

ltx-talk – A class for typesetting presentations*

Joseph Wright[†]

Released 2026-02-10

Contents

I	ltx-talk – Overall set up	1
1	ltx-talk implementation	1
1.1	Set up	1
1.2	Additions for expl3	2
1.3	Extra variants	3
1.4	Scratch space	3
1.5	Option handling	4
1.6	Setting up	5
1.7	Math support	6
1.8	Font selection	6
1.9	Hyperlinks	6
1.10	Tagging	7
II	ltx-talk-color – Color definitions	8
1	ltx-talk-color implementation	8
1.1	Existing definitions	8
1.2	Document (and interface) commands	8
1.3	Color definition	10
1.4	Semantic colors	10
III	ltx-talk-decode – Decoding overlay specs	11
1	ltx-talk-decode implementation	11
IV	ltx-talk-frame – The structure of frames	18

*This file describes v0.4.4, last revised 2026-02-10.

[†]E-mail: joseph@texdev.net

1	ltx-talk-frame implementation	18
1.1	Slides in frames	18
1.2	Counters	21
1.3	Frame options	22
1.4	Tagging for headers	22
1.5	Wallpaper	23
1.6	The <code>frame</code> environment	27
V	ltx-talk-frame – The structure of frames	30
1	ltx-talk-frame-structure implementation	30
1.1	Columns	30
1.2	Floats	32
1.3	Footnotes	34
VI	ltx-talk-mode – Modes	36
1	ltx-talk-mode implementation	36
VII	ltx-talk-overlay – Overlays	37
1	ltx-talk-overlay implementation	37
1.1	Utilities	37
1.2	Opacity utilities	38
1.3	Action commands and environments	38
1.4	Non-action commands and environments	42
1.5	Fixed-size areas	43
1.6	Adding overlays to existing commands	45
VIII	ltx-talk-required – “Required” definitions	48
1	ltx-talk-required implementation	48
1.1	Standard design settings	48
1.2	List support	49
IX	ltx-talk-structure – Structural commands	50
1	ltx-talk-structure implementation	50
1.1	Frame title	50
1.2	Sectioning	51
1.3	Table of contents	53
1.4	Block environments	56
1.5	Lists	56
1.6	Theorems, <i>etc.</i>	60

X	ltx-talk-title – Title pages	61
1	ltx-talk-title implementation	61
	Index	65

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} {2026-02-10} {0.4.4}
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
    Warn if not an engine that is tested.
18 \bool_lazy_or:nnF
19   { \sys_if_engine luatex_p: }
20   { \sys_if_engine pdftex_p: }
21   {
22     \msg_new:nnn { ltx-talk } { unsupported-engine }
23     {
24       The~engine~"\c_sys_engine_str"~
25       is~not~supported~by~the~ltx-talk~class.
26     }
27     \msg_warning:nn { ltx-talk } { unsupported-engine }
28   }
```

1.2 Additions for expl3

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

```

29 \cs_gset_protected:Npn \vbox_set_to_wd:Nnn #1#2#3
30 {
31   \tex_setbox:D #1 \tex_vbox:D
32   {
33     \tex_hsize:D \__box_dim_eval:n {#2}
34     \color_group_begin: #3 \par \color_group_end:
35   }
36   \box_dp:N #1 \__box_dim_eval:n {#2}
37 }
38 \cs_gset_protected:Npn \vbox_set_to_wd:Nnw #1#2
39 {
40   \cs_set_protected:Npn \__box_set_to_wd:
41     { \box_wd:N #1 \__box_dim_eval:n {#2} }
42   \tex_setbox:D #1 \tex_vbox:D
43   \c_group_begin_token
44     \tex_hsize:D \__box_dim_eval:n {#2}
45     \group_insert_after:N \__box_set_to_wd:
46     \color_group_begin:
47 }

```

Some things from `xbox` that would be useful.

```

48 \cs_gset_protected:Npn \rule:nnn #1#2#3
49 {
50   \tex_vrule:D
51     height \dim_eval:n {#2} \exp_stop_f:
52     depth \dim_eval:n {#3} \exp_stop_f:
53     width \dim_eval:n {#1} \exp_stop_f:
54     \scan_stop:
55 }

```

Some extensions are needed to opacity support: this should only be here for a short period.

```

56 \cs_gset_protected:Npn \opacity_begin:n #1
57 { \__opacity_select:nN {#1} \__opacity_backend_begin:n }
58 \cs_gset_protected:Npn \opacity_end:
59 { \__opacity_backend_end: }
60 \AddToHook { begindocument }
61 {
62   \cs_gset_protected:Npe \__opacity_backend_begin:n #1
63   {
64     \bool_lazy_any:nTF
65       {
66         { \sys_if_engine_pdftex_p: }
67         { \sys_if_engine luatex_p: }
68         { \sys_if_engine_xetex_p: }
69       }
70       {
71         \tl_set:Nn \exp_not:N \l__opacity_backend_fill_tl {#1}
72         \tl_set:Nn \exp_not:N \l__opacity_backend_stroke_tl {#1}
73         \pdfmanagement_add:nnn { Page / Resources / ExtGState }
74           { opacity #1 }
75         { << /ca ~ #1 /CA ~ #1 >> }

```

```

76         \sys_if_engine_xetex:TF
77         { \__kernel_backend_literal_pdf:n }
78         {
79             \__kernel_color_backend_stack_push:nn
80             \exp_not:N \c__opacity_backend_stack_int
81         }
82         { /opacity #1 ~ gs }
83     }
84     {
85         \__opacity_backend:nnn {#1} { fill } { ca }
86         \__opacity_backend:nnn {#1} { stroke } { ca }
87     }
88 }
89 \cs_gset_protected:Npe \__opacity_backend_end:
90 {
91     \bool_lazy_any:nTF
92     {
93         { \sys_if_engine_pdftex_p: }
94         { \sys_if_engine luatex_p: }
95         { \sys_if_engine_xetex_p: }
96     }
97     { \__opacity_backend_reset: }
98     {
99         \__opacity_backend_reset_fill:
100         \__opacity_backend_reset_stroke:
101     }
102 }
103 }

```

1.3 Extra variants

```

104 \cs_generate_variant:Nn \clist_set:Nn { cv }
105 \cs_generate_variant:Nn \hook_gput_code:nnn { nne }
106 \exp_args_generate:n { nVv }
107 \cs_generate_variant:Nn \color_select:n { V }
108 \cs_generate_variant:Nn \dim_compare:nNnTF { v }
109 \cs_generate_variant:Nn \dim_compare_p:nNn { vNv }
110 \cs_generate_variant:Nn \dim_max:nn { v }
111 \cs_generate_variant:Nn \str_replace_all:Nnn { NnV }
112 \cs_generate_variant:Nn \text_purify:n { v }
113 \cs_generate_variant:Nn \vbox_to_ht:nn { v }

```

1.4 Scratch space

__talk_tmp:w For one-off processing.

```

114 \cs_new_protected:Npn \__talk_tmp:w { }

```

(End of definition for __talk_tmp:w.)

\l__talk_tmp_box

```

115 \box_new:N \l__talk_tmp_box

```

(End of definition for \l__talk_tmp_box.)

\l__talk_tmp_tl

116 \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

```
117 \keys_define:nn { talk }
118 {
119   aspect-ratio .str_set:N =
120     \l__talk_aspect_ratio_str ,
121   font-size .dim_set:N =
122     \l__talk_fontsize_dim ,
123   frame-title-arg .bool_set:N =
124     \l__talk_frame_title_bool ,
125   handout .code:n =
126     { \str_set:Nn \l__talk_mode_str { handout } } ,
127   handout .value_forbidden:n = true ,
128   mode .choices:nn =
129     { handout , projector }
130     { \str_set:NV \l__talk_mode_str \l_keys_choice_tl }
131 }
```

(End of definition for \l__talk_aspect_ratio_str and others.)

Scope for options.

```
132 \keys_define:nn { talk }
133 {
134   aspect-ratio .usage:n = load ,
135   font-size .usage:n = load ,
136   frame-title-arg .usage:n = load ,
137   mode .usage:n = load
138 }
```

Compatibility keys for classical font size setting.

```
139 \clist_map_inline:nn { 10pt , 11pt , 12pt }
140 {
141   \keys_define:nn { talk }
142   {
143     #1 .meta:n = { font-size = #1 } ,
144     #1 .value_forbidden:n = true ,
145     #1 .usage:n = load
146   }
147 }
```

Initial values.

```
148 \keys_set:nn { talk }
149 {
150   aspect-ratio = 16:9 ,
151   font-size = 11pt ,
152   frame-title-arg = false ,
153   mode = projector
154 }
155 \ProcessKeyOptions [ talk ]
```

1.6 Setting up

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

```

156 \file_if_exist_input:nF { size \dim_to_decimal:n \l__talk_fontsize_dim .clo }
157 {
158   \file_input:n { size10.clo }
159   \RequirePackage { relsize }
160   \hook_gput_code:nne { begindocument } { talk }
161   { \exp_not:N \relsize { \fp_eval:n { \l__talk_fontsize_dim / 10pt } } } }
162 }

```

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`.

