
96         \tag_tool:n { sec-start = #2 , restore-para }
97         \tl_set:Nn \exp_not:c { l__talk_ #2 _tl } {##4}
98         \tl_gset_eq:NN \exp_not:c { g__talk_ #2 _tl }
99         \exp_not:c { l__talk_ #2 _tl }
100        \str_if_eq:nnT {#2} { section }
101          { \tl_clear:N \exp_not:N \l__talk_subsubsection_tl }
102        \str_if_eq:nnF {#2} { subsubsection }
103          { \tl_clear:N \exp_not:N \l__talk_subsubsection_tl }
104        \exp_not:N \addcontentsline { toc } {#2}
105        {
106          \exp_not:N \int_compare:nNnF {#1} >
107            { \exp_not:N \value { secnumdepth } }
108            {
109              \exp_not:N \protect \exp_not:N \numberline
110              { \exp_not:c { the #2 } }
111            }
112          ##4

```

```

113         }
114         \hook_use:n { #2 / begin }
115     }
116     \hook_new:n { #2 / begin }
117 }
118 }

```

(End of definition for `\section` and others. These functions are documented on page ??.)

`__talk_section_tagged:`

```

119 \cs_new_protected:Npn \__talk_section_tagged:
120 {
121     \clist_map_inline:nn { section , subsection , subsubsection }
122     {
123         \tl_if_empty:cF { g__talk_ ##1 _ tl }
124         {
125             \__talk_header_tag_begin:e
126             {
127                 tag = ##1 ,
128                 title = { \text_purify:v { g__talk_ ##1 _ tl } } ,
129             }
130             \__talk_header_tag_end:
131             \tl_gclear:c { g__talk_ ##1 _ tl }
132         }
133     }
134 }

```

(End of definition for `__talk_section_tagged:`)

1.3 Table of contents

`\@starttoc` The standard kernel implementation here deliberately overwrites the file as soon as it's read. That's no good for us as the table of contents can be read multiple times. So we modify the code: we start from the tagging-aware version (this may need to be revisited). We retain the $\text{\LaTeX} 2_{\epsilon}$ code as much as possible.

```

135 \cs_gset_protected:Npn \@starttoc #1
136 {
137     \begingroup
138     \makeatletter
139     \UseTaggingSocket { toc / starttoc / before } {#1}
140     \@input { \jobname .#1 }
141     \UseTaggingSocket { toc / starttoc / after } {#1}
142     \legacy_if:nT { @filesw }
143     {
144         \AddToHook { enddocument / afterlastpage }
145         {
146             \expandafter \newwrite \csname tf@ #1 \endcsname
147             \immediate \openout \csname tf@ #1 \endcsname \jobname .#1 \relax
148         }
149     }
150     \nobreakfalse
151     \endgroup
152 }

```

(End of definition for \@starttoc. This function is documented on page ??.)

`\tableofcontents` For the present simply print the output.

```

153 \NewDocumentCommand \tableofcontents { 0 { } }
154 {
155   \group_begin:
156   \@starttoc { toc }
157   \group_end:
158 }

```

(End of definition for \tableofcontents. This function is documented on page ??.)

`\l@section` Initial hard-coded versions to be templated once we have some other effects also working.

`\l@subsection` We may need to look at this “higher up” as we will need to know the section numbers.

`\l@subsubsection`

```

159 \cs_new_protected:Npn \l@section #1#2
160 { \__talk_toc_aux:nnnn { 1 } { \bfseries \color { structure } } {#1} {#2} }
161 \cs_new_protected:Npn \l@subsection #1#2
162 {
163   \__talk_toc_aux:nnnn
164   { 2 }
165   {
166     \skip_set:Nn \leftskip { 2em }
167     \color { . }
168   }
169   {#1} {#2}
170 }
171 \cs_new_protected:Npn \l@subsubsection #1#2
172 {
173   \__talk_toc_aux:nnnn
174   { 3 }
175   {
176     \skip_set:Nn \leftskip { 4em }
177     \color { . }
178     \footnotesize
179   }
180   {#1} {#2}
181 }
182 \cs_new_protected:Npn \__talk_toc_aux:nnnn #1#2#3#4
183 {
184   \int_compare:nNnTF { \value { section } } < 1
185   { \use:n }
186   { \__talk_toc_dest:n }
187   { \__talk_toc_level:nnnn {#1} {#2} {#3} {#4} }
188 }

```

We can extract the details for the TOC levels from \@contentsline@destination. At present, that is quite simple-minded: if we are in the current section, show fully, else make semi-opaque. Needs a rounded-out interface but the basic idea will be the same.

```

189 \cs_new_protected:Npn \__talk_toc_dest:n
190 {
191   \exp_after:wN \__talk_toc_dest:w \@contentsline@destination
192   . 0 . 0 . 0 . \q_stop
193 }
194 \cs_new_protected:Npn \__talk_toc_dest:w #1 . #2 . #3 . #4 . #5 \q_stop #6

```

```

195 {
196   \int_compare:nNnTF { \value { section } } = {#2}
197     {#6}
198     {
199       \group_begin:
200         \opacity_select:n { 0.2 }
201         #6
202       \group_end:
203     }
204   }
205   \cs_new_protected:Npn \__talk_toc_level:nnnn #1#2#3#4
206   {
207     \int_compare:nNnF {#1} > { \value { tocdepth } }
208     {
209       \group_begin:
210         \noindent
211         #2
212         \UseHookWithArguments { contentsline / text / before } { 4 }
213         {#1} {#3} {#4} { \@contentsline@destination }
214         #3
215         \UseHookWithArguments { contentsline / text / after } { 4 }
216         {#1} {#3} {#4} { \@contentsline@destination }
217         \UseHookWithArguments { contentsline / page / before } { 4 }
218         {#1} {#3} {#4}
219         { \@contentsline@destination }
220         \UseHookWithArguments { contentsline / page / after } { 4 }
221         {#1} {#3} {#4}
222         { \@contentsline@destination }
223         \par
224       \group_end:
225       \vfil
226     }
227   }

```

(End of definition for \l@section and others. These functions are documented on page ??.)

```

228 \setcounter { tocdepth } { 2 }

```

1.4 Block environments

description (env.) Stub logical environments: needed as the tagging setup expects these to exist.

