

ltx-talk – A class for typesetting presentations*

Joseph Wright[†]

Released 2025-11-29

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	2
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 v0.3.2, last revised 2025-11-29.

[†]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
1.3	Footnotes	31
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	59

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} {2025-11-29} {0.3.2}
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     }
15     \msg_fatal:nn { ltx-talk } { kernel-too-old }
16 }
17 \NeedsDocumentMetadata
```

1.2 Additions for expl3

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

```
18 \cs_gset_protected:Npn \vbox_set_to_wd:Nnn #1#2#3
19 {
20     \tex_setbox:D #1 \tex_vbox:D
21     {
22         \tex_hsize:D \__box_dim_eval:n {#2}
23         \color_group_begin: #3 \par \color_group_end:
24     }
25     \box_dp:N #1 \__box_dim_eval:n {#2}
26 }
27 \cs_gset_protected:Npn \vbox_set_to_wd:Nnw #1#2
28 {
29     \cs_set_protected:Npn \__box_set_to_wd:
30     { \box_wd:N #1 \__box_dim_eval:n {#2} }
31     \tex_setbox:D #1 \tex_vbox:D
32     \c_group_begin_token
```

```

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

```

1.3 Extra variants

```

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

```

1.4 Scratch space

_talk_tmp:w For one-off processing.

```

55 \cs_new_protected:Npn \_talk_tmp:w { }

```

(End of definition for _talk_tmp:w.)

\l_talk_tmp_box

```

56 \box_new:N \l\_talk_tmp_box

```

(End of definition for \l_talk_tmp_box.)

\l_talk_tmp_tl

```

57 \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
58 \keys_define:nn { talk }
59 {
60     aspect-ratio .str_set:N =
61     \l\_talk_aspect_ratio_str ,
62     font-size .dim_set:N =
63     \l\_talk_fontsize_dim ,
64     frame-title-arg .bool_set:N =

```

```

65     \l__talk_frame_title_bool ,
66     handout .code:n =
67     { \str_set:Nn \l__talk_mode_str { handout } } ,
68     handout .value_forbidden:n = true ,
69     mode .choices:nn =
70     { handout , projector }
71     { \str_set:NV \l__talk_mode_str \l_keys_choice_tl }
72 }

```

(End of definition for \l__talk_aspect_ratio_str and others.)

Scope for options.

```

73 \keys_define:nn { talk }
74 {
75     aspect-ratio .usage:n = load ,
76     font-size .usage:n = load ,
77     frame-title-arg .usage:n = load ,
78     mode .usage:n = load
79 }

```

Initial values.

```

80 \keys_set:nn { talk }
81 {
82     aspect-ratio = 16:9 ,
83     font-size = 11pt ,
84     frame-title-arg = false ,
85     mode = projector
86 }
87 \ProcessKeyOptions [ talk ]

```

1.6 Setting up

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

```

88 \file_if_exist_input:nF { size \dim_to_decimal:n \l__talk_fontsize_dim .clo }
89 {
90     \file_input:n { size10.clo }
91     \RequirePackage { relsize }
92     \hook_gput_code:nne { begindocument } { talk }
93     { \exp_not:N \relsize { \fp_eval:n { \l__talk_fontsize_dim / 10pt } } }
94 }

```

\c__talk_paper_height_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.

```

95 \dim_const:Nn \c__talk_paper_height_dim { 100mm }
96 \use:e
97 {
98     \cs_set_protected:Npn \exp_not:N \__talk_tmp:w
99     #1 \tl_to_str:n { : } #2 \tl_to_str:n { : } #3 \exp_not:N \q_stop
100     {
101         \dim_const:Nn \exp_not:N \c__talk_paper_width_dim
102         {
103             \exp_not:N \fp_to_dim:n
104             { (#1 / #2) * \exp_not:N \c__talk_paper_height_dim }
105         }

```

```

106     }
107     \exp_not:N \__talk_tmp:w \l__talk_aspect_ratio_str
108     \tl_to_str:n { : } 100 \exp_not:N \q_stop
109 }
110 \use:e
111 {
112     \exp_not:N \RequirePackage
113     [
114         papersize =
115         {
116             \dim_use:N \c__talk_paper_width_dim ,
117             \dim_use:N \c__talk_paper_height_dim
118         } ,
119         tmargin    = 10mm ,
120         bmargin    = 8mm ,
121         lmargin    = 10mm ,
122         rmargin    = 10mm ,
123         headheight = 10mm ,
124         headsep    = 2mm ,
125         footskip   = 6mm
126     ]
127     { geometry }
128 }

```

(End of definition for `\c__talk_paper_height_dim` and `\c__talk_paper_width_dim`.)

Turn off justification

```

129 \raggedright

```

1.7 Math support

We always require `amsmath`: this is forced anyway by `unicode-math` for Lua_{TeX}.

```

130 \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; we also load `mathtools` as that has to be before `unicode-math`. 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.

```

131 \sys_if_engine_opentype:TF
132 {
133     \RequirePackage { mathtools }
134     \RequirePackage { unicode-math }
135     \setsansfont { NewCMSans10-Regular.otf }
136     \setmathfont { NewCMSansMath-Regular.otf }
137 }
138 {

```

```

139 \RequirePackage { sansmathfonts }
140 \RequirePackage [ nomath ] { lmodern }
141 }
142 \cs_set_eq:NN \rmdefault \sfdefault

```

To ensure that math mode fonts are always initialized, force loading at the start of the document. This is left as late as possible: just before typesetting starts

```

143 \AddToHook { begindocument / end } { \check@mathfonts }

```

1.9 Hyperlinks

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

```

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

```

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

A requirement.

```

145 \RequirePackage { hyperref }
146 \hypersetup { hidelinks }

```

1.10 Tagging

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

```

147 \tagpdfsetup
148 {
149   role / user-NS = ltx-talk      ,
150   role / new-tag = frame / Sect  ,
151   role / new-tag = frametitle / H4
152 }
153 </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:nNN
  \__talk_decode_overlay_offset:nNnN
  \__talk_decode_overlay_offset:nNn

```

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.

```

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:nNN {#1} 1
143 }
144 \cs_new_protected:cpn { __talk_decode_overlay_.:nw } #1

```

```
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
95 {
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
```

`__talk_slide_align_stretch:n`

`__talk_slide_align_top:n`

```

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-hspace    : length = 0em ,
189   right-hspace   : 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-hspace    = \l__talk_footelem_left_skip ,
196   right-hspace   = \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 } { subtitle } { talk } { }
215 \DeclareInstance { footer-element } { institute } { talk } { }
216 \DeclareInstance { footer-element } { framenummer } { 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.

```

217 \NewTemplateType { header } { 0 }
218 \DeclareTemplateInterface { header } { talk } { 0 }
219 {
220     background-color : tokenlist,
221     color             : tokenlist = structure ,
222     font              : tokenlist = \normalfont ,
223     height            : length = \Gm@tmargin + \headsep ,
224     left-hspace       : skip = \Gm@lmargin ,
225     print-frame-title : boolean = true ,
226     right-hspace      : skip = \Gm@rmargin
227 }
228 \DeclareTemplateCode { header } { talk } { 0 }
229 {
230     background-color = \l__talk_header_bg_tl ,
231     color            = \l__talk_header_fg_tl ,
232     font             = \l__talk_header_font_tl ,
233     height           = \l__talk_header_ht_dim ,
234     left-hspace      = \l__talk_header_left_skip ,
235     print-frame-title = \l__talk_header_frametitle_bool ,
236     right-hspace     = \l__talk_header_right_skip
237 }
238 {
239     \noindent
240     \__talk_wallpaper_hruler:Nnn
241     \l__talk_header_bg_tl
242     { \l__talk_header_ht_dim - \headsep }
243     { \headsep }
244     \skip_horizontal:n { \l__talk_header_left_skip }
245     \group_begin:
246     \tl_if_empty:NF \l__talk_header_fg_tl
247     { \color_select:V \l__talk_header_fg_tl }
248     \l__talk_header_font_tl
249     \bool_if:NT \l__talk_header_frametitle_bool
250     {
251         \ExpandArgs { nnV }
252         \UseInstance { frametitle } { header }
253         \g__talk_frame_title_tl
254     }
255     \group_end:
256 }
257 \DeclareInstance { header } { std } { talk } { }
258 \AddToHook { begindocument }
259 {
260     \DeclareInstanceCopy { header } { wallpaper } { std }
261     \EditInstance { header } { wallpaper }
262     { print-frame-title = false }
263 }

```

(End of definition for \l__talk_header_bg_tl and others.)

<pre> \l__talk_footer_bg_tl \l__talk_footer_fg_tl \l__talk_footer_font_tl \l__talk_footer_order_clist \l__talk_footer_sep_tl \l__talk_footer_left_skip \l__talk_footer_right_skip </pre>	<p>Templates for the footer area. Again the margins are handled in stages: here we do have a box for the content so the right skip is used, and we avoid an overfull box by including consideration of the right margin of the page layout.</p> <pre> 264 \NewTemplateType { footer } { 0 } </pre>
--	--