```

163 \dim_const:Nn \c__talk_paper_height_dim { 100mm }
164 \use:e
165 {
166   \cs_set_protected:Npn \exp_not:N \__talk_tmp:w
167     #1 \tl_to_str:n { : } #2 \tl_to_str:n { : } #3 \exp_not:N \q_stop
168   {
169     \dim_const:Nn \exp_not:N \c__talk_paper_width_dim
170     {
171       \exp_not:N \fp_to_dim:n
172       { (#1 / #2) * \exp_not:N \c__talk_paper_height_dim }
173     }
174   }
175   \exp_not:N \__talk_tmp:w \l__talk_aspect_ratio_str
176   \tl_to_str:n { : } 100 \exp_not:N \q_stop
177 }
178 \use:e
179 {
180   \exp_not:N \RequirePackage
181   [
182     papersize =
183     {
184       \dim_use:N \c__talk_paper_width_dim ,
185       \dim_use:N \c__talk_paper_height_dim
186     } ,
187     tmargin    = 10mm ,
188     bmargin    = 8mm ,
189     lmargin    = 10mm ,
190     rmargint   = 10mm ,
191     headheight = 10mm ,
192     headsep    = 2mm ,
193     footskip   = 6mm
194   ]
195   { geometry }
196 }

```

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

Turn off justification

```

197 \raggedright

```


1.7 Math support

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

```
198 \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 `(lua-)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.

```
199 \sys_if_engine_opentype:TF
200 {
201   \RequirePackage { fontspec }
202   \RequirePackage { mathtools }
203   \sys_if_engine luatex:TF
204   {
205     \RequirePackage { lua-unicode-math }
206     \tagpdfsetup { math / mathml / luamml / load = true }
207   }
208   { \RequirePackage { unicode-math } }
209   \setmainfont { NewCMSans10-Regular.otf }
210   \setsansfont { NewCMSans10-Regular.otf }
211   \setmathfont { NewCMSansMath-Regular.otf }
212 }
213 {
214   \RequirePackage { sansmathfonts }
215   \RequirePackage [ nomath ] { lmodern }
216   \cs_set_eq:NN \rmdefault \sfdefault
217 }
```

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. This is needed to set up math dimensions for vertical centering.

```
218 \AddToHook { begindocument / end } { \check@mathfonts }
```

1.9 Hyperlinks

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

```
219 \cs_new:Npn \thepage { \@arabic \c@page }
```

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

A requirement.

```
220 \RequirePackage { hyperref }
```

```
221 \hypersetup { hidelinks }
```

1.10 Tagging

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

```
222 \tagpdfsetup
223 {
224   role / user-NS = ltx-talk      ,
225   role / new-tag = frame / Sect  ,
226   role / new-tag = frametitle / H4
227 }
228 \</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 (and interface) 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   \ignorespaces
20 }
21 \RenewDocumentCommand \mathcolor { D <> { all } o m +m }
22 {
```

```

23   \_talk_if_overlay:nT {#1}
24   {
25       \IfNoValueTF {#2}
26       { \color_math:en {#3} {#4} }
27       { \color_math:nen {#2} {#3} {#4} }
28   }
29 }
30 \RenewDocumentCommand \textcolor { D <> { all } o m +m }
31 {
32   \_talk_if_overlay:nT {#1}
33   {
34       \mode_leave_vertical:
35       \group_begin:
36       \IfNoValueTF {#2}
37       { \color_select:e {#3} }
38       { \color_select:ne {#2} {#3} }
39       #4
40       \group_end:
41   }
42 }

```

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

`\pagecolor` Here, the definition is different: we directly use the shipout hook.
`_talk_pagecolor:n`

```

43 \RenewDocumentCommand \pagecolor { D <> { all } o m }
44 {
45   \_talk_if_overlay:nT {#1}
46   {
47       \IfNoValueTF {#2}
48       { \_talk_pagecolor:n { {#3} } }
49       { \_talk_pagecolor:n { [ {#2} ] {#3} } }
50   }
51 }
52 \cs_new_protected:Npn \_talk_pagecolor:n #1
53 {
54   \AddToHook { shipout / background }
55   {
56       \color #1
57       \put ( 0cm, -\paperheight )
58       { \rule { \paperwidth } { \paperheight } }
59   }
60 }

```

(End of definition for `\pagecolor` and `_talk_pagecolor:n`. This function is documented on page ??.)

`\stdset@color`
`\stdreset@color`

```

61 \cs_set_eq:NN \stdset@color \set@color
62 \cs_set_eq:NN \stdreset@color \reset@color

```

(End of definition for `\stdset@color` and `\stdreset@color`. These functions are documented on page ??.)

`\set@color` Part of code-level interface for color: simply use the expl3 version of the same idea.
`\reset@color`

```

63 \cs_set_eq:NN \set@color \color_ensure_current:
64 \cs_set_eq:NN \reset@color \_color_backend_reset:

```

(End of definition for \set@color and \reset@color. 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.

```
65 \NewDocumentCommand \DeclareColor { m o m }  
66 {  
67   \IfNoValueTF {#2}  
68     { \colorlet {#1} {#3} }  
69     { \definecolor {#1} {#2} {#3} }  
70 }
```

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

1.4 Semantic colors

Pick up the standard colors from beamer.

```
71 \DeclareColor { alert } [ RGB ] { 200 , 0 , 0 }  
72 \DeclareColor { example } { green!50!black }  
73 \DeclareColor { structure } [ rgb ] { 0.2 , 0.2 , 0.7 }  
74 </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` 13 `\bool_new:N \l__talk_decode_actions_bool`

`\l__talk_decode_actions_str` 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_auxi:n`

`__talk_decode_parse_auxii:n`

`__talk_decode_parse:w`

```

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

```

Stepping the value assigned to `+` is done in the outer loop, as within one overlay expression it always takes the same value. If the `amsmath \ifmeasuring@` flag is on, the overlay counter is not advanced.

```

35 \cs_new_protected:Npn \__talk_decode_parse_auxii:n #1
36 {
37   \bool_set_false:N \l__talk_decode_step_bool
38   \__talk_decode_parse:w #1 | \q_recursion_tail | \q_recursion_stop
39   \bool_if:NT \l__talk_decode_step_bool
40   {
41     \legacy_if:nF { measuring@ }
42     { \int_gincr:N \g__talk_pauses_int }
43   }
44 }

```

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

```

45 \cs_new_protected:Npn \__talk_decode_parse:w #1 |
46 {
47   \quark_if_recursion_tail_stop_do:nn {#1}
48   {
49     \bool_lazy_and:nnT
50     { \str_if_empty_p:N \l__talk_decode_overlays_str }
51     { ! \l__talk_decode_pure_bool }

```

```

52         { \bool_set_true:N \l__talk_decode_overlays_bool }
53     }
54     \exp_args:Ne \__talk_decode_mode:n
55         { \tl_trim_spaces:n {#1} }
56     \__talk_decode_parse:w
57 }

```

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

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

```

58 \clist_const:Ne \c__talk_modes_clist
59 {
60     \tl_to_str:n { handout } ,
61     \tl_to_str:n { projector }
62 }

```

(End of definition for \c__talk_modes_clist.)

__talk_decode_mode:n 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:w
\__talk_decode_mode_aux:n
63 \cs_new_protected:Npe \__talk_decode_mode:n #1
64 {
65     \clist_if_in:NnTF \exp_not:N \c__talk_modes_clist {#1}
66     {
67         \exp_not:N \str_if_eq:VnT
68         \exp_not:N \l__talk_mode_str {#1}
69         { \bool_set_true:N \exp_not:N \l__talk_decode_overlays_bool }
70     }
71     {
72         \exp_not:N \__talk_decode_mode:w #1 \tl_to_str:n { : : }
73         \exp_not:N \q_stop
74     }
75 }
76 \use:e
77 {
78     \cs_new_protected:Npe \exp_not:N \__talk_decode_mode:w
79     #1 \token_to_str:N :
80     #2 \token_to_str:N :
81     #3 \exp_not:N \q_stop
82 }
83 {
84     \exp_not:N \tl_if_blank:nTF {#2}
85     {
86         \exp_not:N \__talk_decode_mode:nn
87         { \tl_to_str:n { projector } } {#1}
88     }
89     { \exp_not:N \__talk_decode_mode:nn {#1} {#2} }
90 }
91 \cs_new_protected:Npn \__talk_decode_mode:nn #1#2
92 {
93     \str_if_eq:VnTF \l__talk_mode_str {#1}
94     {
95         \__talk_decode_action:n {#2}
96         \str_if_empty:NT \l__talk_decode_overlays_str

```



```

97         { \__talk_decode_overlays:nn { overlays } { * } }
98     }
99     {
100         \tl_if_blank:nF {#2}
101         { \bool_set_false:N \l__talk_decode_pure_bool }
102     }
103 }