```

quote (env.) 229 \NewDocumentEnvironment { description } { } { } { } { }
quotation (env.) 230 \NewDocumentEnvironment { quote } { } { } { } { }
verse (env.) 231 \NewDocumentEnvironment { quotation } { } { } { } { }
stdquote (env.) 232 \NewDocumentEnvironment { verse } { } { } { } { }
stdquotation (env.) 233 \AddToHook { begindocument / before }
stdverse (env.) 234 {
235   \clist_map_inline:nn { quote , quotation , verse }
236   {
237     \NewEnvironmentCopy { std #1 } {#1}
238     \RenewDocumentEnvironment {#1} { D <> { all } !0 { } }
239     {
240       \__talk_action_begin:n {##1}
241       \begin { std #1 } [ {##2} ]
242       \ignorespaces

```

```

243     }
244     {
245         \end { std #1 }
246         \__talk_action_end:
247     }
248 }
249 }

```

block (env.)

```

250 \NewDocumentEnvironment { block } { D <> { all } m }
251 {
252     \__talk_action_begin:n {#1}
253     \par
254     \vbox_set:Nw \l__talk_tmp_box
255     \group_begin:
256         \medskip
257         \leavevmode
258         \normalfont \large \bfseries
259         \color { structure }
260         #2
261         \par
262         \medskip
263     \group_end:
264 }
265 {
266     \vbox_set_end:
267     \box_use:N \l__talk_tmp_box
268     \par
269     \__talk_action_end:
270 }

```

1.5 Lists

`\item` Again, add the additional argument: here, we have to do a little gymnastics. The test for an overlay has to come after the standard item definition: in a list, items have to *close* the structure before them first, so if we test too early, we'd end up covering then uncovering straight away!

```

271 \AddToHook { begindocument / before }
272 {
273     \NewCommandCopy \stditem \item
274     \RenewDocumentCommand \item { d <> o }
275     {
276         \IfNoValueTF {#2}
277         { \stditem }
278         { \stditem [ {#2} ] }
279     \IfNoValueTF {#1}
280     {
281         \exp_after:wN \__talk_item_parse_spec:w
282         \l__talk_action_spec_str < all > \q_stop
283     }
284     { \__talk_item_parse_spec:n {#1} }
285 }
286 }

```

Parsing the spec is a separate function here as there are a couple of routes to get here. At present we only have a `false` branch, but for spacing we likely will need to add something to the `true` branch too. The odd stuff with `\currentgrouplevel` here is needed so we only close the item at the correct nesting, allowing for the group that gets added.

```

287 \cs_new_protected:Npn \__talk_item_parse_spec:w #1 < #2 > #3 \q_stop
288 { \__talk_item_parse_spec:n {#2} }
289 \cs_new_protected:Npn \__talk_item_parse_spec:n #1
290 {
291   \tl_if_blank:nF {#1}
292   {
293     \tl_set:Nx \l__talk_list_end_tl
294     {
295       \exp_not:N \int_compare:nNnT \tex_currentgrouplevel:D =
296       { \int_use:N \tex_currentgrouplevel:D + 1 }
297       {
298         \__talk_action_end:
299         \tl_clear:N \exp_not:N \l__talk_list_end_tl
300       }
301     }
302     \__talk_action_begin:n {#1}
303   }
304 }

```

(End of definition for `\item`, `__talk_item_parse_spec:w`, and `__talk_item_parse_spec:n`. This function is documented on page ??.)

`\l__talk_list_end_tl`

```

305 \tl_new:N \l__talk_list_end_tl

```

(End of definition for `\l__talk_list_end_tl`.)

`__block_inter_item:`

`\endblockenv`

There are no currently no hooks for insertion at the end of list items, so we have to do it manually. We cannot target `__block_list_item_end:/__block_list_end:` as these change definition if tagging is suspended.

```

306 \cs_gset_protected:Npn \__block_inter_item:
307 {
308   \legacy_if:nT { @inlabel }
309   { \indent \par }
310   \mode_if_horizontal:T
311   {
312     \__block_skip_remove_last:
313     \__block_skip_remove_last:
314     \par
315   }
316   \l__talk_list_end_tl
317   \__kernel_list_item_end:
318   \__kernel_list_item_begin:
319   \addpenalty \@itempenalty
320   \addvspace \itemsep
321 }
322 \cs_gset:Npn \endblockenv
323 {
324   \__block_debug_typeout:n { blockenv-common~ending \on@line }
325   \bool_if:NT \l__block_level_incr_bool

```

```

326     { \int_gdecr:N \g_block_nesting_depth_int }
327 \legacy_if:nT { @inlabel }
328 {
329     \mode_leave_vertical:
330     \legacy_if_gset_false:n { @inlabel }
331 }
332 \__block_if_list:T
333 { \legacy_if:nT { @newlist } { \@noitemerr } }
334 \mode_if_horizontal:TF
335 {
336     \__block_skip_remove_last:
337     \__block_skip_remove_last:
338     \par
339 }
340 { \@inmatherr { \end { \@currenvir } } }
341 \l__talk_list_end_tl
342 \__kernel_displayblock_end:
343 \__block_if_list:T { \legacy_if_gset_false:n { @newlist } }
344 \legacy_if:nF { @noparlist }
345 {
346     \__block_skip_set_to_last:N \l_tmpa_skip
347     \dim_compare:nNnT \l_tmpa_skip > \c_zero_dim
348     {
349         \skip_vertical:n { - \l_tmpa_skip }
350         \skip_vertical:n { \l_tmpa_skip + \parskip - \@outerparskip }
351     }
352     \addpenalty \@endparpenalty
353     \addvspace \l__block_topsepadd_skip
354 }
355 \socket_use:n { block / endpe }
356 }

```

(End of definition for __block_inter_item: and \endblockenv. This function is documented on page ??.)

```

itemize (env.) Allow for the classical beamer syntax.
enumerate (env.) 357 \AddToHook { begindocument / before }
description (env.) 358 {
359     \clist_map_inline:nn { itemize , enumerate , description }
360     {
361         \RenewDocumentEnvironment {#1} { = { action-spec } !o }
362         {
363             \IfNoValueTF {##1}
364             { \UseInstance { blockenv } {#1} { } }
365             { \UseInstance { blockenv } {#1} {##1} }
366         }
367         { \endblockenv }
368     }
369 }

```

And add the structural color to item labels.