```

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

```

```

319 \box_use_drop:N \l__talk_tmp_box
320 \skip_horizontal:n { \l__talk_footer_right_skip - \Gm@rmargin }
321 }
322 \DeclareInstance { footer } { std } { talk } { }
323 \AddToHook { begindocument }
324 {
325 \DeclareInstanceCopy { footer } { wallpaper } { std }
326 \EditInstance { footer } { wallpaper }
327 { element-order = }
328 }

```

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

`__talk_metadata_name:n` A simple auxiliary to shorten metadata names but not the frame number. This is used as all the standard metadata is stored in two forms, and other than on the title page we use the shorter version. Full expansion is applied as this avoids any issue with stored names.

```

329 \cs_new:Npn \__talk_metadata_name:n #1
330 {
331 \str_if_eq:eeTF {#1} { framenumbers }
332 {#1}
333 { short #1 }
334 }

```

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

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

```

335 \cs_new_protected:Npn \__talk_wallpaper_hrule:Nnn #1#2#3
336 {
337 \skip_horizontal:n { -\Gm@lmargin }
338 \tl_if_empty:NF #1
339 {
340 \group_begin:
341 \color_select:V #1
342 \rule:nnn { \paperwidth } {#2} {#3}
343 \skip_horizontal:n { -\paperwidth }
344 \group_end:
345 }
346 }

```

(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.

```

347 \cs_set_nopar:Npn \ps@plain
348 {
349 \cs_set_nopar:Npn \@oddhead
350 {
351 \hfil
352 }
353 \cs_set_nopar:Npn \@oddfoot { }
354 \cs_set_eq:NN \@evenhead \@oddhead

```

```

355   \cs_set_eq:NN \@evenfoot \@oddfoot
356 }
357 \cs_set_nopar:Npn \ps@wallpaper
358 {
359   \cs_set_nopar:Npn \@oddhead
360   {
361     \UseInstance { header } { wallpaper }
362     \hfil
363   }
364   \cs_set_nopar:Npn \@oddfoot
365   {
366     \UseInstance { footer } { wallpaper }
367     \hfil
368   }
369   \cs_set_eq:NN \@evenhead \@oddhead
370   \cs_set_eq:NN \@evenfoot \@oddfoot
371 }
372 \cs_new_nopar:Npn \ps@talk
373 {
374   \cs_set_nopar:Npn \@oddhead
375   {
376     \UseInstance { header } { std }
377     \hfil
378   }
379   \cs_set_nopar:Npn \@oddfoot { \UseInstance { footer } { std } }
380   \cs_set_eq:NN \@evenhead \@oddhead
381   \cs_set_eq:NN \@evenfoot \@oddfoot
382 }
383 \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.
384 \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).
385 \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).
386 \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 LATEX counter-like access.
\c@frame 387 \int_new:N \g__talk_frame_int
\theframe 388 \cs_new_eq:NN \c@frame \g__talk_frame_int
\@framenum 389 \cs_new:Npn \theframe { \@arabic \c@frame }
390 \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.

```
391 \property_new:nnnn { totalframes } { shipout } { -1 }
392   { \int_use:N \g__talk_frame_int }
393 \AddToHook { enddocument / afterlastpage }
394   { \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.

```
395 \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.

```
396 \cs_new_protected:Npn \__talk_frame_process:nn #1#2
397   {
398     \int_gincr:N \g__talk_frame_int
399     \bool_set_true:N \l__talk_frame_bool
400     \__talk_slide:nn {#1} {#2}
401   }
```

(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.

```
402 \cs_new_protected:Npn \__talk_frame_tag:n #1
403   {
404     \tag_struct_begin:n { tag = frame }
405     \int_gset:Nn \g__talk_frame_struct_int { \tag_get:n { struct_num } }
406     \bool_gset_true:N \g__talk_frame_tag_bool
407     #1
408     \tag_struct_end:
409   }
```

(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.

```
410 \cs_new_protected:Npn \__talk_frame_notag:n #1
411   {
412     \tag_mc_begin:n { artifact }
413     \tag_suspend:n { frame }
414     \bool_gset_false:N \g__talk_frame_tag_bool
415     #1
416     \par
417     \tag_resume:n { frame }
418     \tag_mc_end:
419   }
```

(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.

```

420 \bool_if:NTF \l__talk_frame_title_bool
421 {
422   \RenewDocumentEnvironment { frame }
423     { D <> { all } = { action-spec } 0 { } +m +b }
424     {
425       \keys_set:nn { talk / frame } {#2}
426       \bool_set_false:N \l__talk_frame_verb_bool
427       \__talk_frame_process:nn {#1} { \frametitle {#3} #4 }
428     }
429     { }
430   \NewDocumentEnvironment { frame* }
431     { D <> { all } = { action-spec } 0 { } +m c }
432     {
433       \keys_set:nn { talk / frame } {#2}
434       \bool_set_true:N \l__talk_frame_verb_bool
435       \tl_gset:Nn \g__talk_frame_title_tl {#3}
436       \exp_args:Nne \__talk_frame_process:nn {#1}
437         { \tl_to_str:n { \frametitle } \exp_not:n { {#3} #4 } }
438     }
439     { }
440   }
441   {
442     \RenewDocumentEnvironment { frame }
443       { !D <> { all } = { action-spec } !0 { } +b }
444       {
445         \keys_set:nn { talk / frame } {#2}
446         \bool_set_false:N \l__talk_frame_verb_bool
447         \__talk_frame_process:nn {#1} {#3}
448       }
449       { }
450     \NewDocumentEnvironment { frame* }
451       { !D <> { all } = { action-spec } !0 { } c }
452       {
453         \keys_set:nn { talk / frame } {#2}
454         \bool_set_true:N \l__talk_frame_verb_bool
455         \__talk_frame_process:nn {#1} {#3}
456       }
457       { }
458   }

```

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

459 </class>

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. the action is applied inside the box after starting horizontal mode to avoid spacing issues when switching whatsits in and out.

```

69 \NewDocumentEnvironment { column } { D <> { all } 0 { } m }

```

```

70 {
71   \par
72   \keys_set:nn { talk / column } {#2}
73   \vbox_set_to_wd:Nnw \l__talk_tmp_box {#3}
74   \dim_set:Nn \textwidth {#3}
75   \dim_set_eq:NN \columnwidth \textwidth
76   \@parboxrestore
77   \leavevmode
78   \raggedright
79   \__talk_action_begin:n {#1}
80   \ignorespaces
81 }

```

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

```

82 {
83   \__talk_action_end:
84   \vbox_set_end:
85   \use:c { __talk_column_align_ \l__talk_column_alignment_tl :n }
86   { \vbox_unpack_drop:N \l__talk_tmp_box }
87   \hfil
88   \par
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 \@capttype {#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} }
\tablename 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 ??.)

1.3 Footnotes

`\@makefntext` A copy of the version provided by article: as for `\@caption`, we likely want a template here. It's not at present completely clear what will happen in the kernel (as the footnote templates currently leave `\@makefntext` alone).

```

153 \cs_new_protected:Npn \@makefntext #1
154 {
155   \parindent 1em
156   \noindent
157   \hb@xt@ 1.8em { \hss \@makefnmark }
158   #1
159 }

```

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

```

160 </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. At present, we need to create a whatsit here to avoid spacing issues.
(In LuaTEX, if we can move to attributes, this can be removed.)
```

```
17 \cs_new_protected:Npn __talk_action_:N #1 { \opacity_select: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` Simply hide unconditionally.

```

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

```

\__talk_action_only_end:N
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(enu)` 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

<pre> \textbf \textit \textmd \textnormal \textrm \textsc \textsf \textsl \texttt \textup \emph \stdtextbf \stdtextit \stdtextmd \stdtextnormal \stdtextrm \stdtextsc \stdtextsf \stdtextsl </pre>	<p>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.</p> <pre> 282 \tl_map_inline:nn 283 { 284 \textbf 285 \textit 286 \textmd 287 \textnormal </pre>
--	--

```

288 \textrm
289 \textsc
290 \textsf
291 \textsl
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
`\stdincludegraphics` the documented behavior of starred commands generally.