```

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

__talk_decode_action: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.

__talk_decode_action:w

```

104 \cs_new_protected:Npe \__talk_decode_action:n #1
105 {
106     \exp_not:N \__talk_decode_action:w
107     #1 \tl_to_str:n { @ @ } \exp_not:N \q_stop
108 }
109 \use:e
110 {
111     \cs_new_protected:Npn \exp_not:N \__talk_decode_action:w
112     #1 \tl_to_str:n { @ } #2 \tl_to_str:n { @ } #3 \exp_not:N \q_stop
113 }
114 {
115     \tl_if_blank:nTF {#2}
116     { \__talk_decode_overlays:nn { overlays } {#1} }
117     {
118         \cs_if_exist:cTF { __talk_action_ #1 :N }
119         {
120             \bool_set_false:N \l__talk_decode_pure_bool
121             \str_set:Nn \l__talk_decode_action_str {#1}
122             \tl_if_blank:nF {#2}
123             { \__talk_decode_overlays:nn { actions } {#2} }
124         }
125         {
126             \msg_error:nnV { talk } { bad-action-spec }
127             \l__talk_decode_arg_str
128         }
129     }
130 }

```

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

__talk_decode_overlays:nn

__talk_decode_overlays:nN

\@_decode_overlay_+:nw

__talk_decode_overlay_.:nw

__talk_decode_overlay_aux:nN

__talk_decode_overlay_offset:nNn

__talk_decode_overlay_offset:nNn

The loop here needs to replace all + and . characters by the current automatic value, allowing for any offsets. Stepping the value assigned here is done in the outer loop (see above).

```

131 \cs_new_protected:Npn \__talk_decode_overlays:nn #1#2
132 {
133     \__talk_decode_overlays:nN {#1} #2 \q_recursion_tail \q_recursion_stop
134     \__talk_decode_check:n {#1}
135 }
136 \cs_new_protected:Npn \__talk_decode_overlays:nN #1#2
137 {
138     \quark_if_recursion_tail_stop:N #2

```

```

139 \cs_if_exist_use:cF { __talk_decode_overlay_ #2 :nw }
140 {
141   \str_put_right:cn { l__talk_decode_ #1 _str } {#2}
142   \__talk_decode_overlays:nN
143 }
144 {#1}
145 }
146 \cs_new_protected:cpn { __talk_decode_overlay_+:nw } #1
147 {
148   \bool_set_true:N \l__talk_decode_step_bool
149   \__talk_decode_overlay_aux:nNN {#1} 1
150 }
151 \cs_new_protected:cpn { __talk_decode_overlay_.:nw } #1
152 { \__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 (.

```

153 \cs_new_protected:Npn \__talk_decode_overlay_aux:nNN #1#2#3
154 {
155   \quark_if_recursion_tail_stop_do:Nn #3
156   {
157     \__talk_decode_overlay_offset:nNn {#1} #2 { 0 }
158     \q_recursion_tail \q_recursion_stop
159   }
160   \token_if_eq_meaning:NNTF #3 ( % )
161   { \__talk_decode_overlay_offset:nNn {#1} #2 { } }
162   { \__talk_decode_overlay_offset:nNn {#1} #2 { 0 } #3 }
163 }

```

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.

```

164 \cs_new_protected:Npn \__talk_decode_overlay_offset:nNn #1#2#3#4
165 {
166   \quark_if_recursion_tail_stop_do:Nn #4
167   {
168     \msg_error:nnV { talk } { bad-action-spec }
169     \l__talk_decode_arg_str
170   } % (
171   \token_if_eq_meaning:NNTF #4 )
172   { \__talk_decode_overlay_offset:nNn {#1} #2 {#3} }
173   { \__talk_decode_overlay_offset:nNn {#1} #2 {#3#4} }
174 }

```

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

```

175 \cs_new_protected:Npn \__talk_decode_overlay_offset:nNn #1#2#3
176 {
177   \str_put_right:ce { l__talk_decode_ #1 _str }
178   { \int_max:nn { 0 } { #3 + \g__talk_pauses_int + #2 } }
179   \__talk_decode_overlays:nN {#1}
180 }

```

(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.

```

181 \cs_new_protected:Npn \__talk_decode_check:n #1
182 {
183   \clist_set:cv { l__talk_decode_ #1 _clist } { l__talk_decode_ #1 _str }
184   \clist_if_in:cnTF { l__talk_decode_ #1 _clist } { * }
185   { \bool_set_true:c { l__talk_decode_ #1 _bool } }
186   {
187     \clist_map_inline:cn { l__talk_decode_ #1 _clist }
188     { \__talk_decode_check:nw {#1} 0 ##1 - - \q_stop }
189   }
190 }

```

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.

```

191 \cs_new_protected:Npn \__talk_decode_check:nw #1#2 - #3 - #4 \q_stop
192 {
193   \tl_if_empty:nTF {#4}
194   { \__talk_decode_check_single:nn {#1} {#2} }
195   {
196     \tl_if_blank:nTF {#3}
197     { \__talk_decode_check_range:nnn {#1} {#2} { \c_max_int } }
198     { \__talk_decode_check_range:nnn {#1} {#2} {#3} }
199   }
200 }
201 \cs_new_protected:Npn \__talk_decode_check_single:nn #1#2
202 {
203   \int_compare:nNnTF \g__talk_slide_int = {#2}
204   { \bool_set_true:c { l__talk_decode_ #1 _bool } }
205   {
206     \int_compare:nNnT {#2} > \g__talk_slide_int
207     { \bool_gset_true:N \g__talk_slide_continue_bool }
208   }
209 }

```

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.

```

210 \cs_set_protected:Npn \__talk_decode_check_range:nnn #1#2#3
211 {
212   \int_compare:nNnF \g__talk_slide_int > {#3}
213   {
214     \int_compare:nNnTF \g__talk_slide_int < {#2}
215     { \bool_gset_true:N \g__talk_slide_continue_bool }
216     {
217       \bool_set_true:c { l__talk_decode_ #1 _bool }
218       \bool_lazy_and:nnT
219       { \int_compare_p:nNn \g__talk_slide_int < {#3} }
220       { \int_compare_p:nNn {#3} < \c_max_int }
221       { \bool_gset_true:N \g__talk_slide_continue_bool }
222     }
223   }

```

```

223         }
224     }
225 }

(End of definition for \_talk\_decode\_check:n and others.)

226 \msg_new:nnnn { talk } { bad-action-spec }
227 { Bad~overlay~specification~"#1". }
228 {
229     The~overlay~specification~given~doesn't~follow~the~pattern~described~in~
230     the~ltx-talk-manual:~it~has~been~ignored.
231 }
232 </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   \box_gclear:N \g__talk_footnote_box
78   \__talk_cnt_save:
79   \vbox_set:Nw \l__talk_slide_box
80   \tl_gclear:N \g__talk_onslide_tl
81 }
82 \cs_new_protected:Npn \__talk_slide_end:
83 {
84   \tl_use:N \g__talk_onslide_tl
85   \vbox_set_end:
86   \bool_if:NT \g__talk_slide_continue_bool
87   { \__talk_cnt_restore: }
88   \vbox_to_ht:nn { \textheight }
89   {
90     \use:c { __talk_slide_align_ \l__talk_frame_alignment_tl :n }
91     { \vbox_unpack_drop:N \l__talk_slide_box }
92     \box_if_empty:NF \g__talk_footnote_box
93     {
94       \footnoterule
95       \vbox_unpack_drop:N \g__talk_footnote_box
96     }
97   }
98   \clearpage
99 }
```

(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`

```
100 \cs_new_protected:Npn \__talk_slide_align_bottom:n #1
```

`__talk_slide_align_stretch:n`

```
101 {
102   \skip_vertical:n { Opt~plus~1fil }
103   #1
```

`__talk_slide_align_top:n`

```

104     \skip_vertical:n { Opt }
105   }
106 \cs_new_protected:Npn \__talk_slide_align_center:n #1
107 {
108   \skip_vertical:n { Opt~plus~0.5fil }
109   #1
110   \skip_vertical:n { Opt~plus~0.5fil }
111 }
112 \cs_new_protected:Npn \__talk_slide_align_stretch:n #1
113 {
114   \skip_vertical:n { Opt }
115   #1
116   \skip_vertical:n { Opt }
117 }
118 \cs_new_protected:Npn \__talk_slide_align_top:n #1
119 {
120   \skip_vertical:n { Opt }
121   #1
122   \skip_vertical:n { Opt~plus~1fil }
123 }

```

(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.

```

124 \seq_new:N \l__talk_cnt_reset_seq
125 \seq_set_from_clist:Nn \l__talk_cnt_reset_seq
126 {
127   equation      ,
128   footnote      ,
129   mpfootnote    ,
130   parentequation
131 }
132 \seq_map_inline:Nn \l__talk_cnt_reset_seq
133 {
134   \int_new:c { g__talk_saved_ #1 _int }
135   \int_gset_eq:cc { g__talk_saved_ #1 _int } { c@ #1 }
136 }

```

(End of definition for \l__talk_cnt_reset_seq.)

__talk_cnt_save: A simple save-and-restore pair.

__talk_cnt_restore:

```

137 \cs_new_protected:Npn \__talk_cnt_save:
138 {
139   \seq_map_inline:Nn \l__talk_cnt_reset_seq
140   { \int_gset_eq:cc { g__talk_saved_ ##1 _int } { c@ ##1 } }
141 }
142 \cs_new_protected:Npn \__talk_cnt_restore:
143 {
144   \seq_map_inline:Nn \l__talk_cnt_reset_seq

```



```

145     { \int_gset_eq:cc { c@ ##1 } { g__talk_saved_ ##1 _int } }
146   }

```

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

\@definecounter Track all counters for resetting.