```

370 \AddToHook { begindocument / before }
371 {
372     \EditInstance { item } { basic }
373     { label-format = \color { structure } #1 }

```



```

374 \EditInstance { item } { description }
375     { label-format = \normalfont \bfseries \color { structure } #1 }
376 }

```

`\l__talk_action_spec_str` Add an overlay key to the block template. Placed here, it applies *before* the `\item` starts, so we do not have to redefine the latter to do actions up-front. This also means it can apply to whatever we want it to within a block.

```

377 \keys_define:nn { template / block / display }
378   { action-spec .str_set:N = \l__talk_action_spec_str }

```

(End of definition for `\l__talk_action_spec_str`.)

1.6 Theorems, *etc.*

`\newtheorem` We need to extend the creation of theorems in two ways: add the overlay argument, and
`\stdnewtheorem` add the counter to the list of those reset during overlay creation.

```

379 \NewCommandCopy \stdnewtheorem \newtheorem
380 \RenewDocumentCommand \newtheorem { m O {#1} m o }
381 {
382   \IfNoValueTF {#4}
383     { \stdnewtheorem {#1} [ {#2} ] {#3} }
384     { \stdnewtheorem {#1} [ {#2} ] {#3} [ {#4} ] }
385   \NewEnvironmentCopy { std #1 } {#1}
386   \RenewDocumentEnvironment {#1} { D <> { all } o }
387   {
388     \__talk_action_begin:n {##1}
389     \IfNoValueTF {##2}
390       { \begin { std #1 } }
391       { \begin { std #1 } [ {##2} ] }
392     \ignorespaces
393   }
394   {
395     \end { std #1 }
396     \__talk_action_end:
397   }
398 }

```

(End of definition for `\newtheorem` and `\stdnewtheorem`. These functions are documented on page ??.)

```

399 </class>

```

Part X

ltx-talk-title – Title pages

1 ltx-talk-title implementation

Start the DocStrip guards.

```
1 <{*class>
    Identify the internal prefix.
2 <{@=talk>
```

```
\institute Simple storage at present: we use names similar to the kernel ones for author, etc., as
\subtitle this makes data management easier.
\@institute 3 \cs_new_nopar:Npn \@institute { }
\@subtitle 4 \cs_new_nopar:Npn \@subtitle { }
5 \NewDocumentCommand \institute { = { short-institute } 0 {#2} m }
6 { \cs_gset_nopar:Npn \@institute {#2} }
7 \NewDocumentCommand \subtitle { = { short-subtitle } 0 {#2} m }
8 { \cs_gset_nopar:Npn \@subtitle {#2} }
```

(End of definition for \institute and others. These functions are documented on page ??.)

```
\l__talk_titlelem_after_skip As the various elements of the titlepage share certain characteristics, we use a single
\l__talk_titlelem_before_skip template and split them as instances.
\l__talk_titlelem_color_tl
\l__talk_titlelem_font_tl
\l__talk_titlelem_tag_begin_tl
\l__talk_titlelem_tag_end_tl
9 \NewTemplateType { titlepage-element } { 1 }
10 \DeclareTemplateInterface { titlepage-element } { talk } { 1 }
11 {
12   after-skip : length = 0em ,
13   before-skip : length = 0em ,
14   color : tokenlist = . ,
15   font : tokenlist = \normalfont ,
16   tag-begin : tokenlist = ,
17   tag-end : tokenlist =
18 }
19 \DeclareTemplateCode { titlepage-element } { talk } { 1 }
20 {
21   after-skip = \l__talk_titlelem_after_skip ,
22   before-skip = \l__talk_titlelem_before_skip ,
23   color = \l__talk_titlelem_color_tl ,
24   font = \l__talk_titlelem_font_tl ,
25   tag-begin = \l__talk_titlelem_tag_begin_tl ,
26   tag-end = \l__talk_titlelem_tag_end_tl
27 }
28 {
29   \tl_if_empty:nF {#1}
30   {
31     \vspace { \l__talk_titlelem_before_skip }
32     \group_begin:
33       \tl_if_empty:nF \l__talk_titlelem_color_tl
34       { \color_select:V \l__talk_titlelem_color_tl }
35       \l__talk_titlelem_font_tl
36       \l__talk_titlelem_tag_begin_tl
```

```

37         #1
38         \par
39         \l__talk_titlelem_tag_end_tl
40     \group_end:
41     \vspace { \l__talk_titlelem_after_skip }
42 }
43 }

```

Standard settings are taken from beamer with minor adjustments.

```

44 \DeclareInstance { titlepage-element } { author } { talk }
45 { before-skip = 1em }
46 \DeclareInstance { titlepage-element } { date } { talk }
47 { after-skip = 0.5em }
48 \DeclareInstance { titlepage-element } { institute } { talk }
49 { font = \scriptsize }
50 \DeclareInstance { titlepage-element } { subtitle } { talk }
51 { before-skip = 0.25em , color = structure }
52 \DeclareInstance { titlepage-element } { title } { talk }
53 {
54     color = structure ,
55     font = \Large ,
56     tag-begin = \tag_struct_begin:n { tag = Title } ,
57     tag-end = \tag_struct_end:
58 }

```

(End of definition for \l__talk_titlelem_after_skip and others.)

```

\l__talk_titlepage_order_clist
\l__talk_titlepage_alignment_tl
\l__talk_titlepage_framestyle_tl
\l__talk_frame_alignment_tl

```

Here, we deal with the overall style: notice that frame vertical alignment actually applies elsewhere, which is why it doesn't show up in the template code part. As a result, we have a slightly repetitive key interface.