```

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
`__talk_label:n` be declared from scratch. There is also a non-standard overlay default. At present, no

special tricks as seen in beamer.

```
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: it 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         \UseTaggingSocket { sec / end } { \use:c { toplevel@ #2 } }
97         \UseTaggingSocket { sec / begin }
98         {
99           { \use:c { toplevel@ #2 } }
100          {
101            tag =
102            \exp_not:N \UseStructureName
103            { sec / \use:c { toplevel@ #2 } }
104          }
105        }
106        \tl_set:Nn \exp_not:c { l__talk_ #2 _tl } {##4}
107        \UseTaggingSocket { talk / sec / title } {#2}
108        \str_if_eq:nnT {#2} { section }
109          { \tl_clear:N \exp_not:N \l__talk_subsection_tl }
110        \str_if_eq:nnF {#2} { subsubsection }
111          { \tl_clear:N \exp_not:N \l__talk_subsubsection_tl }
112        \exp_not:N \addcontentsline { toc } {#2}

```

```

113         {
114             \exp_not:N \int_compare:nNnF {#1} >
115             { \exp_not:N \value { secnumdepth } }
116             {
117                 \exp_not:N \protect \exp_not:N \numberline
118                 { \exp_not:c { the #2 } }
119             }
120             ##4
121         }
122         \hook_use:n { #2 / begin }
123     }
124     \hook_new:n { #2 / begin }
125 }
126 }

```

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

```

talk/sec/title The argument is one of section, subsection or subsubsection.
__talk_sect_tag:nn
127 \NewTaggingSocket { talk / sec / title } { 1 }
128 \NewTaggingSocketPlug { talk / sec / title } { default }
129 { \exp_args:Ne __talk_sect_tag:nn { \text_purify:v { l__talk_ #1 _ t1 } } {#1} }
130 \cs_new_protected:Npn __talk_sect_tag:nn #1#2
131 {
132     \tag_struct_begin:e
133     {
134         tag =
135         \UseStructureName { sec / \use:c { toplevel@ #2 } / title } ,
136         title = {#1} ,
137         actualtext = {#1} ,
138     }
139     \tag_struct_end:
140 }
141 \AssignTaggingSocketPlug { talk / sec / title } { default }

```

(End of definition for `talk/sec/title` and `__talk_sect_tag:nn`. This function is documented on page ??.)

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 L^AT_EX 2_ε code as much as possible.

```

142 \cs_gset_protected:Npn \@starttoc #1
143 {
144     \begingroup
145     \makeatletter
146     \UseTaggingSocket { toc / starttoc / before } {#1}
147     \@input { \jobname .#1 }
148     \UseTaggingSocket { toc / starttoc / after } {#1}
149     \legacy_if:nT { @filesw }
150     {
151         \AddToHook { enddocument / afterlastpage }
152         {

```

```

153             \expandafter \newwrite \csname tf@ #1 \endcsname
154             \immediate \openout \csname tf@ #1 \endcsname \jobname .#1 \relax
155         }
156     }
157     \@nobreakfalse
158 \endgroup
159 }

```

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

\tableofcontents For the present simply print the output.

```

160 \NewDocumentCommand \tableofcontents { 0 { } }
161 {
162     \group_begin:
163     \@starttoc { toc }
164     \group_end:
165 }

```

(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@subsection
166 \cs_new_protected:Npn \l@section #1#2
167 { \__talk_toc_aux:nnnn { 1 } { \bfseries \color { structure } } {#1} {#2} }
168 \cs_new_protected:Npn \l@subsection #1#2
169 {
170     \__talk_toc_aux:nnnn
171     { 2 }
172     {
173         \skip_set:Nn \leftskip { 2em }
174         \color { . }
175     }
176     {#1} {#2}
177 }
178 \cs_new_protected:Npn \l@subsubsection #1#2
179 {
180     \__talk_toc_aux:nnnn
181     { 3 }
182     {
183         \skip_set:Nn \leftskip { 4em }
184         \color { . }
185         \footnotesize
186     }
187     {#1} {#2}
188 }
189 \cs_new_protected:Npn \__talk_toc_aux:nnnn #1#2#3#4
190 {
191     \int_compare:nNnTF { \value { section } } < 1
192     { \use:n }
193     { \__talk_toc_dest:n }
194     { \__talk_toc_level:nnnn {#1} {#2} {#3} {#4} }
195 }

```

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.

```

196 \cs_new_protected:Npn \__talk_toc_dest:n
197 {
198   \exp_after:wN \__talk_toc_dest:w \@contentsline@destination
199   . 0 . 0 . 0 . \q_stop
200 }
201 \cs_new_protected:Npn \__talk_toc_dest:w #1 . #2 . #3 . #4 . #5 \q_stop #6
202 {
203   \int_compare:nNnTF { \value { section } } = {#2}
204   {#6}
205   {
206     \group_begin:
207     \opacity_select:n { 0.2 }
208     #6
209     \group_end:
210   }
211 }
212 \cs_new_protected:Npn \__talk_toc_level:nmmn #1#2#3#4
213 {
214   \int_compare:nNnF {#1} > { \value { tocdepth } }
215   {
216     \group_begin:
217     \noindent
218     #2
219     \UseHookWithArguments { contentsline / text / before } { 4 }
220     {#1} {#3} {#4} { \@contentsline@destination }
221     #3
222     \UseHookWithArguments { contentsline / text / after } { 4 }
223     {#1} {#3} {#4} { \@contentsline@destination }
224     \UseHookWithArguments { contentsline / page / before } { 4 }
225     {#1} {#3} {#4}
226     { \@contentsline@destination }
227     \UseHookWithArguments { contentsline / page / after } { 4 }
228     {#1} {#3} {#4}
229     { \@contentsline@destination }
230     \par
231     \group_end:
232     \vfil
233   }
234 }

```

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

```

235 \setcounter { tocdepth } { 2 }

```

1.4 Block environments

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

`quote (env.)` `\NewDocumentEnvironment { description } { } { } { }`

`quotation (env.)` `\NewDocumentEnvironment { quote } { } { } { }`

`verse (env.)` `\NewDocumentEnvironment { quotation } { } { } { }`

`stdquote (env.)` `\NewDocumentEnvironment { verse } { } { } { }`

`stdquotation (env.)`

`stdverse (env.)`

```

240 \AddToHook { begindocument / before }
241 {
242   \clist_map_inline:nn { quote , quotation , verse }
243   {
244     \NewEnvironmentCopy { std #1 } {#1}
245     \RenewDocumentEnvironment {#1} { D <> { all } !0 { } }
246     {
247       \__talk_action_begin:n {##1}
248       \begin { std #1 } [ {##2} ]
249       \ignorespaces
250     }
251     {
252       \end { std #1 }
253       \__talk_action_end:
254     }
255   }
256 }

```

block (env.)

```

257 \NewDocumentEnvironment { block } { D <> { all } m }
258 {
259   \__talk_action_begin:n {#1}
260   \par
261   \vbox_set:Nw \l__talk_tmp_box
262   \group_begin:
263     \medskip
264     \leavevmode
265     \normalfont \large \bfseries
266     \color { structure }
267     #2
268     \par
269     \medskip
270   \group_end:
271 }
272 {
273   \vbox_set_end:
274   \box_use:N \l__talk_tmp_box
275   \par
276   \__talk_action_end:
277 }

```

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!

```

278 \AddToHook { begindocument / before }
279 {
280   \NewCommandCopy \stditem \item
281   \RenewDocumentCommand \item { d <> o }
282   {
283     \IfNoValueTF {#2}

```

```

284     { \stditem }
285     { \stditem [ {#2} ] }
286 \IfNoValueTF {#1}
287 {
288     \exp_after:wN \__talk_item_parse_spec:w
289     \l__talk_action_spec_str < all > \q_stop
290 }
291 { \__talk_item_parse_spec:n {#1} }
292 }
293 }

```

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.