```

\std@definecounter
147 \cs_new_eq:NN \std@definecounter \@definecounter
148 \cs_gset_protected:Npn \@definecounter #1
149   {
150     \std@definecounter {#1}
151     \int_new:c { g__talk_saved_ #1 _int }
152     \seq_gput_right:Nn \l__talk_cnt_reset_seq {#1}
153   }

```

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

1.3 Frame options

\l__talk_frame_alignment_tl

```

154 \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

```

155 \keys_define:nn { talk / frame }
156   {
157     action-spec .str_set:N
158       = \l__talk_action_spec_str ,
159     tag-slides .str_set:N
160       = \l__talk_frame_tagging_str ,
161     vertical-alignment .choices:nn =
162       { bottom , center , stretch , top }
163       {
164         \tl_set_eq:NN \l__talk_frame_alignment_tl
165         \l_keys_value_tl
166       }
167   }
168 \keys_set:nn { talk / frame }
169   {
170     action-spec      = ,
171     tag-slides       = n ,
172     vertical-alignment = center
173   }

```

(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
__talk_header_tag_begin:e
__talk_header_tag_end:

Generalized control for inserting material into the header area (which is otherwise outside of tagging).

```

174 \cs_new_protected:Npn \__talk_header_tag_begin:n #1
175   {
176     \tag_resume:n { header }

```

```

177 \tag_mc_end:
178 \tag_struct_begin:n {#1}
179 \tag_mc_begin:n { }
180 }
181 \cs_generate_variant:Nn \__talk_header_tag_begin:n { e }
182 \cs_new_protected:Npn \__talk_header_tag_end:
183 {
184 \tag_mc_end:
185 \tag_struct_end:
186 \tag_mc_begin:n { artifact }
187 \tag_suspend:n { header }
188 }

```

(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
189 \NewTemplateType { footer-element } { 1 }
190 \DeclareTemplateInterface { footer-element } { talk } { 1 }
191 {
192 color : tokenlist ,
193 font : tokenlist = ,
194 left-hspace : length = 0em ,
195 right-hspace : length = 0em
196 }
197 \DeclareTemplateCode { footer-element } { talk } { 1 }
198 {
199 color = \l__talk_footelem_color_tl ,
200 font = \l__talk_footelem_font_tl ,
201 left-hspace = \l__talk_footelem_left_skip ,
202 right-hspace = \l__talk_footelem_right_skip
203 }
204 {
205 \tl_if_empty:nF {#1}
206 {
207 \hspace { \l__talk_footelem_left_skip }
208 \group_begin:
209 \tl_if_empty:NF \l__talk_footelem_color_tl
210 { \color_select:V \l__talk_footelem_color_tl }
211 \l__talk_footelem_font_tl
212 #1
213 \group_end:
214 \hspace { \l__talk_footelem_right_skip }
215 }
216 }
217 \DeclareInstance { footer-element } { date } { talk } { }
218 \DeclareInstance { footer-element } { author } { talk } { }
219 \DeclareInstance { footer-element } { title } { talk } { }
220 \DeclareInstance { footer-element } { subtitle } { talk } { }
221 \DeclareInstance { footer-element } { institute } { talk } { }
222 \DeclareInstance { footer-element } { framenumbers } { talk } { }
223 \DeclareInstance { footer-element } { totalframes } { talk } { }

```

(End of definition for \l__talk_footelem_left_skip and others.)

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

```

224 \NewTemplateType { header } { 0 }
225 \DeclareTemplateInterface { header } { talk } { 0 }
226 {
227   background-color : tokenlist,
228   color            : tokenlist = structure ,
229   font             : tokenlist = \normalfont ,
230   height           : length = \Gm@tmargin + \headsep ,
231   left-hspace      : skip = \Gm@lmargin ,
232   print-frame-title : boolean = true ,
233   right-hspace     : skip = \Gm@rmargin
234 }
235 \DeclareTemplateCode { header } { talk } { 0 }
236 {
237   background-color = \l__talk_header_bg_tl ,
238   color           = \l__talk_header_fg_tl ,
239   font            = \l__talk_header_font_tl ,
240   height          = \l__talk_header_ht_dim ,
241   left-hspace     = \l__talk_header_left_skip ,
242   print-frame-title = \l__talk_header_frametitle_bool ,
243   right-hspace    = \l__talk_header_right_skip
244 }
245 {
246   \noindent
247   \__talk_wallpaper_hruler:Nnn
248   \l__talk_header_bg_tl
249   { \l__talk_header_ht_dim - \headsep }
250   { \headsep }
251   \skip_horizontal:n { \l__talk_header_left_skip }
252   \group_begin:
253     \tl_if_empty:NF \l__talk_header_fg_tl
254     { \color_select:V \l__talk_header_fg_tl }
255     \l__talk_header_font_tl
256     \bool_if:NT \l__talk_header_frametitle_bool
257     {
258       \ExpandArgs { nnV }
259       \UseInstance { frametitle } { header }
260       \g__talk_frame_title_tl
261     }
262   \group_end:
263 }
264 \DeclareInstance { header } { std } { talk } { }
265 \AddToHook { begindocument }
266 {
267   \DeclareInstanceCopy { header } { wallpaper } { std }
268   \EditInstance { header } { wallpaper }
269     { print-frame-title = false }
270 }

```

(End of definition for \l__talk_header_bg_tl and others.)

`\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`

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.

```

271 \NewTemplateType { footer } { 0 }
272 \DeclareTemplateInterface { footer } { talk } { 0 }
273 {
274   background-color : tokenlist ,
275   color            : tokenlist ,
276   element-order    : commalist ,
277   font             : tokenlist = \tiny ,
278   left-hspace      : length = \Gm@lmargin ,
279   right-hspace     : length = \Gm@rmargin ,
280   separator        : tokenlist = \hfil
281 }
282 \DeclareTemplateCode { footer } { talk } { 0 }
283 {
284   background-color = \l__talk_footer_bg_tl ,
285   color           = \l__talk_footer_fg_tl ,
286   element-order   = \l__talk_footer_order_clist ,
287   font            = \l__talk_footer_font_tl ,
288   left-hspace     = \l__talk_footer_left_skip ,
289   right-hspace    = \l__talk_footer_right_skip ,
290   separator       = \l__talk_footer_sep_tl
291 }
292 {
293   \noindent
294   \__talk_wallpaper_hrule:Nnn
295   \l__talk_footer_bg_tl
296   { \footskip }
297   { \Gm@bmargin - \footskip }
298   \skip_horizontal:n { \l__talk_footer_left_skip }
299   \vbox_set_to_wd:Nnn \l__talk_tmp_box
300   {
301     \paperwidth
302     - \l__talk_footer_left_skip
303     - \l__talk_footer_right_skip
304   }
305   {
306     \tl_if_empty:NF \l__talk_footer_fg_tl
307     { \color_select:V \l__talk_footer_fg_tl }
308     \l__talk_footer_font_tl
309     \clist_pop:NNT \l__talk_footer_order_clist \l__talk_tmp_tl
310     {
311       \ExpandArgs { nVv } \UseInstance { footer-element } \l__talk_tmp_tl
312       { @ \__talk_metadata_name:n { \l__talk_tmp_tl } }
313       \clist_map_inline:Nn \l__talk_footer_order_clist
314       {
315         \tl_if_empty:cF { @ \__talk_metadata_name:n { ##1 } }
316         {
317           \l__talk_footer_sep_tl
318           \ExpandArgs { nnv }
319           \UseInstance { footer-element } {##1}
320           { @ \__talk_metadata_name:n { ##1 } }
321         }
322       }
323     }
324   }
325 }

```

```

322     }
323   }
324   \hfil
325 }
326 \box_use_drop:N \l__talk_tmp_box
327 \skip_horizontal:n { \l__talk_footer_right_skip - \Gm@rmargin }
328 }
329 \DeclareInstance { footer } { std } { talk } { }
330 \AddToHook { begindocument }
331 {
332   \DeclareInstanceCopy { footer } { wallpaper } { std }
333   \EditInstance { footer } { wallpaper }
334   { element-order = }
335 }

```

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

`__talk_metadata_name:n` A simple auxiliary to shorten metadata names if appropriate. Full expansion is applied as this avoids any issue with stored names.

```

336 \cs_new:Npn \__talk_metadata_name:n #1
337 {
338   \tl_if_exist:cTF { @ short #1 }
339   { short #1 }
340   {#1}
341 }

```

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

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

```

342 \cs_new_protected:Npn \__talk_wallpaper_hrulerule:Nnn #1#2#3
343 {
344   \skip_horizontal:n { -\Gm@lmargin }
345   \tl_if_empty:NF #1
346   {
347     \group_begin:
348     \color_select:V #1
349     \rule:nnn { \paperwidth } {#2} {#3}
350     \skip_horizontal:n { -\paperwidth }
351     \group_end:
352   }
353 }

```

(End of definition for `__talk_wallpaper_hrulerule: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.