```

59 \NewTemplateType { titlepage } { 0 }
60 \DeclareTemplateInterface { titlepage } { talk } { 0 }
61 {
62     element-order : commalist =
63     {
64         title      ,
65         subtitle   ,
66         author     ,
67         institute  ,
68         date
69     } ,
70     framestyle : tokenlist = talk ,
71     horizontal-alignment : choice { left , center , right } = center ,
72     vertical-alignment : choice { bottom , center , stretch , top } = center
73 }
74 \DeclareTemplateCode { titlepage } { talk } { 0 }
75 {
76     element-order = \l__talk_titlepage_order_clist ,
77     framestyle = \l__talk_titlepage_framestyle_tl ,
78     horizontal-alignment =
79     {
80         left = \tl_set:Nn \l__talk_titlepage_alignment_tl { flushleft } ,
81         center = \tl_set:Nn \l__talk_titlepage_alignment_tl { center } ,
82         right = \tl_set:Nn \l__talk_titlepage_alignment_tl { flushright }
83     } ,

```

```

84     vertical-alignment =
85     {
86         bottom = \tl_set:Nn \l__talk_frame_alignment_tl { bottom } ,
87         center = \tl_set:Nn \l__talk_frame_alignment_tl { center } ,
88         stretch = \tl_set:Nn \l__talk_frame_alignment_tl { stretch } ,
89         top = \tl_set:Nn \l__talk_frame_alignment_tl { top }
90     }
91 }
92 {
93     \tl_if_empty:NF \l__talk_titlepage_framestyle_tl
94     { \exp_args:NV \thispagestyle \l__talk_titlepage_framestyle_tl }
95     \begin { \l__talk_titlepage_alignment_tl }
96         \cs_set_protected:Npn \and { \quad }
97         \clist_map_inline:Nn \l__talk_titlepage_order_clist
98         {
99             \ExpandArgs { nnv } \UseInstance { titlepage-element }
100             {##1} { @ ##1 }
101         }
102     \end { \l__talk_titlepage_alignment_tl }
103 }

```

(End of definition for \l__talk_titlepage_order_clist and others.)

\maketitle A very simple setup.

```

104 \NewDocumentCommand \maketitle { 0 {} }
105 {
106     \bool_if:NTF \l__talk_frame_bool
107     { \UseTemplate { titlepage } { talk } {##1} }
108     {
109         \begin { frame }
110             \UseTemplate { titlepage } { talk } {##1}
111         \end { frame }
112     }
113 }

```

(End of definition for \maketitle. This function is documented on page ??.)