```

294 \cs_new_protected:Npn \__talk_item_parse_spec:w #1 < #2 > #3 \q_stop
295 { \__talk_item_parse_spec:n {#2} }
296 \cs_new_protected:Npn \__talk_item_parse_spec:n #1
297 {
298     \tl_if_blank:nF {#1}
299     {
300         \tl_set:Nx \l__talk_list_end_tl
301         {
302             \exp_not:N \int_compare:nNnT \tex_currentgrouplevel:D =
303             { \int_use:N \tex_currentgrouplevel:D + 1 }
304             {
305                 \__talk_action_end:
306                 \tl_clear:N \exp_not:N \l__talk_list_end_tl
307             }
308         }
309         \__talk_action_begin:n {#1}
310     }
311 }

```

(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`

```

312 \tl_new:N \l__talk_list_end_tl

```

(End of definition for `\l__talk_list_end_tl`.)

`__block_inter_item:` 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.

`\endblockenv`

```

313 \cs_gset_protected:Npn \__block_inter_item:
314 {
315     \legacy_if:nT { @inlabel }
316     { \indent \par }
317 \mode_if_horizontal:T
318 {
319     \__block_skip_remove_last:
320     \__block_skip_remove_last:
321     \par
322 }

```

```

323 \l__talk_list_end_tl
324 \__kernel_list_item_end:
325 \__kernel_list_item_begin:
326 \addpenalty \@itempenalty
327 \addvspace \itemsep
328 }
329 \cs_gset:Npn \endblockenv
330 {
331   \__block_debug_typeout:n { blockenv-common~ending \on@line }
332   \bool_if:NT \l__block_level_incr_bool
333   { \int_gdecr:N \g_block_nesting_depth_int }
334   \legacy_if:nT { @inlabel }
335   {
336     \mode_leave_vertical:
337     \legacy_if_gset_false:n { @inlabel }
338   }
339   \__block_if_list:T
340   { \legacy_if:nT { @newlist } { \@noitemerr } }
341   \mode_if_horizontal:TF
342   {
343     \__block_skip_remove_last:
344     \__block_skip_remove_last:
345     \par
346   }
347   { \@inmatherr { \end { \@currenvir } } }
348   \l__talk_list_end_tl
349   \__kernel_displayblock_end:
350   \__block_if_list:T { \legacy_if_gset_false:n { @newlist } }
351   \legacy_if:nF { @nparlist }
352   {
353     \__block_skip_set_to_last:N \l_tmpa_skip
354     \dim_compare:nNnT \l_tmpa_skip > \c_zero_dim
355     {
356       \skip_vertical:n { - \l_tmpa_skip }
357       \skip_vertical:n { \l_tmpa_skip + \parskip - \@outerparskip }
358     }
359     \addpenalty \@endparpenalty
360     \addvspace \l__block_topsepadd_skip
361   }
362   \socket_use:n { block / endpe }
363 }

```

(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.) 364 \AddToHook { begindocument / before }
description (env.) 365 {
366   \clist_map_inline:nn { itemize , enumerate , description }
367   {
368     \RenewDocumentEnvironment {#1} { = { action-spec } !o }
369     {
370       \IfNoValueTF {##1}
371       { \UseInstance { blockenv } {#1} { } }

```



```

372         { \UseInstance { blockenv } {#1} {##1} }
373     }
374     { \endblockenv }
375 }
376 }

```

And add the structural color to item labels.

```

377 \AddToHook { begindocument / before }
378 {
379     \EditInstance { item } { basic }
380     { label-format = \color { structure } #1 }
381     \EditInstance { item } { description }
382     { label-format = \normalfont \bfseries \color { structure } #1 }
383 }

```

`\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.

```

384 \keys_define:nn { template / block / display }
385 { 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.

```

386 \NewCommandCopy \stdnewtheorem \newtheorem
387 \RenewDocumentCommand \newtheorem { m O {#1} m o }
388 {
389     \IfNoValueTF {#4}
390     { \stdnewtheorem {#1} [ {#2} ] {#3} }
391     { \stdnewtheorem {#1} [ {#2} ] {#3} [ {#4} ] }
392     \NewEnvironmentCopy { std #1 } {#1}
393     \RenewDocumentEnvironment {#1} { D <> { all } o }
394     {
395         \__talk_action_begin:n {##1}
396         \IfNoValueTF {##2}
397         { \begin { std #1 } }
398         { \begin { std #1 } [ {##2} ] }
399         \ignorespaces
400     }
401     {
402         \end { std #1 }
403         \__talk_action_end:
404     }
405 }

```

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

```

406 </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>
```

```
\@author We create a set of keys and variables in one go. Following the classical kernel approach,
\@date    all of the underlying storage is global. The short values will always be set in the following
\@institute code so can be used automatically anywhere we might want them.
\@subtitle 3 \clist_map_inline:nn
\@title    4 { author , date , institute , subtitle , title }
\@shortauthor 5 {
\@shortdate 6 \keys_define:nn { talk / metadata }
\@shortinstitute 7 {
\@shortsubtitle 8 #1 .tl_gset:c = @ #1 ,
\@shorttitle 9 short- #1 .tl_gset:c = @short #1
10 }
11 }
```

Allow empty values for author and title.

```
12 \tl_gclear:N \@author
13 \tl_gclear:N \@title
```

As the date has a standard value, that has to be propagated.

```
14 \tl_gset_eq:NN \@shortdate \@date
```

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

```
\author Slightly repetitive but as we need to handle the tagging aspects, this is easier than using
\date    a loop. The main aim is to add the short metadata concept. Notice that keys are set
\title   before the main data storage in case someone set the value as a key as well as a mandatory
         argument.
```

```
15 \RenewDocumentCommand \author { = { short-author } 0 {#2} m }
16 {
17   \keys_set:nn { talk / metadata } {#1}
18   \tl_gset:Nn \@author {#2}
19   \tl_gset_eq:NN \g__tag_title_author_tl \@author
20   \keys_set_known:nn { hyp } {#1}
21 }
22 \RenewDocumentCommand \date { = { short-date } 0 {#2} m }
23 {
24   \keys_set:nn { talk / metadata } {#1}
25   \tl_gset:Nn \@date {#2}
26 }
27 \RenewDocumentCommand \title { = { short-title } 0 {#2} m }
28 {
29   \keys_set:nn { talk / metadata } {#1}
```

```

30 \tl_gset:Nn \@title {#2}
31 \tl_gset_eq:NN \g__tag_title_title_tl \@title
32 \keys_set_known:nn { hyp } {#1}
33 }

```

(End of definition for \author, \date, and \title. These functions are documented on page ??.)

\institute Simple storage at present: unlike some of the kernel data, there is not a lot to do here.

```

\subtitle
34 \NewDocumentCommand \institute { = { short-institute } 0 {#2} m }
35 {
36   \keys_set:nn { talk / metadata } {#1}
37   \tl_gset:Nn \@institute {#2}
38 }
39 \NewDocumentCommand \subtitle { = { short-subtitle } 0 {#2} m }
40 {
41   \keys_set:nn { talk / metadata } {#1}
42   \tl_gset:Nn \@subtitle {#2}
43 }

```

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

\l__talk_titlelem_after_skip \l__talk_titlelem_before_skip \l__talk_titlelem_color_tl \l__talk_titlelem_font_tl \l__talk_titlelem_tag_begin_tl \l__talk_titlelem_tag_end_tl As the various elements of the titlepage share certain characteristics, we use a single template and split them as instances.

```

44 \NewTemplateType { titlepage-element } { 1 }
45 \DeclareTemplateInterface { titlepage-element } { talk } { 1 }
46 {
47   after-skip : length = 0em ,
48   before-skip : length = 0em ,
49   color : tokenlist = . ,
50   font : tokenlist = \normalfont ,
51   tag-begin : tokenlist = ,
52   tag-end : tokenlist =
53 }
54 \DeclareTemplateCode { titlepage-element } { talk } { 1 }
55 {
56   after-skip = \l__talk_titlelem_after_skip ,
57   before-skip = \l__talk_titlelem_before_skip ,
58   color = \l__talk_titlelem_color_tl ,
59   font = \l__talk_titlelem_font_tl ,
60   tag-begin = \l__talk_titlelem_tag_begin_tl ,
61   tag-end = \l__talk_titlelem_tag_end_tl
62 }
63 {
64   \tl_if_empty:nF {#1}
65   {
66     \vspace { \l__talk_titlelem_before_skip }
67     \group_begin:
68       \tl_if_empty:NF \l__talk_titlelem_color_tl
69       { \color_select:V \l__talk_titlelem_color_tl }
70       \l__talk_titlelem_font_tl
71       \l__talk_titlelem_tag_begin_tl
72       #1
73       \par
74       \l__talk_titlelem_tag_end_tl

```

```

75     \group_end:
76     \vspace { \l__talk_titlelem_after_skip }
77   }
78 }

```

Standard settings are taken from beamer with minor adjustments.