```

354 \cs_set_nopar:Npn \ps@plain
355 {
356   \cs_set_nopar:Npn \@oddhead
357   {
358     \hfil
359   }

```

```

360 \cs_set_nopar:Npn \@oddfoot { }
361 \cs_set_eq:NN \@evenhead \@oddhead
362 \cs_set_eq:NN \@evenfoot \@oddfoot
363 }
364 \cs_set_nopar:Npn \ps@wallpaper
365 {
366 \cs_set_nopar:Npn \@oddhead
367 {
368 \UseInstance { header } { wallpaper }
369 \hfil
370 }
371 \cs_set_nopar:Npn \@oddfoot
372 {
373 \UseInstance { footer } { wallpaper }
374 \hfil
375 }
376 \cs_set_eq:NN \@evenhead \@oddhead
377 \cs_set_eq:NN \@evenfoot \@oddfoot
378 }
379 \cs_new_nopar:Npn \ps@talk
380 {
381 \cs_set_nopar:Npn \@oddhead
382 {
383 \UseInstance { header } { std }
384 \hfil
385 }
386 \cs_set_nopar:Npn \@oddfoot { \UseInstance { footer } { std } }
387 \cs_set_eq:NN \@evenhead \@oddhead
388 \cs_set_eq:NN \@evenfoot \@oddfoot
389 }
390 \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.

```
391 \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).

```
392 \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).

```
393 \bool_new:N \l__talk_frame_verb_bool
```

(End of definition for \l__talk_frame_verb_bool.)

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

```

\c@frame      394 \int_new:N \g__talk_frame_int
\theframe     395 \cs_new_eq:NN \c@frame \g__talk_frame_int
\@framenumber 396 \cs_new:Npn \theframe { \@arabic \c@frame }
               397 \cs_new:Npn \@framenumber { \arabic { frame } }

```

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

`\@totalframes` The total frames can be handled using the kernel properties.

```

398 \property_new:nnnn { totalframes } { shipout } { -1 }
399 { \int_use:N \g__talk_frame_int }
400 \AddToHook { enddocument / afterlastpage }
401 { \property_record:nn { lastpage } { totalframes } }
402 \cs_new:Npn \@totalframes { \property_ref:nn { lastpage } { totalframes } }

```

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

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

```

403 \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.

```

404 \cs_new_protected:Npn \__talk_frame_process:nn #1#2
405 {
406   \int_gincr:N \g__talk_frame_int
407   \bool_set_true:N \l__talk_frame_bool
408   \__talk_slide:nn {#1} {#2}
409 }

```

(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.

```

410 \cs_new_protected:Npn \__talk_frame_tag:n #1
411 {
412   \tag_struct_begin:n { tag = frame }
413   \int_gset:Nn \g__talk_frame_struct_int { \tag_get:n { struct_num } }
414   \bool_gset_true:N \g__talk_frame_tag_bool
415   #1
416   \tag_struct_end:
417 }

```

(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.

```

418 \cs_new_protected:Npn \__talk_frame_notag:n #1
419 {
420   \tag_mc_begin:n { artifact }
421   \tag_suspend:n { frame }
422   \bool_gset_false:N \g__talk_frame_tag_bool
423   #1
424   \par

```

```

425 \tag_resume:n { frame }
426 \tag_mc_end:
427 }

```

(End of definition for __talk_frame_notag:n.)

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

```

428 \bool_if:NTF \l__talk_frame_title_bool
429 {
430   \RenewDocumentEnvironment { frame }
431     { D <> { all } = { action-spec } 0 { } +m +b }
432     {
433       \keys_set:nn { talk / frame } {#2}
434       \bool_set_false:N \l__talk_frame_verb_bool
435       \__talk_frame_process:nn {#1} { \frametitle {#3} #4 }
436     }
437     { }
438   \NewDocumentEnvironment { frame* }
439     { D <> { all } = { action-spec } 0 { } +m c }
440     {
441       \keys_set:nn { talk / frame } {#2}
442       \bool_set_true:N \l__talk_frame_verb_bool
443       \tl_gset:Nn \g__talk_frame_title_tl {#3}
444       \exp_args:Nne \__talk_frame_process:nn {#1}
445         { \tl_to_str:n { \frametitle } \exp_not:n { {#3} #4 } }
446     }
447     { }
448   }
449   {
450     \RenewDocumentEnvironment { frame }
451       { !D <> { all } = { action-spec } !0 { } +b }
452       {
453         \keys_set:nn { talk / frame } {#2}
454         \bool_set_false:N \l__talk_frame_verb_bool
455         \__talk_frame_process:nn {#1} {#3}
456       }
457       { }
458     \NewDocumentEnvironment { frame* }
459       { !D <> { all } = { action-spec } !0 { } c }
460       {
461         \keys_set:nn { talk / frame } {#2}
462         \bool_set_true:N \l__talk_frame_verb_bool
463         \__talk_frame_process:nn {#1} {#3}
464       }
465       { }
466     }

```

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

```

467 </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.)

`\l__talk_column_int` For tracking which column we are in, and allowing for nesting.

```
\g__talk_column_int
9 \int_new:N \l__talk_column_int
10 \int_new:N \g__talk_column_int
```

(End of definition for \l__talk_column_int and \g__talk_column_int.)

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

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

```

28   \par
29   \__talk_action_end:
30 }

```

\l__talk_column_alignment_tl

```

31 \keys_define:nn { talk / column }
32 {
33   b .meta:n =
34     { vertical-alignment = bottom } ,
35   b .value_forbidden:n = true ,
36   c .meta:n =
37     { vertical-alignment = center } ,
38   c .value_forbidden:n = true ,
39   t .meta:n =
40     { vertical-alignment = top } ,
41   t .value_forbidden:n = true ,
42   vertical-alignment .choices:nn =
43     { bottom , center , top }
44     {
45       \tl_set_eq:NN \l__talk_column_alignment_tl
46       \l_keys_value_tl
47     }
48 }
49 \keys_set:nn { talk / column }
50 {
51   vertical-alignment = center
52 }

```

(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.

```

53 \cs_new_protected:Npn \__talk_column_align_bottom:n #1
54   { \vbox:n {#1} }
55 \cs_new_protected:Npn \__talk_column_align_center:n #1
56   {
57     \vbox:n
58     {
59       \hbox:n
60       {
61         \box_move_down:nn
62         {
63           0.5 \box_ht:N \l__talk_tmp_box
64           - \tex_fontdimen:D 22 ~ \tex_textfont:D 2 ~
65         }
66         { \vbox:n {#1} }
67       }
68     }
69   }
70 \cs_new_protected:Npn \__talk_column_align_top:n #1
71   { \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.

```

72 \NewDocumentEnvironment { column } { D <> { all } O { } m }
73 {
74   \par
75   \int_gincr:N \g__talk_column_int
76   \int_compare:nNnF \g__talk_column_int = 1
77     { \hfil }
78   \keys_set:nn { talk / column } {#2}
79   \vbox_set_to_wd:Nnw \l__talk_tmp_box {#3}
80     \dim_set:Nn \textwidth {#3}
81     \dim_set_eq:NN \columnwidth \textwidth
82   \@parboxrestore
83   \leavevmode
84   \raggedright
85   \__talk_action_begin:n {#1}
86   \ignorespaces
87 }

```

The `\@ignore` here means that any spaces after `\end{column}` are suppressed by a `\ignorespaces` inserted by the kernel. The `\par` before `__talk_action_end:` is needed as the group formed for actions would otherwise trap for example alignment changes.

```

88 {
89   \par
90   \__talk_action_end:
91   \vbox_set_end:
92   \use:c { __talk_column_align_ \l__talk_column_alignment_tl :n }
93     { \vbox_unpack_drop:N \l__talk_tmp_box }
94   \par
95   \@ignoretrue
96 }

```

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.

```

97 \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.

```

98 \NewTemplateType { floatenv } { 2 }
99 \DeclareTemplateInterface { floatenv } { talk } { 2 }
100 {
101   float-placement : tokenlist ,
102   horizontal-alignment : choice { left , center , right } = left
103 }
104 \DeclareTemplateCode { floatenv } { talk } { 2 }
105 {

```

```

106 float-placement = \l__talk_tmp_tl ,
107 horizontal-alignment =
108 {
109   left = \tl_set:Nn \l__talk_float_alignment_tl { flushleft } ,
110   center = \tl_set:Nn \l__talk_float_alignment_tl { center } ,
111   right = \tl_set:Nn \l__talk_float_alignment_tl { flushright }
112 }
113 }
114 {
115   \SetTemplateKeys { floatenv } { talk } {#1}
116   \begin { minipage } { \columnwidth }
117     \begin { \l__talk_float_alignment_tl }
118       \cs_set_nopar:Npn \@capttype {#2}
119     }
120   \DeclareInstance { floatenv } { std } { talk } { horizontal-alignment = left }

```

`\endfloatenv` And the common end function.

```

121 \cs_new_protected:Npn \endfloatenv
122 {
123   \end { \l__talk_float_alignment_tl }
124   \end { minipage }
125 }

```

(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.)`

```

126 \clist_map_inline:nn { figure , table }
127 {
128   \NewDocumentEnvironment {#1} { D <> { all } = { float-placement } 0 { } }
129   {
130     \__talk_action_begin:n {##1}
131     \UseInstance { floatenv } { std } {##2} {#1}
132   }
133   {
134     \endfloatenv
135     \__talk_action_end:
136   }

```

`\c@figure` The standard variables needed to make captions work (nothing for list of floats, as at present those are not offered).

```

\thefigure
\c@table 137 \newcounter {#1}
\thetable 138 \tl_new:c { #1 name }
\figurename 139 \tl_set:ce { #1 name } { \text_titlecase_first:n {#1} }
\tableename 140 \tl_new:c { fnum@ #1 }
\fnum@figure 141 \tl_set:ce { fnum@ #1 }
\fnum@table 142 { \exp_not:c { #1 name } \exp_not:N \nobreakspace \exp_not:c { the #1 } }
143 }

```

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

The spacing values needed for the standard function.

```

144 \newlength \abovecaptionskip
145 \newlength \belowcaptionskip
146 \setlength \abovecaptionskip { 7pt }
147 \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.