```

114 \</class>

```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols

@ commands:

\@_decode_overlay_+:nw 121
 \\ 13, 279

A

\abovecaptionskip 138, 140
 \action 102
 actionenv (env.) 102
 \addcontentsline 104
 \addpenalty 319, 352
 \AddToHook 66,
 144, 233, 257, 271, 318, 357, 370, 385
 \addtolength 48
 \addvspace 320, 353
 \alert 79
 alertenv (env.) 89
 \alt 141
 \and 96
 \arabic 382
 \arraycolsep 25
 \arrayrulewidth 27

B

\begin 95, 109, 110, 111, 241, 390, 391
 \begingroup 137, 145, 341
 \belowcaptionskip 139, 141
 \bfseries 21, 39, 160, 258, 375
 \bigskipamount 18
 block (env.) 250
 block commands:
 \g_block_nesting_depth_int 326
 block internal commands:
 __block_debug_typeout:n 324
 __block_if_list:TF 332, 343
 __block_inter_item: 306, 306
 \l_block_level_incr_bool 325
 __block_list_end: 52
 __block_list_item_end: 52
 __block_skip_remove_last:
 312, 313, 336, 337
 __block_skip_set_to_last:N 346
 \l_block_topsepadd_skip 353
 bool commands:
 \bool_do_while:Nn 27
 \bool_gset_false:N ... 30, 36, 40, 406
 \bool_gset_true:N . 203, 211, 217, 398

\bool_if:NTF 6,
 20, 26, 32, 38, 43, 44, 46, 55, 85,
 106, 123, 125, 132, 178, 248, 325, 412
 \bool_lazy_and:nnTF 39, 214, 267
 \bool_lazy_or:nnTF 5, 19
 \bool_new:N 3,
 3, 7, 8, 13, 99, 101, 376, 377, 378
 \bool_set_eq:NN 117, 121
 \bool_set_false:N
 25, 26, 91, 110, 123, 418, 438
 \bool_set_true:N 22, 27,
 42, 59, 141, 178, 198, 213, 391, 426, 446
 \c_false_bool 15
 \c_true_bool 14

box commands:

\box_dp:N 27
 \box_ht:N 60, 225, 250
 \box_move_down:nn 58
 \box_new:N 4, 58
 \box_use:N 267
 \box_use_drop:N
 24, 203, 230, 234, 236, 314
 \box_wd:N 32

box internal commands:

__box_dim_eval:n 24, 27, 32, 35
 __box_set_to_wd: 31, 36

C

\clearpage 92
 clist commands:
 \clist_const:Nn 48
 \clist_if_in:NnTF 55, 177
 \clist_map_break: 199, 218
 \clist_map_inline:Nn 97, 180, 305
 \clist_map_inline:nn
 59, 79, 120, 121, 235, 359
 \clist_new:N 10, 14
 \clist_pop:NNTF 301
 \clist_set:Nn 47, 176
 \color .. 4, 11, 160, 167, 177, 259, 373, 375
 color commands:
 \color_group_begin: 25, 37
 \color_group_end: 25
 \color_math:nn 9, 25
 \color_math:nnn 10, 26
 \color_select:n 7, 16,
 21, 22, 34, 35, 36, 50, 204, 246, 299, 330
 \color_select:nn 8, 17, 37

96, 97, 97, 97, 98, 98, 99, 100, 101, 101, 102, 102, 103, 103, 104, 106, 106, 107, 107, 109, 110, 111, 136, 256, 268, 295, 299, 301, 302, 303, 324	\hook_use:n 114
\exp_not:n 258, 326, 327, 429	\hspace 201, 208
\exp_stop_f: 42, 43, 44	\hypersetup 143
\expandafter 146	
\ExpandArgs 61, 71, 81, 89, 99, 250, 297, 298, 303, 308	
F	I
\fboxrule 32	\IfBooleanT 325
\fboxsep 31	\ifcase 5
\fi 18, 147	\IfFormatAtLeastF 7
figure (env.) 120	\IfNoValueF 326, 327
\figurename 131	\IfNoValueTF 15, 24, 35, 44, 185, 276, 279, 363, 382, 389
file commands:	\ignorespaces ... 18, 80, 149, 195, 242, 392
\file_if_exist_input:nTF 87	\immediate 147
\file_input:n 89	\includegraphics 316
\footins 30	\indent 309
\footnotesize 178	\insertsection 81
\footskip 288, 289	\insertsubsection 81
fp commands:	\insertsubsubsection 81
\fp_eval:n 92	\institute 3
\fp_to_dim:n 102	int commands:
\frame 24, 26, 387	\int_compare:nNnTF 106, 184, 196, 196, 202, 207, 208, 210, 295
frame 412	\int_compare_p:nNn 215, 216
frame* 412	\int_eval:n 188
\framesubtitle 10	\int_gdecr:N 326
\frametitle 5, 419, 429	\int_gincr:N 29, 126, 186, 390
	\int_gset:Nn 187, 397
	\int_gset_eq:NN 129, 134, 139
	\int_gzero:N 25, 74
	\int_incr:N 218
	\int_max:nn 171
	\int_new:N . 4, 5, 71, 128, 145, 208, 379
G	\int_to_Roman:n 211
\gdef 256	\int_to_roman:n 212
\geometry 3	\int_use:N .. 8, 14, 52, 56, 68, 296, 384
group commands:	\c_max_int 190, 216
\group_begin: 11, 32, 33, 34, 58, 60, 64, 84, 104, 115, 155, 199, 202, 209, 244, 255, 329	\invisible 59
\c_group_begin_token 34	invisibleenv (env.) 71
\group_end: 39, 40, 40, 54, 62, 63, 69, 87, 108, 139, 157, 202, 207, 224, 254, 263, 333	iow commands:
\group_insert_after:N ... 36, 159, 161	\iow_now:Nn 254
	\item 54, 271
	itemize (env.) 357
	\itemsep 54, 62, 69, 320
H	J
hbox commands:	\jobname 140, 147
\hbox:n 56	
\hbox_set_end: 23	
\hbox_set_to_wd:Nnw 14	K
\headsep 222, 241, 242	kernel internal commands:
\hfil 17, 22, 86, 272, 312, 341, 353, 358, 369	__kernel_displayblock_end: 342
hook commands:	__kernel_list_item_begin: 318
\hook_gput_code:nnn 48, 91, 262	__kernel_list_item_end: 317
\hook_new:n 116	keys commands:
	\l_keys_choice_tl 70

<code>\keys_define:nn</code>	3, 5, 28, 60, 72, 149, 377	<code>\NewCommandCopy</code>
<code>\keys_set:nn</code> 7,	4, 5, 6, 273, 297, 317, 379, 387
	13, 46, 73, 79, 162, 417, 425, 437, 445	<code>\newcounter</code> 18, 72, 73, 74, 131
<code>\l_keys_value_tl</code> 43, 159	<code>\NewDocumentCommand</code>
		5, 5, 7, 10, 11, 42, 61, 81,
			87, 92, 102, 104, 141, 147, 153, 173, 183
L		<code>\NewDocumentEnvironment</code>
<code>\label</code> 332	9, 69, 71, 89, 110, 122, 197,
<code>\labelenumi</code> 34		214, 229, 230, 231, 232, 250, 422, 442
<code>\labelenumii</code> 35	<code>\NewEnvironmentCopy</code> 237, 385
<code>\labelenumiii</code> 36	<code>\newlabel</code> 345
<code>\labelenumiv</code> 37	<code>\newlength</code> 138, 139