```

79 \DeclareInstance { titlepage-element } { author } { talk }
80   { before-skip = 1em }
81 \DeclareInstance { titlepage-element } { date } { talk }
82   { after-skip = 0.5em }
83 \DeclareInstance { titlepage-element } { institute } { talk }
84   { font = \scriptsize }
85 \DeclareInstance { titlepage-element } { subtitle } { talk }
86   { before-skip = 0.25em , color = structure }
87 \DeclareInstance { titlepage-element } { title } { talk }
88   {
89     color = structure ,
90     font = \Large ,
91     tag-begin = \tag_struct_begin:n { tag = Title } ,
92     tag-end = \tag_struct_end:
93   }

```

(End of definition for \l__talk_titlelem_after_skip and others.)

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.

```

\l__talk_titlepage_order_clist
\l__talk_titlepage_alignment_tl
\l__talk_titlepage_framestyle_tl
\l__talk_frame_alignment_tl
94 \NewTemplateType { titlepage } { 0 }
95 \DeclareTemplateInterface { titlepage } { talk } { 0 }
96   {
97     element-order : commalist =
98     {
99       title      ,
100       subtitle   ,
101       author     ,
102       institute  ,
103       date
104     } ,
105     framestyle : tokenlist = talk ,
106     horizontal-alignment : choice { left , center , right } = center ,
107     vertical-alignment : choice { bottom , center , stretch , top } = center
108   }
109 \DeclareTemplateCode { titlepage } { talk } { 0 }
110   {
111     element-order = \l__talk_titlepage_order_clist ,
112     framestyle = \l__talk_titlepage_framestyle_tl ,
113     horizontal-alignment =
114     {
115       left = \tl_set:Nn \l__talk_titlepage_alignment_tl { flushleft } ,
116       center = \tl_set:Nn \l__talk_titlepage_alignment_tl { center } ,
117       right = \tl_set:Nn \l__talk_titlepage_alignment_tl { flushright }
118     } ,
119     vertical-alignment =
120     {
121       bottom = \tl_set:Nn \l__talk_frame_alignment_tl { bottom } ,

```

```

122     center = \tl_set:Nn \l__talk_frame_alignment_tl { center } ,
123     stretch = \tl_set:Nn \l__talk_frame_alignment_tl { stretch } ,
124     top = \tl_set:Nn \l__talk_frame_alignment_tl { top }
125   }
126 }
127 {
128   \tl_if_empty:NF \l__talk_titlepage_framestyle_tl
129   { \exp_args:NV \thispagestyle \l__talk_titlepage_framestyle_tl }
130   \begin { \l__talk_titlepage_alignment_tl }
131     \cs_set_protected:Npn \and { \quad }
132     \clist_map_inline:Nn \l__talk_titlepage_order_clist
133     {
134       \ExpandArgs { nnv } \UseInstance { titlepage-element }
135       {##1} { @ ##1 }
136     }
137   \end { \l__talk_titlepage_alignment_tl }
138 }

```

(End of definition for \l__talk_titlepage_order_clist and others.)

\maketitle A very simple setup.

```

139 \NewDocumentCommand \maketitle { 0 {} }
140 {
141   \bool_if:NTF \l__talk_frame_bool
142   { \UseTemplate { titlepage } { talk } {##1} }
143   {
144     \begin { frame }
145       \UseTemplate { titlepage } { talk } {##1}
146     \end { frame }
147   }
148 }

```

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

```

149 </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
 \\ 279

A

\abovecaptionskip 138, 140
 \action 102
 \actionenv (env.) 102
 \addcontentsline 112
 \addpenalty 326, 359
 \AddToHook 66, 143,
 151, 240, 258, 278, 323, 364, 377, 393
 \addtolength 48
 \addvspace 327, 360
 \alert 79
 \alertenv (env.) 89
 \alt 141
 \and 131
 \arabic 390
 \arraycolsep 25
 \arrayrulewidth 27
 \AssignTaggingSocketPlug 141
 \author 15

B

\begin ... 110, 111, 130, 144, 248, 397, 398
 \begingroup 144, 145, 341
 \belowcaptionskip 139, 141
 \bfseries 21, 39, 167, 265, 382
 \bigskipamount 18
 block (env.) 257
 block commands:
 \g_block_nesting_depth_int 333
 block internal commands:
 __block_debug_typeout:n 331
 __block_if_list:TF 339, 350
 __block_inter_item: 313, 313
 \l_block_level_incr_bool 332
 __block_list_end: 52
 __block_list_item_end: 52
 __block_skip_remove_last:
 319, 320, 343, 344
 __block_skip_set_to_last:N 353
 \l_block_topsepadd_skip 360
 bool commands:
 \bool_do_while:Nn 27
 \bool_gset_false:N ... 30, 36, 40, 414

\bool_gset_true:N . 203, 211, 217, 406
 \bool_if:NNTF 6,
 20, 26, 32, 38, 43, 44, 46, 55, 85,
 123, 125, 132, 141, 178, 249, 332, 420
 \bool_lazy_and:nnTF 39, 214, 267
 \bool_lazy_or:nnTF 5, 19
 \bool_new:N 3,
 3, 7, 8, 13, 99, 101, 384, 385, 386
 \bool_set_eq:NN 117, 121
 \bool_set_false:N
 25, 26, 91, 110, 123, 426, 446
 \bool_set_true:N 22, 27,
 42, 59, 141, 178, 198, 213, 399, 434, 454
 \c_false_bool 15
 \c_true_bool 14

box commands:

\box_dp:N 25
 \box_ht:N 60, 225, 250
 \box_move_down:nn 58
 \box_new:N 4, 56
 \box_use:N 274
 \box_use_drop:N
 24, 203, 230, 234, 236, 319
 \box_wd:N 30

box internal commands:

__box_dim_eval:n 22, 25, 30, 33
 __box_set_to_wd: 29, 34

C

\clearpage 92
 clist commands:
 \clist_const:Nn 48
 \clist_if_in:NnTF 55, 177
 \clist_map_break: 199, 218
 \clist_map_inline:Nn ... 132, 180, 306
 \clist_map_inline:nn
 3, 59, 79, 120, 242, 366
 \clist_new:N 10, 14
 \clist_pop:NNTF 302
 \clist_set:Nn 45, 176
 \color .. 4, 11, 167, 174, 184, 266, 380, 382
 color commands:
 \color_group_begin: 23, 35
 \color_group_end: 23
 \color_math:nn 9, 25
 \color_math:nnn 10, 26
 \color_select:n 7, 16,
 21, 22, 35, 36, 48, 69, 204, 247, 300, 341

`\colorselect:nn` 8, 17, 37
`\colorlet` 45
`column (env.)` 69
`columns (env.)` 9
`\columnwidth` 16, 75, 110
cs commands:
`\cs_generate_variant:Nn` 7, 8, 9, 10,
45, 46, 48, 49, 50, 51, 52, 53, 54, 175
`\cs_gset:Npn` 78, 79, 80, 329
`\cs_gset_eq:NN` 56
`\cs_gset_nopar:Npn` 49, 57, 64
`\cs_gset_protected:Npn`
..... 18, 27, 37, 142, 142, 274, 313
`\cs_if_exist:NTF` 67, 75, 108, 220
`\cs_if_exist_p:N` 268
`\cs_if_exist_use:NTF` 132, 134
`\cs_new:Npn` . 6, 7, 34, 35, 36, 37, 38,
39, 40, 41, 42, 144, 209, 329, 389, 390
`\cs_new_eq:NN` 5, 6, 141, 388
`\cs_new_nopar:Npn` 3, 372
`\cs_new_protected:Npe` 53, 68, 94
`\cs_new_protected:Npn` 9,
11, 16, 17, 18, 24, 30, 33, 35, 36, 41,
44, 50, 51, 52, 53, 55, 56, 67, 72, 81,
81, 94, 100, 101, 106, 112, 113, 115,
121, 129, 130, 130, 131, 136, 139,
144, 146, 149, 153, 157, 166, 168,
168, 168, 170, 174, 176, 178, 184,
189, 190, 196, 201, 212, 239, 265,
294, 296, 307, 335, 339, 396, 402, 410
`\cs_set:Npn` 14, 15, 59
`\cs_set_eq:NN`
..... 142, 354, 355, 369, 370, 380, 381
`\cs_set_nopar:Npn` 112,
347, 349, 353, 357, 359, 364, 374, 379
`\cs_set_protected:Npn`
..... 29, 98, 131, 142, 194, 206
`\cs_to_str:N` 297, 302
`\csname` 149, 153, 154