```

148 \cs_set_protected:Npn \@caption #1 [ #2 ] #3
149 {
150   \par
151   \begingroup
152     \@parboxrestore
153     \if@minipage \@setminipage \fi
154     \normalsize
155     \@makecaption { \csname fnum@ #1 \endcsname } { \ignorespaces #3 }
156     \par
157   \endgroup
158 }

```

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

1.3 Footnotes

`\g__talk_footnote_box` Holds footnotes as they are constructed.

```

159 \box_new:N \g__talk_footnote_box

```

(End of definition for \g__talk_footnote_box.)

`\g_talk_footnote_overlay_seq` For tracking the overlays to apply.

```

160 \seq_new:N \g_talk_footnote_overlay_seq

```

(End of definition for \g_talk_footnote_overlay_seq.)

`\stdfootnote`

```

161 \NewCommandCopy \stdfootnote \footnote

```

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

`\footnote` Sort-of overlay aware!

```

162 \RenewDocumentCommand \footnote { D <> { all } o +m }
163 {
164   \seq_gpush:Nn \g__talk_footnote_overlay_seq {#1}
165   \IfNoValueTF {#2}
166     { \stdfootnote {#3} }
167     { \stdfootnote [ {#2} ] {#3} }
168 }

```

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

This socket receives all of the footnote content: in the standard setup it would be an insert. Hence this is the best place to grab the entire content. Notice that the footnote rule is only inserted when the box is used, if it turns out it's needed. The overlay code is added here as it needs to be inside the box used to collect the footnotes but around all of the content: currently there's not a "tighter" place to target.

```

169 \NewSocketPlug { fntext / process } { talk }
170 {
171   \vbox_gset:Nn \g__talk_footnote_box
172     {
173       \vbox_unpack:N \g__talk_footnote_box

```

```

174         \seq_gpop_left:NN \g__talk_footnote_overlay_seq
175         \l__talk_tmp_tl
176         \exp_args:NV \__talk_decode_parse:n \l__talk_tmp_tl
177         \__talk_action_uncover:N \l__talk_decode_overlays_bool
178         #1
179         \__talk_action_uncover_end:N \l__talk_decode_overlays_bool
180     }
181 }
182 \AssignSocketPlug { fntext / process } { talk }

\@makefntext Use a copy of the standard setup.
183 \cs_new_eq:NN \@makefntext \fnote_makefntext:n

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

184 \</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.)

\l__talk_shuffle_skip For tracking.

```
17 \skip_new:N \l__talk_shuffle_skip
```

(End of definition for \l__talk_shuffle_skip.)

__talk_shuffle_skip:n As opacity uses whatsits at present, we need to make sure that any spaces come *after* them. This is done by “shuffling” the last skip past the opacity.

```
18 \cs_new_protected:Npn __talk_shuffle_skip:n #1
19 {
20   \skip_set_eq:NN \l__talk_shuffle_skip \tex_lastskip:D
21   \bool_lazy_and:nnTF
22   { ! \skip_if_eq_p:nn \l__talk_shuffle_skip { Opt } }
23   {
24     \bool_lazy_or_p:nn
25     { \mode_if_horizontal_p: }
26     { \mode_if_vertical_p: }
27   }
28   {
29     \tex_unskip:D
```



```

30         #1
31         \mode_if_horizontal:TF
32         { \skip_horizontal:n }
33         { \skip_vertical:n }
34         \l__talk_shuffle_skip
35     }
36     {#1}
37 }

```

(End of definition for __talk_shuffle_skip:n.)

1.2 Opacity utilities

Currently, opacity is applied using what sits at a low level. That means that to preserve spacing, we need to insert no-op versions in various places. To do that and get correct overlays, we need to track the current opacity. At present, this seems very ltx-talk-specific, so is handled here with a few auxiliaries.

```

\__talk_opacity_begin:n
\__talk_opacity_end:
38 \cs_new_protected:Npn \__talk_opacity_begin:n #1
39 { \__talk_shuffle_skip:n { \opacity_begin:n {#1} } }
40 \cs_new_protected:Npn \__talk_opacity_end:
41 { \__talk_shuffle_skip:n { \opacity_end: } }

```

(End of definition for __talk_opacity_begin:n and __talk_opacity_end:.)

1.3 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.

```

\__talk_action_alert:N
42 \cs_new_protected:Npn \__talk_action_alert:N #1
43 {
44     \bool_if:NTF #1
45     { \color_select:n { alert } }
46     { \color_select:n { . } }
47 }

```

(End of definition for __talk_action_alert:N.)

```

\__talk_action_invisible:N
\__talk_action_invisible_end:N
\__talk_action_visible:N
\__talk_action_visible_end:N
48 \cs_new_protected:Npn \__talk_action_invisible:N #1
49 {
50     \bool_if:NTF #1
51     { \__talk_opacity_begin:n { 0 } }
52     { \__talk_opacity_begin:n { 1 } }
53 }
54 \cs_new_protected:Npn \__talk_action_invisible_end:N #1
55 { \__talk_opacity_end: }
56 \cs_new_protected:Npn \__talk_action_visible:N #1
57 {
58     \bool_if:NTF #1

```

```

59     { \_talk_opacity_begin:n { 1 } }
60     { \_talk_opacity_begin:n { 0 } }
61   }
62   \cs_new_protected:Npn \_talk_action_visible_end:N #1
63   { \_talk_opacity_end: }

```

(End of definition for _talk_action_invisible:N and others.)

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

```

64   \cs_new_protected:Npn \_talk_action_only:N #1
65   {
66     \bool_if:NF #1
67     { \vbox_set:Nw \l__talk_tmp_box }
68   }
69   \cs_new_protected:Npn \_talk_action_only_end:N #1
70   {
71     \bool_if:NF #1
72     { \vbox_set_end: }
73   }

```

(End of definition for _talk_action_only:N and _talk_action_only_end:N.)

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

```

74   \NewTemplateType { hidden } { 0 }
75   \DeclareTemplateInterface { hidden } { talk } { 0 }
76   { opacity : real = 0 }
77   \DeclareTemplateCode { hidden } { talk } { 0 }
78   { opacity = \l__talk_uncover_hidden_fp }
79   { \_talk_opacity_begin:n { \l__talk_uncover_hidden_fp } }
80   \DeclareInstance { hidden } { std } { talk } { }

```

(End of definition for \l__talk_uncover_hidden_fp.)

_talk_action_uncover:N Use the template: we may need to extend that to deal with the end-of-template case
_talk_action_uncover_end:N later.

```

81   \cs_new_protected:Npn \_talk_action_uncover:N #1
82   {
83     \bool_if:NTF #1
84     { \_talk_opacity_begin:n { 1 } }
85     { \UseInstance { hidden } { std } }
86   }
87   \cs_new_protected:Npn \_talk_action_uncover_end:N #1
88   { \_talk_opacity_end: }

```

(End of definition for _talk_action_uncover:N and _talk_action_uncover_end:N.)