<code>\labelitemfont</code> 38, 39, 40, 41, 42	<code>\NewTemplateType</code>
<code>\labelitemi</code> 38	9, 15, 46, 59, 92, 183, 216, 263
<code>\labelitemii</code> 39	<code>\newtheorem</code> 379
<code>\labelitemiii</code> 40	<code>\newwrite</code> 146
<code>\labelitemiv</code> 41	<code>\nobreakspace</code> 136
<code>\labelsep</code> 29, 33, 46, 48	<code>\noindent</code> 31, 210, 238, 285
<code>\labelwidth</code> 47, 48	<code>\normalfont</code> 15, 42, 221, 258, 375
<code>\Large</code> 21, 55	<code>\normalsize</code> 148
<code>\large</code> 258	<code>\number</code> 20, 22
<code>\leavevmode</code> 78, 257	<code>\numberline</code> 109
<code>\leftmargin</code> 51, 59, 66		
<code>\leftmargini</code> 43, 47, 51	O	
<code>\leftmarginii</code> 44, 59	<code>\obeyedline</code> 15, 59
<code>\leftmarginiii</code> 45, 66	<code>\only</code> 39, 59, 94
<code>\leftskip</code> 166, 176	<code>\onlyenv (env.)</code> 71
legacy commands:		<code>\onslide</code> 147
<code>\legacy_if:nTF</code>	opacity commands:	
.....	142, 252, 308, 327, 333, 344	<code>\opacity_select:n</code> 27,
<code>\legacy_if_gset_false:n</code> 330, 343	28, 33, 34, 51, 56, 155, 158, 170, 200
		<code>\openout</code> 147
M		<code>\or</code>	.. 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
<code>\makeatletter</code> 138	<code>overlayarea (env.)</code> 197
<code>\maketitle</code> 104	<code>overprint (env.)</code> 214
<code>\mathcolor</code> 5, 11		
<code>\mbox</code> 310	P	
<code>\medskip</code> 256, 262	<code>\pagestyle</code> 375
<code>\mode</code> 32, 11	<code>\paperwidth</code> 293, 331, 332
mode commands:		<code>\par</code> 27, 12, 25, 25, 38,
<code>\mode_if_horizontal:TF</code> 310, 334	39, 72, 87, 144, 150, 192, 206, 223,
<code>\mode_if_math:TF</code> 309		237, 253, 261, 268, 309, 314, 338, 408
<code>\mode_leave_vertical:</code> 33, 312, 329	<code>\parsep</code> 53, 61, 62, 68, 69
<code>\month</code> 5	<code>\parskip</code> 350
msg commands:		<code>\partopsep</code> 71
<code>\msg_error:nnn</code> 116, 161	<code>\pause</code> 183
<code>\msg_fatal:nn</code> 17	prg commands:	
<code>\g_msg_module_name_prop</code> 5	<code>\prg_generate_conditional_-</code>	
<code>\g_msg_module_type_prop</code> 6	variant:Nnn 10
<code>\msg_new:nnn</code> 277	<code>\prg_new_protected_conditional:Nppn</code> 3, 3
<code>\msg_new:nnnn</code> 9, 222	<code>\prg_return_false:</code> 8, 9
<code>\msg_warning:nn</code> 273	<code>\prg_return_true:</code> 7, 8
		<code>\ProcessedArgument</code> 14, 15
N			
<code>\NeedsDocumentMetadata</code> 19		

<code>\ProcessKeyOptions</code>	86	<code>\setcounter</code>	24, 228
prop commands:		<code>\setlength</code>	25, 26, 27, 28, 29, 31, 32, 33, 43, 44, 45, 46, 47, 71, 140, 141
<code>\prop_gput:Nnn</code>	5, 6	<code>\setmathfont</code>	134
property commands:		<code>\setsansfont</code>	133
<code>\property_new:nnnn</code>	8, 383	<code>\SetTemplateKeys</code>	109
<code>\property_record:nn</code>	52, 68, 386	<code>\sfdefault</code>	140
<code>\property_ref:nn</code>	14	<code>\skip</code>	30
<code>\protect</code>	109	skip commands:	
<code>\ProvidesExplClass</code>	3	<code>\skip_horizontal:n</code>	243, 290, 315, 326, 332
Q			
<code>\quad</code>	96	<code>\skip_set:Nn</code>	166, 176
quark commands:		<code>\skip_vertical:n</code>	96, 98, 102, 104, 108, 110, 114, 116, 349, 350
<code>\quark_if_recursion_tail_stop:N</code>	131	<code>\l_tmpa_skip</code>	346, 347, 349, 350
<code>\quark_if_recursion_tail_stop_-</code> <code>do:Nn</code>	148, 159	socket commands:	
<code>\quark_if_recursion_tail_stop_-</code> <code>do:nn</code>	37	<code>\socket_use:n</code>	355
<code>\q_recursion_stop</code>	34, 124, 151	<code>\space</code>	19, 21
<code>\q_recursion_tail</code>	34, 124, 151	<code>\stdcolor</code>	4
<code>\q_stop</code>	63, 71, 97, 98, 102, 107, 181, 184, 192, 194, 282, 287	<code>\stdemph</code>	282
quotation (env.)	229	<code>\stdincludgraphics</code>	316
quote (env.)	229	<code>\stditem</code>	273, 277, 278
R		<code>\stdmathcolor</code>	4
<code>\raggedright</code>	79, 128, 194	<code>\stdnewtheorem</code>	379
<code>\refstepcounter</code>	95	<code>\stdquotation (env.)</code>	229
<code>\relax</code>	147	<code>\stdquote (env.)</code>	229
<code>\relsize</code>	92	<code>\stdtextbf</code>	282
<code>\RenewCommandCopy</code>	26	<code>\stdtextcolor</code>	4
<code>\RenewDocumentCommand</code>	11, 20, 29, 94, 274, 298, 318, 332, 380	<code>\stdtextit</code>	282
<code>\RenewDocumentEnvironment</code>	238, 361, 386, 414, 434	<code>\stdtextmd</code>	282
<code>\RequirePackage</code>	3, 90, 111, 129, 132, 137, 138, 142, 316	<code>\stdtextnormal</code>	282
<code>\rmdefault</code>	140	<code>\stdtextrm</code>	282
rule commands:		<code>\stdtextsc</code>	282
<code>\rule:nnn</code>	39, 331	<code>\stdtextsf</code>	282
S		<code>\stdtextsl</code>	282
scan commands:		<code>\stdtexttt</code>	282
<code>\scan_stop:</code>	45	<code>\stdtextup</code>	282
<code>\scriptsize</code>	49	<code>\stdverse (env.)</code>	229
<code>\section</code>	72, 81	<code>\stepcounter</code>	18
seq commands:		str commands:	
<code>\seq_gput_right:Nn</code>	146	<code>\str_clear:N</code>	18, 28, 29