D

`\date` 15
`\day` 20
`\DeclareColor` 42, 48, 49, 50
`\DeclareInstance`
... 43, 52, 79, 81, 83, 85, 87, 114,
211, 212, 213, 214, 215, 216, 257, 322
`\DeclareInstanceCopy` 260, 325
`\DeclareTemplateCode`
.... 23, 49, 54, 98, 109, 191, 228, 275
`\DeclareTemplateInterface`
.... 16, 45, 47, 93, 95, 184, 218, 265
`\definecolor` 46
`description (env.)` 236, 364

dim commands:

`\dim_compare:nNnTF` 49, 223, 354
`\dim_compare_p:nNn` 50, 270
`\dim_const:Nn` 95, 101
`\dim_eval:n` 40, 41, 42
`\dim_max:nn` 51, 249
`\dim_set:Nn` 15, 74
`\dim_set_eq:NN` 16, 75
`\dim_to_decimal:n` 88
`\dim_use:N` 116, 117
`\c_zero_dim` 354
`\DocumentMetadata` 6
`\doublerulesep` 28

E

`\EditInstance` 261, 326, 379, 381
`\emph` 282
`\end` 117, 118, 137, 146, 252, 347, 402
`\endblockenv` 313, 374
`\endcsname` 149, 153, 154
`\endfloatenv` 115, 128
`\endgroup` 151, 158, 354
`enumerate (env.)` 364
environments:
`actionenv` 102
`alertenv` 89
`block` 257
`column` 69
`columns` 9
`description` 236, 364
`enumerate` 364
`figure` 120
`invisibleenv` 71
`itemize` 364
`onlyenv` 71
`overlayarea` 197
`overprint` 214
`quotation` 236
`quote` 236
`stdquotation` 236
`stdquote` 236
`stdverse` 236
`table` 120
`uncoverenv` 71
`verse` 236
`visibleenv` 89

exp commands:

`\exp_after:wN` 198, 288
`\exp_args:Ne` 44, 129, 188
`\exp_args:Nne` 436
`\exp_args:No` 30
`\exp_args:NV` 129
`\exp_args_generate:n` 47

<code>\exp_not:N</code>	55, 57, 58, 59, 62, 63, 65, 68, 68, 71, 73, 74, 76, 76, 79, 85, 87, 89, 90, 91, 92, 93, 95, 96, 97, 98, 99, 101, 101, 102, 102, 103, 104, 106, 107, 108, 109, 111, 112, 112, 114, 115, 117, 118, 136, 256, 268, 301, 302, 302, 303, 306, 324
<code>\exp_not:n</code>	258, 326, 327, 437
<code>\exp_stop_f:</code>	40, 41, 42
<code>\expandafter</code>	153
<code>\ExpandArgs</code>	61, 71, 81, 89, 134, 251, 297, 298, 304, 311
F	
<code>\fboxrule</code>	32
<code>\fboxsep</code>	31
<code>\fi</code>	18, 147
<code>figure (env.)</code>	120
<code>\figurename</code>	131
file commands:	
<code>\file_if_exist_input:nTF</code>	88
<code>\file_input:n</code>	90
<code>\footins</code>	30
<code>\footnotesize</code>	185
<code>\footskip</code>	289, 290
fp commands:	
<code>\fp_eval:n</code>	93
<code>\fp_to_dim:n</code>	103
<code>\frame</code>	25, 26, 395
<code>frame*</code>	420
<code>\framesubtitle</code>	10
<code>\frametitle</code>	5, 427, 437
G	
<code>\gdef</code>	256
<code>\geometry</code>	3
group commands:	
<code>\group_begin:</code>	11, 33, 34, 58, 60, 64, 67, 84, 104, 115, 162, 202, 206, 216, 245, 262, 340
<code>\c_group_begin_token</code>	32
<code>\group_end:</code>	39, 40, 54, 62, 63, 69, 75, 87, 108, 139, 164, 207, 209, 231, 255, 270, 344
<code>\group_insert_after:N</code>	34, 159, 161
H	
hbox commands:	
<code>\hbox:n</code>	56
<code>\hbox_set_end:</code>	23
<code>\hbox_set_to_wd:Nnw</code>	14
<code>\headsep</code>	223, 242, 243
<code>\hfil</code>	17, 22, 87, 273, 317, 351, 362, 367, 377
hook commands:	
<code>\hook_gput_code:nnn</code>	46, 92, 262
<code>\hook_new:n</code>	124
<code>\hook_use:n</code>	122
<code>\hspace</code>	201, 208
<code>\hss</code>	157
<code>\hypersetup</code>	146
I	
<code>\IfBooleanT</code>	325
<code>\ifcase</code>	5
<code>\IfFormatAtLeastF</code>	7
<code>\IfNoValueF</code>	326, 327
<code>\IfNoValueTF</code>	15, 24, 35, 44, 185, 283, 286, 370, 389, 396
<code>\ignorespaces</code>	18, 80, 149, 195, 249, 399
<code>\immediate</code>	154
<code>\includegraphics</code>	316
<code>\indent</code>	316
<code>\insertsection</code>	81
<code>\insertsubsection</code>	81
<code>\insertsubsubsection</code>	81
<code>\institute</code>	34
int commands:	
<code>\int_compare:nNnTF</code>	114, 191, 196, 202, 203, 208, 210, 214, 302
<code>\int_compare_p:nNn</code>	215, 216
<code>\int_eval:n</code>	188
<code>\int_gdecr:N</code>	333
<code>\int_gincr:N</code>	29, 126, 186, 398
<code>\int_gset:Nn</code>	187, 405
<code>\int_gset_eq:NN</code>	129, 134, 139
<code>\int_gzero:N</code>	25, 74
<code>\int_incr:N</code>	218
<code>\int_max:nn</code>	171
<code>\int_new:N</code>	4, 5, 71, 128, 145, 208, 387
<code>\int_to_Roman:n</code>	211
<code>\int_to_roman:n</code>	212
<code>\int_use:N</code>	8, 14, 52, 56, 68, 303, 392
<code>\c_max_int</code>	190, 216
<code>\invisible</code>	59
<code>invisibleenv (env.)</code>	71
iow commands:	
<code>\iow_now:Nn</code>	254
<code>\item</code>	54, 278
<code>itemize (env.)</code>	364
<code>\itemsep</code>	54, 62, 69, 327
J	
<code>\jobname</code>	147, 154
K	
kernel internal commands:	
<code>__kernel_displayblock_end:</code>	349