\invisible All generated automatically using the above implementations.

```

\uncover
\visible
89   \clist_map_inline:nn { invisible , uncover , visible }
90   {
91     \ExpandArgs { cne } \NewDocumentCommand {#1}
92     { > { \_talk_overlay_arg:n } D <> { all } +m }
93     {
94       \exp_not:c { __talk_action_ #1 :N } ##1
95       ##2

```

```

96         \exp_not:c { __talk_action_ #1 _end:N } ##1
97     }

```

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

`invisibleenv (env.)` And the environment versions.

```

\uncoverenv (env.) 98     \ExpandArgs { nnee } \NewDocumentEnvironment { #1 env }
\visibleenv (env.) 99     { > { \__talk_overlay_arg:n } D <> { all } }
100     { \exp_not:c { __talk_action_ #1 :N } ##1 }
101     { \exp_not:c { __talk_action_ #1 _end:N } ##1 }
102 }

```

`\alert` The `\alert` command requires a group to contain color, so is done separately even though it still uses basically the same mechanism.

```

103 \NewDocumentCommand \alert { > { \__talk_overlay_arg:n } D <> { all } +m }
104 {
105     \group_begin:
106     \__talk_action_alert:N #1
107     #2
108     \group_end:
109 }

```

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

`alertenv (env.)` As does the environment.

```

110 \NewDocumentEnvironment { alertenv } { > { \__talk_overlay_arg:n } D <> { all } }
111 { \__talk_action_alert:N #1 }
112 { }

```

`\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).

```

113 \NewDocumentCommand \only { D <> { all } +m }
114 {
115     \__talk_if_overlay:nT {#1}
116     {#2}
117 }

```

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

`onlyenv (env.)` The environment version could be done above, but it is clearer to keep this code entirely separate from the rest.

```

118 \NewDocumentEnvironment { onlyenv } { > { \__talk_overlay_arg:n } D <> { all } }
119 { \__talk_action_only:N #1 }
120 { \__talk_action_only_end:N #1 }

```

```

\l__talk_saved_overlays_bool
\l__talk_saved_action_str 121 \bool_new:N \l__talk_saved_overlays_bool
\l__talk_saved_actions_bool 122 \str_new:N \l__talk_saved_action_str
123 \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`.)

\l__talk_overlay_all_bool

124 \bool_new:N \l__talk_overlay_all_bool

(End of definition for \l__talk_overlay_all_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. When an \onslide/\pause is active, it takes priority: sorted by applying up-front. Actions can be skipped entirely if the overlay spec is simply all, as there will never be any spacing issues, *etc.*

__talk_action_begin:n

__talk_action_begin_aux:n

__talk_action_end:

125 \NewDocumentCommand \action { D <> { all } +m }

126 {

127 \group_begin:

128 __talk_action_begin:n {#1}

129 #2

130 __talk_action_end:

131 \group_end:

132 }

133 \NewDocumentEnvironment { actionenv } { D <> { all } }

134 { __talk_action_begin:n {#1} }

135 { __talk_action_end: }

136 \cs_new_protected:Npn __talk_action_begin:n #1

137 {

138 \group_begin:

139 \str_if_eq:nnTF {#1} { all }

140 { \bool_set_true:N \l__talk_overlay_all_bool }

141 {

142 \bool_set_false:N \l__talk_overlay_all_bool

143 __talk_action_begin_aux:n {#1}

144 }

145 }

146 \cs_new_protected:Npn __talk_action_begin_aux:n #1

147 {

148 __talk_decode_parse:n {#1}

149 \bool_set_eq:NN \l__talk_saved_overlays_bool

150 \l__talk_decode_overlays_bool

151 \str_set_eq:NN \l__talk_saved_action_str

152 \l__talk_decode_action_str

153 \bool_set_eq:NN \l__talk_saved_actions_bool

154 \l__talk_decode_actions_bool

155 \tl_if_empty:NTF \g__talk_onslide_tl

156 {

157 \bool_if:NTF \l__talk_decode_overlays_bool

158 {

159 \cs_if_exist_use:cF

160 { __talk_action_ \l__talk_decode_action_str :N }

161 { \use_none:n }

162 \l__talk_decode_actions_bool

163 }

164 { \UseInstance { hidden } { std } }

165 }

166 { __talk_action_invisible:N \c_true_bool }

167 }

```

168 \cs_new_protected:Npn \__talk_action_end:
169 {
170     \bool_if:NF \l__talk_overlay_all_bool
171     {
172         \bool_if:NTF \l__talk_saved_overlays_bool
173         {
174             \cs_if_exist_use:cF
175             { __talk_action_ \l__talk_saved_action_str _end:N }
176             { \use_none:n }
177             \l__talk_saved_actions_bool
178         }
179         { \__talk_opacity_end: }
180     }
181     \group_end:
182 }

```

(End of definition for \action and others. This function is documented on page ??.)

1.4 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.

```

183 \NewDocumentCommand \alt { D <> { all } +m +m }
184 {
185     \__talk_if_overlay:nTF {#1}
186     {#2}
187     {#3}
188 }

```

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

\onslide Simply make transparent: this is done without grouping so we can work for example in tabular cells.

```

\__talk_onslide:n
189 \NewDocumentCommand \onslide { D <> { all } }
190 {
191     \__talk_onslide:n {#1}
192     \ignorespaces
193 }
194 \cs_new_protected:Npn \__talk_onslide:n #1
195 {
196     \tl_use:N \g__talk_onslide_tl
197     \tl_gclear:N \g__talk_onslide_tl
198     \__talk_if_overlay:nF {#1}
199     {
200         \__talk_opacity_begin:n { 0 }
201         \tl_gput_right:Nn \g__talk_onslide_tl
202         { \__talk_opacity_end: }
203     }
204 }

```

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

`\g__talk_onslide_tl`

205 `\tl_new:N \g__talk_onslide_tl`

(End of definition for `\g__talk_onslide_tl`.)

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

```
206 \NewDocumentCommand \temporal { D <> { all } +m +m +m }
207 {
208   \__talk_if_overlay:nTF {#1}
209     {#3}
210     {
211       \bool_if:NTF \g__talk_slide_continue_bool
212         {#4}
213         {#2}
214     }
215 }
```

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

`\pause` A thin wrapper.

```
216 \NewDocumentCommand \pause { o }
217 {
218   \legacy_if:nF { measuring@ }
219   {
220     \IfNoValueTF {#1}
221       { \int_gincr:N \g__talk_pauses_int }
222       { \int_gset:Nn \g__talk_pauses_int {#1} }
223     \exp_args:Ne \__talk_onslide:n { \int_eval:n { \g__talk_pauses_int + 1 } - }
224   }
225 }
```

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

1.5 Fixed-size areas

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

```
226 \cs_new_protected:Npn \__talk_overprint_begin:n #1
227 {
228   \par
229   \vbox_set_to_wd:Nnw \l__talk_tmp_box {#1}
230   \raggedright
231   \ignorespaces
232 }
```

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

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

```
233 \NewDocumentEnvironment { overlayarea } { m m }
234 { \__talk_overprint_begin:n {#1} }
235 {
236   \vbox_set_end:
237   \vbox_to_ht:nn {#2}
238   {
```

```

239         \box_use_drop:N \l__talk_tmp_box
240         \vfil
241     }
242     \par
243 }

```

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

```

244 \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.

```

245 \cs_new:Npn \__talk_frame_overprint:
246 {
247     \int_to_Roman:n \g__talk_frame_int
248     \int_to_roman:n \l__talk_overprint_int
249 }

```

(End of definition for __talk_frame_overprint:.)

`__talk_overprint_int:env` 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.

`__talk_overprint_check_ht:n`

```

250 \NewDocumentEnvironment { overprint } { 0 { \textwidth } }
251 { \__talk_overprint_begin:n {#1} }
252 {
253     \vbox_set_end:
254     \int_incr:N \l__talk_overprint_int
255     \__talk_overprint_save_ht:
256     \cs_if_exist:cTF
257     { overprint@ \__talk_frame_overprint: }
258     {
259         \dim_compare:vNnTF
260         { overprint@ \__talk_frame_overprint: }
261         > { \box_ht:N \l__talk_tmp_box }
262         {
263             \vbox_to_ht:vn
264             { overprint@ \__talk_frame_overprint: }
265             {
266                 \box_use_drop:N \l__talk_tmp_box
267                 \vfil
268             }
269         }
270         { \box_use_drop:N \l__talk_tmp_box }
271     }
272     { \box_use_drop:N \l__talk_tmp_box }
273     \par
274 }

```

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.

```

275 \cs_new_protected:Npn \__talk_overprint_save_ht:

```

```

276 {
277   \tl_if_exist:cF { g__talk_overprint_ \__talk_frame_overprint: _tl }
278   {
279     \tl_new:c { g__talk_overprint_ \__talk_frame_overprint: _tl }
280     \tl_gset:cn { g__talk_overprint_ \__talk_frame_overprint: _tl }
281     { 0pt }
282   }
283   \tl_gset:ce { g__talk_overprint_ \__talk_frame_overprint: _tl }
284   {
285     \dim_max:vn { g__talk_overprint_ \__talk_frame_overprint: _tl }
286     { \box_ht:N \l__talk_tmp_box }
287   }
288   \legacy_if:nT { @files }
289   {
290     \iow_now:Ne \@auxout
291     {
292       \gdef \exp_not:c { overprint@ \__talk_frame_overprint: }
293       {
294         \exp_not:v { g__talk_overprint_ \__talk_frame_overprint: _tl }
295       }
296     }
297   }
298   \hook_gput_code:nne { enddocument / afterlastpage } { talk }
299   { \__talk_overprint_check_ht:n { \__talk_frame_overprint: } }
300 }
301 \cs_new_protected:Npn \__talk_overprint_check_ht:n #1
302 {
303   \bool_lazy_and:nnF
304   { \exp_not:N \cs_if_exist_p:c { overprint@ #1 } }
305   {
306     \dim_compare_p:vNv { overprint@ #1 } = { g__talk_overprint_ #1 _tl }
307   }
308   {
309     \msg_warning:nn { talk } { overprint-ht }
310     \cs_gset_protected:Npn \__talk_overprint_check_ht:n ##1 { }
311   }
312 }
313 \msg_new:nnn { talk } { overprint-ht }
314 {
315   Overprint~area~height~has~changed:\\
316   rerun~LaTeX.
317 }

```

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

1.6 Adding overlays to existing commands

\textbf 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.