<code>\seq_gremove_all:Nn</code>	75, 76, 77	<code>\str_if_empty:NTF</code>	86
<code>\seq_map_indexed_inline:Nn</code>	83	<code>\str_if_empty_p:N</code>	40
<code>\seq_map_inline:Nn</code>	126, 133, 138	<code>\str_if_eq:nnTF</code> ..	17, 57, 83, 100, 102
<code>\seq_new:N</code>	118	<code>\str_if_eq_p:nn</code>	6, 7, 21
<code>\seq_set_from_clist:Nn</code>	81, 119	<code>\str_new:N</code>	9, 11, 12, 15, 100
<code>\l_tmpa_seq</code>	81, 83	<code>\str_put_right:Nn</code>	134, 170
		<code>\str_replace_all:Nnn</code>	20, 22, 54
		<code>\str_set:Nn</code>	18, 24, 70, 111
		<code>\str_set_eq:NN</code>	119
		<code>\string</code>	345
		<code>\subsection</code>	72, 81
		<code>\subsubsection</code>	72, 81
		<code>\subtitle</code>	3

sys commands:

- \sys_if_engine_opentype:TF 130

T

- \tabbingssep 29
- \tabcolsep 26
- table (env.) 120
- \tablename 131
- \tableofcontents 153

tag commands:

- \tag_get:n 397
- \tag_mc_begin:n 173, 180, 404
- \tag_mc_end: 171, 178, 410
- \tag_resume:n 170, 409
- \tag_struct_begin:n 56, 172, 396
- \tag_struct_end: 57, 179, 400
- \tag_suspend:n 181, 405
- \tag_tool:n 96
- \tagpdfparaOff 61
- \tagpdfsetup 144

talk internal commands:

- __talk_action:N 17, 17
- __talk_action_alert:N 18, 18
- __talk_action_begin:n 11, 71, 102, 105, 111, 113, 124, 240, 252, 302, 388
- __talk_action_end: . . 26, 88, 102, 107, 112, 129, 130, 246, 269, 298, 396
- __talk_action_invisible:N . . 24, 24
- __talk_action_only:N 36
- __talk_action_only_begin:N 36
- __talk_action_only_end:N . . . 36, 41
- \l__talk_action_spec_str 149, 282, 377
- __talk_action_uncover:N 53, 53
- __talk_action_visible:N 24, 30
- \l__talk_aspect_ratio_str . . 60, 106
- \l__talk_cnt_reset_seq 75, 76, 77, 118, 133, 138, 146
- __talk_cnt_restore: 86, 131, 136
- __talk_cnt_save: 77, 131, 131
- __talk_column_align_bottom:n 50, 50
- __talk_column_align_center:n 50, 52
- __talk_column_align_top:n . . 50, 67
- \l__talk_column_alignment_tl . 28, 84
- \l__talk_columns_wd_tl 5, 14, 15
- __talk_decode_action:n . . 85, 94, 94
- __talk_decode_action:w . . 94, 96, 101
- \l__talk_decode_action_str 12, 18, 111, 120, 125
- \l__talk_decode_actions_bool 13, 25, 122, 126
- \l__talk_decode_actions_clist . . 13
- \l__talk_decode_actions_str . . 13, 29
- \l__talk_decode_arg_str 9, 24, 30, 117, 162
- __talk_decode_check:n . 127, 174, 174
- __talk_decode_check:nw 174, 181, 184
- __talk_decode_check_range:nnn 174, 190, 191, 206
- __talk_decode_check_single:nn 174, 187, 194
- __talk_decode_mode:n 44, 53, 53
- __talk_decode_mode:nn . . . 76, 79, 81
- __talk_decode_mode:w 53, 62, 68
- __talk_decode_mode_aux:n 53
- __talk_decode_overlay_:nw 121
- __talk_decode_overlay_aux:nNn 121, 142, 145, 146
- __talk_decode_overlay_offset:nNn 121, 150, 155, 165, 168
- __talk_decode_overlay_offset:nNnN 121, 154, 157, 166
- __talk_decode_overlays:nN 121, 124, 129, 135, 172
- __talk_decode_overlays:nn 87, 106, 113, 121, 121
- \l__talk_decode_overlays_bool 3, 6, 22, 26, 42, 59, 118, 123
- \l__talk_decode_overlays_clist . . 10
- \l__talk_decode_overlays_str 10, 28, 40, 86
- __talk_decode_parse:n . 5, 16, 16, 116
- __talk_decode_parse:w . 16, 34, 35, 46
- __talk_decode_parse_aux:n 16, 30, 33
- \l__talk_decode_pure_bool 7, 27, 41, 91, 110
- \l__talk_decode_step_bool 8, 123, 125, 141
- \l__talk_float_alignment_tl 91, 103, 104, 105, 111, 117
- \l__talk_fontsize_dim 60, 87, 92
- \l__talk_footelem_color_tl 183
- \l__talk_footelem_font_tl 183
- \l__talk_footelem_left_skip 183
- \l__talk_footelem_right_skip . . . 183
- \l__talk_footer_bg_tl 263
- \l__talk_footer_fg_tl 263
- \l__talk_footer_font_tl 263
- \l__talk_footer_left_skip 263
- \l__talk_footer_order_clist 263
- \l__talk_footer_right_skip 263
- \l__talk_footer_sep_tl 263
- \l__talk_frame_alignment_tl 59, 89, 148, 158
- \l__talk_frame_bool 106, 376, 391
- \g__talk_frame_int 14, 52, 68, 211, 379, 384, 390
- __talk_frame_notag:n 41, 402, 402

__talk_frame_overprint:	__talk_onslide:n
. 209, 209, 221, 224, 228,	\g__talk_onslide_escape_tl
241, 243, 244, 247, 249, 256, 258, 263 156, 159, 161, 165, 171
__talk_frame_process:nn	__talk_onslide_reset:
. 388, 388, 419, 428, 439, 447 147, 153, 166, 170
\g__talk_frame_struct_int 56, 71, 397	\g__talk_onslide_tl
\g__talk_frame_subtitle_tl 3, 13, 76 79, 83, 151, 162, 164, 171
__talk_frame_tag:n 37, 394, 394	__talk_overlay_arg:n
\g__talk_frame_tag_bool 3, 11, 62, 72, 82, 90
. 46, 377, 398, 406	__talk_overprint_begin:n
\l__talk_frame_tagging_str 190, 190, 198, 215
. 17, 18, 20, 22, 34, 149	__talk_overprint_check_ht:n
__talk_frame_title:n 15, 38, 44 214, 263, 265, 274
\l__talk_frame_title_bool 60, 412	\l__talk_overprint_int 208, 212, 218
__talk_frame_title_tagged:n	__talk_overprint_save_ht:
. 15, 47, 51 214, 219, 239
\g__talk_frame_title_tl	\c__talk_paper_height_dim 94
. 3, 8, 58, 75, 252, 427	\c__talk_paper_width_dim 94
\l__talk_frame_verb_bool	\g__talk_pauses_int
. 44, 378, 418, 426, 438, 446 8, 4, 74, 126, 171, 186, 187, 188
\l__talk_frametitle_after_skip	\l__talk_saved_action_str 99, 119, 135
. 25, 41	\l__talk_saved_actions_bool
\l__talk_frametitle_before_skip 99, 121, 137
. 26, 32	\l__talk_saved_overlays_bool
\l__talk_frametitle_color_tl 99, 117, 132