prg commands:	seq commands:
\prg_generate_conditional_-	\seq_gput_right:Nn 146
variant:Nnn 10	\seq_gremove_all:Nn 75, 76, 77
\prg_new_protected_conditional:Npnn	\seq_map_indexed_inline:Nn 83
..... 3, 3	\seq_map_inline:Nn 126, 133, 138
\prg_return_false: 8, 9	\seq_new:N 118
\prg_return_true: 7, 8	\seq_set_from_clist:Nn 81, 119
\ProcessedArgument 14, 15	\l_tmpa_seq 81, 83
\ProcessKeyOptions 87	\setcounter 24, 235
prop commands:	\setlength 25, 26, 27, 28, 29, 31,
\prop_gput:Nnn 5, 6	32, 33, 43, 44, 45, 46, 47, 71, 140, 141
property commands:	\setmathfont 136
\property_new:nnnn 8, 391	\setsansfont 135
\property_record:nn 52, 68, 394	\SetTemplateKeys 109
\property_ref:nn 14	\sfdefault 142
\protect 117	\skip 30
\ProvidesExplClass 3	skip commands:
	\skip_horizontal:n 244, 291, 320, 337, 343
Q	\skip_set:Nn 173, 183
\quad 131	\skip_vertical:n 96, 98,
quark commands:	102, 104, 108, 110, 114, 116, 356, 357
\quark_if_recursion_tail_stop:N 131	\l_tmpa_skip 353, 354, 356, 357
\quark_if_recursion_tail_stop_-	socket commands:
do:Nn 148, 159	\socket_use:n 362
\quark_if_recursion_tail_stop_-	\space 19, 21
do:nn 37	\stdcolor 4
\q_recursion_stop 34, 124, 151	\stdemph 282
\q_recursion_tail 34, 124, 151	\stdincludegraphics 316
\q_stop 63, 71, 97, 99,	\stditem 280, 284, 285
102, 108, 181, 184, 199, 201, 289, 294	\stdmathcolor 4
quotation (env.) 236	\stdnewtheorem 386
quote (env.) 236	stdquotation (env.) 236
	stdquote (env.) 236
R	\stdtextbf 282
\raggedright 78, 129, 194	\stdtextcolor 4
\refstepcounter 95	\stdtextit 282
\relax 154	\stdtextmd 282
\relsize 93	\stdtextnormal 282
\RenewCommandCopy 26	\stdtextrm 282
\RenewDocumentCommand 11, 15, 20,	\stdtextsc 282
22, 27, 29, 94, 281, 298, 318, 332, 387	\stdtextsf 282
\RenewDocumentEnvironment 245, 368, 393, 422, 442	\stdtextsl 282
\RequirePackage 3, 91,	\stdtexttt 282
112, 130, 133, 134, 139, 140, 145, 316	\stdtextup 282
\rmdefault 142	stdverse (env.) 236
rule commands:	\stepcounter 18
\rule:nnn 37, 342	str commands:
	\str_clear:N 18, 28, 29
S	\str_if_empty:NTF 86
scan commands:	\str_if_empty_p:N 40
\scan_stop: 43	\str_if_eq:nnTF 17, 57, 83, 108, 110, 331
\scriptsize 84	\str_if_eq_p:nn 6, 7, 21
\section 72, 81	\str_new:N 9, 11, 12, 15, 100

\str_put_right:Nn	134, 170	__talk_decode_action:n	85, 94, 94
\str_replace_all:Nnn	20, 22, 52	__talk_decode_action:w	94, 96, 101
\str_set:Nn	18, 24, 67, 71, 111	\l__talk_decode_action_str	12, 18, 111, 120, 125
\str_set_eq:NN	119	\l__talk_decode_actions_bool	13, 25, 122, 126
\string	345	\l__talk_decode_actions_clist	13
\subsection	72, 81	\l__talk_decode_actions_str	13, 29
\subsubsection	72, 81	\l__talk_decode_arg_str	9, 24, 30, 117, 162
\subtitle	34	__talk_decode_check:n	127, 174, 174
sys commands:		__talk_decode_check:nw	174, 181, 184
\sys_if_engine_opentype:TF	131	__talk_decode_check_range:nnn	174, 190, 191, 206
T			
\tabbingsep	29	__talk_decode_check_single:nn	174, 187, 194
\tabcolsep	26	__talk_decode_mode:n	44, 53, 53
table (env.)	120	__talk_decode_mode:nn	76, 79, 81
\tableename	131	__talk_decode_mode:w	53, 62, 68
\tableofcontents	160	__talk_decode_mode_aux:n	53
tag commands:		__talk_decode_overlay.:nw	121
\tag_get:n	405	__talk_decode_overlay_aux:nNN	121, 142, 145, 146
\tag_mc_begin:n	173, 180, 412	__talk_decode_overlay_offset:nNn	121, 150, 155, 165, 168
\tag_mc_end:	171, 178, 418	__talk_decode_overlay_offset:nNnN	121, 154, 157, 166
\tag_resume:n	170, 417	__talk_decode_overlays:nN	121, 124, 129, 135, 172
\tag_struct_begin:n	91, 132, 172, 404	__talk_decode_overlays:nn	87, 106, 113, 121, 121
\tag_struct_end:	92, 139, 179, 408	\l__talk_decode_overlays_bool	3, 6, 22, 26, 42, 59, 118, 123
\tag_suspend:n	181, 413	\l__talk_decode_overlays_clist	10
tag internal commands:		\l__talk_decode_overlays_str	10, 28, 40, 86
\g__tag_title_author_tl	19	__talk_decode_parse:n	5, 16, 16, 116
\g__tag_title_title_tl	31	__talk_decode_parse:w	16, 34, 35, 46
\tagpdfparaOff	61	__talk_decode_parse_aux:n	16, 30, 33
\tagpdfsetup	147	\l__talk_decode_pure_bool	7, 27, 41, 91, 110
talk internal commands:		\l__talk_decode_step_bool	8, 123, 125, 141
__talk_action:N	17, 17	\l__talk_float_alignment_tl	91, 103, 104, 105, 111, 117
__talk_action_alert:N	18, 18	\l__talk_fontsize_dim	58, 88, 93
__talk_action_begin:n	11, 79, 102, 105, 111, 113, 124, 247, 259, 309, 395	\l__talk_footelem_color_tl	183
__talk_action_end:	26, 83, 102, 107, 112, 129, 130, 253, 276, 305, 403	\l__talk_footelem_font_tl	183
__talk_action_invisible:N	24, 24	\l__talk_footelem_left_skip	183
__talk_action_only:N	36	\l__talk_footelem_right_skip	183
__talk_action_only_begin:N	36	\l__talk_footer_bg_tl	264
__talk_action_only_end:N	36, 41	\l__talk_footer_fg_tl	264
\l__talk_action_spec_str	149, 289, 384	\l__talk_footer_font_tl	264
__talk_action_uncover:N	53, 53	\l__talk_footer_left_skip	264
__talk_action_visible:N	24, 30		
\l__talk_aspect_ratio_str	58, 107		
\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, 85		
\l__talk_columns_wd_tl	5, 14, 15		