. 27, 34, 35	__talk_section_tagged:
\l__talk_frametitle_font_tl 28, 36 119, 119, 340, 351, 367
\l__talk_header_bg_tl 216	\g__talk_section_tl 66
\l__talk_header_fg_tl 216	\l__talk_section_tl 66
\l__talk_header_font_tl 216	__talk_slide:nn 9, 9, 392
\l__talk_header_frametitle_bool 216	__talk_slide_align_bottom:n 94, 94
\l__talk_header_ht_dim 216	__talk_slide_align_center:n 94, 100
\l__talk_header_left_skip 216	__talk_slide_align_stretch:n 94, 106
\l__talk_header_right_skip 216	__talk_slide_align_top:n 94, 112
__talk_header_tag_begin:n	__talk_slide_aux:n 9, 45, 56
. 53, 125, 168, 168, 175	__talk_slide_begin: 33, 72, 72
__talk_header_tag_end:	\l__talk_slide_box 4, 78, 90
. 64, 130, 168, 176	\g__talk_slide_continue_bool 3,
__talk_if_overlay:n 3, 10	27, 30, 36, 40, 85, 178, 203, 211, 217
__talk_if_overlay:nTF	__talk_slide_end: 49, 72, 81
. 3, 7, 12, 13, 13, 22, 31,	\g__talk_slide_int
31, 34, 96, 143, 152, 175, 301, 320, 335 5, 8, 25, 29, 196, 202, 208, 210, 215
__talk_item_parse_spec:n	\g__talk_subsection_tl 66
. 271, 284, 288, 289	\l__talk_subsection_tl 66, 101
__talk_item_parse_spec:w	\g__talk_subsubsection_tl 66
. 271, 281, 287	\l__talk_subsubsection_tl 66, 103
__talk_label:n 332, 336, 339	__talk_textcmd_equiv:n 282, 303, 307
__talk_latex_frame:n 26, 387, 387	\l__talk_titlelem_after_skip 9
\l__talk_list_end_tl	\l__talk_titlelem_before_skip 9
. 293, 299, 305, 316, 341	\l__talk_titlelem_color_tl 9
__talk_mode:n 3	\l__talk_titlelem_font_tl 9
__talk_mode:nTF 3, 12	\l__talk_titlelem_tag_begin_tl 9
\l__talk_mode_str 7, 58, 60, 83	\l__talk_titlelem_tag_end_tl 9
\c__talk_modes_clist 48, 55	\l__talk_titlepage_alignment_tl 59

<code>\l__talk_titlepage_framestyle_tl</code>	59	<code>\setminipage</code>	147
<code>\l__talk_titlepage_order_clist</code>	59	<code>\starttoc</code>	135, 156
<code>__talk_tmp:w</code>	57, 57, 97, 106	<code>\@subtitle</code>	3
<code>\l__talk_tmp_box</code>	14,	<code>\c@figure</code>	131
	24, 39, 58, 60, 74, 85, 193, 203, 225,	<code>\c@frame</code>	379
	230, 234, 236, 250, 254, 267, 291, 314	<code>\c@page</code>	141
<code>\l__talk_tmp_tl</code>		<code>\c@pauses</code>	4
	12, 18, 21, 23, 59, 100, 301, 303, 304	<code>\c@section</code>	78
<code>__talk_toc_aux:nnnn</code>		<code>\c@slide</code>	5
	159, 160, 163, 173, 182	<code>\c@subsection</code>	79
<code>__talk_toc_dest:n</code>	159, 186, 189	<code>\c@subsubsection</code>	80
<code>__talk_toc_dest:w</code>	159, 191, 194	<code>\c@table</code>	131
<code>__talk_toc_level:nnnn</code>	159, 187, 205	<code>\currentgrouplevel</code>	52
<code>\l__talk_uncover_hidden_fp</code>	46	<code>\fnum@figure</code>	131
<code>__talk_wallpaper_hruler:Nnn</code>		<code>\fnum@table</code>	131
	239, 286, 324, 324	<code>\Gm@bmargin</code>	289
<code>\temporal</code>	173	<code>\Gm@lmargin</code>	223, 270, 326
TeX and L ^A T _E X 2 _ε commands:			
<code>\@arabic</code>	6, 7, 78, 79, 80, 141, 381	<code>\Gm@rmargin</code>	225, 271, 315
<code>\@auxout</code>	254, 343	<code>\Gm@tmargin</code>	222
<code>\@bsphack</code>	334	<code>\if@minipage</code>	147
<code>\@caption</code>	142	<code>\ignorespaces</code>	29
<code>\@capttype</code>	112	<code>\l@section</code>	159
<code>\@contentsline@destination</code>		<code>\l@subsection</code>	159
	49, 191, 213, 216, 219, 222	<code>\l@subsubsection</code>	159
<code>\@currentHref</code>	350	<code>\on@line</code>	324
<code>\@currentlabel</code>	347	<code>\protected@write</code>	343
<code>\@currentlabelname</code>	349	<code>\ps@plain</code>	336
<code>\@currenvir</code>	340	<code>\ps@talk</code>	336
<code>\@definecounter</code>	141	<code>\ps@wallpaper</code>	336
<code>\@endparpenalty</code>	352	<code>\std@definecounter</code>	141
<code>\@esphack</code>	337	tex commands:	
<code>\@evenfoot</code>	345, 361, 373	<code>\tex_currentgrouplevel:D</code>	295, 296
<code>\@evenhead</code>	344, 360, 372	<code>\tex_fontdimen:D</code>	61
<code>\@framenumber</code>	379	<code>\tex_hsize:D</code>	24, 35
<code>\@ignore</code>	29	<code>\tex_setbox:D</code>	22, 33
<code>\@ignoretrue</code>	89	<code>\tex_textfont:D</code>	61
<code>\@inmatherr</code>	340	<code>\tex_vbox:D</code>	22, 33
<code>\@input</code>	140	<code>\tex_vrule:D</code>	41
<code>\@institute</code>	3	text commands:	
<code>\@itempenalty</code>	319	<code>\text_purify:n</code>	55, 58, 128
<code>\@kernel@reserved@label@data</code>	351	<code>\text_titlecase_first:n</code>	133
<code>\@listI</code>	56	<code>\textasteriskcentered</code>	40
<code>\@listi</code>	49, 56	<code>\textbf</code>	282
<code>\@listii</code>	57	<code>\textbullet</code>	38
<code>\@listiii</code>	64	<code>\textcolor</code>	6, 11
<code>\@makecaption</code>	149	<code>\textendash</code>	39
<code>\@mpfootins</code>	30	<code>\textheight</code>	87
<code>\@nobreakfalse</code>	150	<code>\textit</code>	282
<code>\@noitemerr</code>	333	<code>\textmd</code>	282
<code>\@oddfoot</code>	343, 345, 355, 361, 371, 373	<code>\textnormal</code>	282
<code>\@oddhead</code>	338, 344, 349, 360, 365, 372	<code>\textperiodcentered</code>	41
<code>\@outerparskip</code>	350	<code>\textrm</code>	282
<code>\@parboxrestore</code>	77, 146	<code>\textsc</code>	282
		<code>\textsf</code>	282