<code>\l__talk_footer_order_clist</code>	264	<code>__talk_label:n</code>	332 , 336 , 339
<code>\l__talk_footer_right_skip</code>	264	<code>__talk_latex_frame:n</code> . .	26 , 395 , 395
<code>\l__talk_footer_sep_tl</code>	264	<code>\l__talk_list_end_tl</code>	
<code>\l__talk_frame_alignment_tl</code>			300 , 306 , 312 , 323 , 348
	89 , 94 , 148 , 158	<code>__talk_metadata_name:n</code>	
<code>\l__talk_frame_bool</code>	141 , 384 , 399		305 , 308 , 313 , 329 , 329
<code>\g__talk_frame_int</code>		<code>__talk_mode:n</code>	3
	14 , 52 , 68 , 211 , 387 , 392 , 398	<code>__talk_mode:nTF</code>	3 , 12
<code>__talk_frame_notag:n</code> . . .	41 , 410 , 410	<code>\l__talk_mode_str</code>	7 , 58 , 58 , 83
<code>__talk_frame_overprint:</code>		<code>\c__talk_modes_clist</code>	48 , 55
	209 , 209 , 221 , 224 , 228 ,	<code>__talk_onslide:n</code> .	147 , 148 , 149 , 188
	241 , 243 , 244 , 247 , 249 , 256 , 258 , 263	<code>\g__talk_onslide_escape_tl</code>	
<code>__talk_frame_process:nn</code>			156 , 159 , 161 , 165 , 171
	396 , 396 , 427 , 436 , 447 , 455	<code>__talk_onslide_reset:</code>	
<code>\g__talk_frame_struct_int</code> .	56 , 71 , 405		147 , 153 , 166 , 170
<code>\g__talk_frame_subtitle_tl</code> .	3 , 13 , 76	<code>\g__talk_onslide_tl</code>	
<code>__talk_frame_tag:n</code>	37 , 402 , 402		79 , 83 , 151 , 162 , 164 , 171
<code>\g__talk_frame_tag_bool</code>		<code>__talk_overlay_arg:n</code>	
	46 , 385 , 406 , 414		3 , 11 , 62 , 72 , 82 , 90
<code>\l__talk_frame_tagging_str</code>		<code>__talk_overprint_begin:n</code>	
	17 , 18 , 20 , 22 , 34 , 149		190 , 190 , 198 , 215
<code>__talk_frame_title:n</code>	15 , 38 , 44	<code>__talk_overprint_check_ht:n</code> . . .	
<code>\l__talk_frame_title_bool</code> . .	58 , 420		214 , 263 , 265 , 274
<code>__talk_frame_title_tagged:n</code> . . .		<code>\l__talk_overprint_int</code> .	208 , 212 , 218
	15 , 47 , 51	<code>__talk_overprint_save_ht:</code>	
<code>\g__talk_frame_title_tl</code>			214 , 219 , 239
	3 , 8 , 58 , 75 , 253 , 435	<code>\c__talk_paper_height_dim</code>	95
<code>\l__talk_frame_verb_bool</code>		<code>\c__talk_paper_width_dim</code>	95
	44 , 386 , 426 , 434 , 446 , 454	<code>\g__talk_pauses_int</code>	
<code>\l__talk_frametitle_after_skip</code> .			8 , 4 , 74 , 126 , 171 , 186 , 187 , 188
	25 , 41	<code>\l__talk_saved_action_str</code> 99 , 119 , 135	
<code>\l__talk_frametitle_before_skip</code>		<code>\l__talk_saved_actions_bool</code>	
	26 , 32		99 , 121 , 137
<code>\l__talk_frametitle_color_tl</code> . . .		<code>\l__talk_saved_overlays_bool</code>	
	27 , 34 , 35		99 , 117 , 132
<code>\l__talk_frametitle_font_tl</code> . .	28 , 36	<code>__talk_sect_tag:nn</code>	127 , 129 , 130
<code>\l__talk_header_bg_tl</code>	217	<code>\g__talk_section_tl</code>	66
<code>\l__talk_header_fg_tl</code>	217	<code>\l__talk_section_tl</code>	66
<code>\l__talk_header_font_tl</code>	217	<code>__talk_slide:nn</code>	9 , 9 , 400
<code>\l__talk_header_frametitle_bool</code>	217	<code>__talk_slide_align_bottom:n</code> .	94 , 94
<code>\l__talk_header_ht_dim</code>	217	<code>__talk_slide_align_center:n</code>	94 , 100
<code>\l__talk_header_left_skip</code>	217	<code>__talk_slide_align_stretch:n</code>	94 , 106
<code>\l__talk_header_right_skip</code>	217	<code>__talk_slide_align_top:n</code> . .	94 , 112
<code>__talk_header_tag_begin:n</code>		<code>__talk_slide_aux:n</code>	9 , 45 , 56
	53 , 168 , 168 , 175	<code>__talk_slide_begin:</code>	33 , 72 , 72
<code>__talk_header_tag_end:</code> .	64 , 168 , 176	<code>\l__talk_slide_box</code>	4 , 78 , 90
<code>__talk_if_overlay:n</code>	3 , 10	<code>\g__talk_slide_continue_bool</code> . .	3 ,
<code>__talk_if_overlay:nTF</code>			27 , 30 , 36 , 40 , 85 , 178 , 203 , 211 , 217
	3 , 7 , 12 , 13 , 13 , 22 , 31 ,	<code>__talk_slide_end:</code>	49 , 72 , 81
	31 , 34 , 96 , 143 , 152 , 175 , 301 , 320 , 335	<code>\g__talk_slide_int</code>	
<code>__talk_item_parse_spec:n</code>			5 , 8 , 25 , 29 , 196 , 202 , 208 , 210 , 215
	278 , 291 , 295 , 296	<code>\g__talk_subsection_tl</code>	66
<code>__talk_item_parse_spec:w</code>		<code>\l__talk_subsection_tl</code>	66 , 109
	278 , 288 , 294	<code>\g__talk_subsubsection_tl</code>	66

<code>\l__talk_subsubsection_tl</code> ..	66, 111	<code>\@listI</code>	56
<code>__talk_textcmd_equiv:n</code> .	282, 303, 307	<code>\@listi</code>	49, 56
<code>\l__talk_titlelem_after_skip</code>	44	<code>\@listii</code>	57
<code>\l__talk_titlelem_before_skip</code> ...	44	<code>\@listiii</code>	64
<code>\l__talk_titlelem_color_tl</code>	44	<code>\@makecaption</code>	149
<code>\l__talk_titlelem_font_tl</code>	44	<code>\@makefnmark</code>	157
<code>\l__talk_titlelem_tag_begin_tl</code> ..	44	<code>\@makefnstext</code>	31, 153
<code>\l__talk_titlelem_tag_end_tl</code>	44	<code>\@mpfootins</code>	30
<code>\l__talk_titlepage_alignment_tl</code> .	94	<code>\@nbreakfalse</code>	157
<code>\l__talk_titlepage_framestyle_tl</code> .	94	<code>\@noitemerr</code>	340
<code>\l__talk_titlepage_order_clist</code> ..	94	<code>\@oddfnfoot</code> .	353, 355, 364, 370, 379, 381
<code>__talk_tmp:w</code>	55, 55, 98, 107	<code>\@oddhead</code> .	349, 354, 359, 369, 374, 380
<code>\l__talk_tmp_box</code>	14,	<code>\@outerparskip</code>	357
	24, 39, 56, 60, 73, 86, 193, 203, 225,	<code>\@parboxrestore</code>	76, 146
	230, 234, 236, 250, 261, 274, 292, 319	<code>\@setminipage</code>	147
<code>\l__talk_tmp_tl</code>		<code>\@shortauthor</code>	3
	12, 18, 21, 23, 57, 100, 302, 304, 305	<code>\@shortdate</code>	3
<code>__talk_toc_aux:nnnn</code>		<code>\@shortinstitute</code>	3
	166, 167, 170, 180, 189	<code>\@shortsubtitle</code>	3
<code>__talk_toc_dest:n</code>	166, 193, 196	<code>\@shorttitle</code>	3
<code>__talk_toc_dest:w</code>	166, 198, 201	<code>\@starttoc</code>	142, 163
<code>__talk_toc_level:nnnn</code> .	166, 194, 212	<code>\@subtitle</code>	3, 42
<code>\l__talk_uncover_hidden_fp</code>	46	<code>\@title</code>	3, 30, 31
<code>__talk_wallpaper_hruler:Nnn</code>		<code>\c@figure</code>	131
	240, 287, 335, 335	<code>\c@frame</code>	387
<code>talk/sec/title</code>	127	<code>\c@page</code>	144
<code>\temporal</code>	173	<code>\c@pauses</code>	4
T _E X and L ^A T _E X 2 _ε commands:			
<code>\@arabic</code>	6, 7, 78, 79, 80, 144, 389	<code>\c@section</code>	78
<code>\@author</code>	3, 18, 19	<code>\c@slide</code>	5
<code>\@auxout</code>	254, 343	<code>\c@subsection</code>	79
<code>\@bsphack</code>	334	<code>\c@subsubsection</code>	80
<code>\@caption</code>	31, 142	<code>\c@table</code>	131
<code>\@capttype</code>	112	<code>\check@mathfonts</code>	143
<code>\@contentsline@destination</code>		<code>\currentgrouplevel</code>	52
	50, 198, 220, 223, 226, 229	<code>\fnum@figure</code>	131
<code>\@currentHref</code>	350	<code>\fnum@table</code>	131
<code>\@currentlabel</code>	347	<code>\Gm@bmargin</code>	290
<code>\@currentlabelname</code>	349	<code>\Gm@lmargin</code>	224, 271, 337
<code>\@currenvir</code>	347	<code>\Gm@rmargin</code>	226, 272, 320
<code>\@date</code>	3, 25	<code>\Gm@tmargin</code>	223
<code>\@definecounter</code>	141	<code>\hb@xt@</code>	157
<code>\@endparpenalty</code>	359	<code>\if@minipage</code>	147
<code>\@esphack</code>	337	<code>\ignorespaces</code>	29
<code>\@evenfoot</code>	355, 370, 381	<code>\l@section</code>	166
<code>\@evenhead</code>	354, 369, 380	<code>\l@subsection</code>	166
<code>\@framenumbers</code>	387	<code>\l@subsubsection</code>	166
<code>\@ignore</code>	29	<code>\on@line</code>	331
<code>\@ignoretrue</code>	89	<code>\protected@write</code>	343
<code>\@inmatherr</code>	347	<code>\ps@plain</code>	347
<code>\@input</code>	147	<code>\ps@talk</code>	347
<code>\@institute</code>	3, 37	<code>\ps@wallpaper</code>	347
<code>\@itempenalty</code>	326	<code>\std@definecounter</code>	141
<code>\@kernel@reserved@label@data</code> ...	351	tex commands:	
		<code>\tex_currentgrouplevel:D</code> ..	302, 303