\textit

\textmd

\textnormal 318 \tl_map_inline:nn

\textrm 319 {

\textsc 320 \textbf

\textsf 321 \textit

\textsl

\texttt

\textup

\emph

\stdtextbf

\stdtextit

\stdtextmd

\stdtextnormal

\stdtextrm

\stdtextsc


```

322 \textmd
323 \textnormal
324 \textrm
325 \textsc
326 \textsf
327 \textsl
328 \texttt
329 \textup
330 \emph
331 }
332 {
333 \ExpandArgs { c } \NewCommandCopy { std \cs_to_str:N #1 } #1
334 \ExpandArgs { Nne } \RenewDocumentCommand #1
335 { D <> { all } +m }
336 {
337 \exp_not:N \__talk_if_overlay:nTF {##1}
338 { \exp_not:c { std \cs_to_str:N #1 } }
339 { \exp_not:N \__talk_textcmd_equiv:n }
340 {##2}
341 }
342 }
343 \cs_new_protected:Npn \__talk_textcmd_equiv:n #1
344 {
345 \mode_if_math:TF
346 { { \mbox {#1} } }
347 {
348 \mode_leave_vertical:
349 {#1}
350 }
351 }

```

(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.

```

352 \RequirePackage { graphicx }
353 \NewCommandCopy \stdincludegraphics \includegraphics
354 \RenewDocumentCommand \includegraphics { s D <> { all } o o m }
355 {
356 \__talk_if_overlay:nT {#2}
357 {
358 \use:e
359 {
360 \exp_not:N \stdincludegraphics
361 \IfBooleanT #1 { * }
362 \IfNoValueF {#3} { [ \exp_not:n { {#3} } ] }
363 \IfNoValueF {#4} { [ \exp_not:n { {#4} } ] }
364 }
365 {#5}
366 }
367 }

```

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

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

```

368 \RenewDocumentCommand \label { D <> { 1 } m }
369 {
370   \@bsphack
371   \__talk_if_overlay:nT {#1}
372   { \__talk_label:n {#2} }
373   \@esphack
374 }
375 \cs_new_protected:Npn \__talk_label:n #1
376 {
377   \begingroup
378     \UseHookWithArguments { label } { 1 } {#1}
379     \protected@write \@auxout { }
380     {
381       \string \newlabel {#1}
382       {
383         { \@currentlabel }
384         { \thepage }
385         { \@currentlabelname }
386         { \@currentHref }
387         { \@kernel@reserved@label@data }
388       }
389     }
390   \endgroup
391 }

```

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

```

392 </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 Inserting the frame title requires we deal with tagging as well as appearance: if there is
\__talk_frame_title_tagged:n 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
\g__talk_section_tl
\l__talk_subsection_tl
\g__talk_subsection_tl
\l__talk_subsubsection_tl
\g__talk_subsubsection_tl

```

Two versions of the data store: one set locally (but at the top level) for general use, one set (and more importantly cleared) globally to allow insertion in the header area just once per name.

```

66 \tl_new:N \l__talk_section_tl
67 \tl_new:N \g__talk_section_tl
68 \tl_new:N \l__talk_subsection_tl
69 \tl_new:N \g__talk_subsection_tl
70 \tl_new:N \l__talk_subsubsection_tl
71 \tl_new:N \g__talk_subsubsection_tl

```

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

```

\section
\subsection
\subsubsection
\thesection
\thesubsection
\thesubsubsection

```

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

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 ??.)

<pre> \section \subsection \subsubsection \insertsection \insertsubsection \insertsubsubsection __talk_sect_section:Nnn __talk_sect_subsection:Nnn __talk_sect_subsubsection:Nnn </pre>	<p>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.</p> <pre> 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 \bool_if:NF \exp_not:N \l__talk_frame_bool 96 { 97 __talk_if_overlay:nT {##2} 98 { \exp_not:c { __talk_sect_ #1 :Nnn } ##1 {##3} {##4} } 99 } 100 } 101 \cs_new_protected:Npn \exp_not:c { __talk_sect_ #1 :Nnn } ##1##2##3 102 { 103 \exp_not:N \refstepcounter {#2} 104 \UseTaggingSocket { sec / end } { \use:c { toplevel@ #2 } } 105 \UseTaggingSocket { sec / begin } 106 { 107 { \use:c { toplevel@ #2 } } 108 { 109 tag = 110 \exp_not:N \UseStructureName 111 { sec / \use:c { toplevel@ #2 } } 112 } </pre>
--	--

```

113     }
114     \tl_set:Nn \exp_not:c { l__talk_ #2 _tl } {##3}
115     \UseTaggingSocket { talk / sec / title } {#2}
116     \str_if_eq:nnF {#2} { section }
117     { \tl_clear:N \exp_not:N \l__talk_subsection_tl }
118     \str_if_eq:nnF {#2} { subsection }
119     { \tl_clear:N \exp_not:N \l__talk_subsubsection_tl }
120     \exp_not:N \addcontentsline { toc } {#2}
121     {
122         \exp_not:N \int_compare:nNnF {#1} >
123         { \exp_not:N \value { secnumdepth } }
124         {
125             \exp_not:N \protect \exp_not:N \numberline
126             { \exp_not:c { the #2 } }
127         }
128         ##3
129     }
130     \hook_use:n { #2 / begin }
131 }
132 \hook_new:n { #2 / begin }
133 }
134 }

```

(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
135 \NewTaggingSocket { talk / sec / title } { 1 }
136 \NewTaggingSocketPlug { talk / sec / title } { default }
137 { \exp_args:Ne \__talk_sect_tag:nn { \text_purify:v { l__talk_ #1 _tl } } {#1} }
138 \cs_new_protected:Npn \__talk_sect_tag:nn #1#2
139 {
140     \tag_struct_begin:e
141     {
142         tag =
143         \UseStructureName { sec / \use:c { toplevel@ #2 } / title } ,
144         title = {#1} ,
145         actualtext = {#1} ,
146     }
147     \tag_struct_end:
148 }
149 \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.

```

150 \cs_gset_protected:Npn \@starttoc #1
151 {
152     \begingroup

```



```

153 \makeatletter
154 \UseTaggingSocket { toc / starttoc / before } {#1}
155 \@input { \jobname .#1 }
156 \UseTaggingSocket { toc / starttoc / after } {#1}
157 \legacy_if:nT { @filesw }
158 {
159     \AddToHook { enddocument / afterlastpage }
160     {
161         \expandafter \newwrite \csname tf@ #1 \endcsname
162         \immediate \openout \csname tf@ #1 \endcsname \jobname .#1 \relax
163     }
164 }
165 \@nobreakfalse
166 \endgroup
167 }

```

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

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

```

168 \NewDocumentCommand \tableofcontents { 0 { } }
169 {
170     \group_begin:
171     \@starttoc { toc }
172     \group_end:
173 }

```

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

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

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

```

\l@subsubsection 174 \cs_new_protected:Npn \l@section #1#2
\__talk_toc_aux:nnnn 175 { \__talk_toc_aux:nnnn { 1 } { \bfseries \color { structure } } {#1} {#2} }
\__talk_toc_dest:n 176 \cs_new_protected:Npn \l@subsection #1#2
\__talk_toc_dest:w 177 {
\__talk_toc_level:nnnn 178     \__talk_toc_aux:nnnn
179     { 2 }
180     {
181         \skip_set:Nn \leftskip { 2em }
182         \color_ensure_current:
183     }
184     {#1} {#2}
185 }
186 \cs_new_protected:Npn \l@subsubsection #1#2
187 {
188     \__talk_toc_aux:nnnn
189     { 3 }
190     {
191         \skip_set:Nn \leftskip { 4em }
192         \color_ensure_current:
193         \footnotesize
194     }
195     {#1} {#2}
196 }
197 \cs_new_protected:Npn \__talk_toc_aux:nnnn #1#2#3#4

```

```

198 {
199   \int_compare:nNnTF { \value { section } } < 1
200     { \use:n }
201     { \__talk_toc_dest:n }
202       { \__talk_toc_level:nnnn {#1} {#2} {#3} {#4} }
203 }

```

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.

```

204 \cs_new_protected:Npn \__talk_toc_dest:n
205 {
206   \exp_after:wN \__talk_toc_dest:w \@contentsline@destination
207   . 0 . 0 . 0 . \q_stop
208 }
209 \cs_new_protected:Npn \__talk_toc_dest:w #1 . #2 . #3 . #4 . #5 \q_stop #6
210 {
211   \int_compare:nNnTF { \value { section } } = {#2}
212     {#6}
213     {
214       \opacity_begin:n { 0.2 }
215       #6
216       \opacity_end:
217     }
218 }
219 \cs_new_protected:Npn \__talk_toc_level:nnnn #1#2#3#4
220 {
221   \int_compare:nNnF {#1} > { \value { tocdepth } }
222     {
223       \group_begin:
224       \noindent
225       #2
226       \UseHookWithArguments { contentsline / text / before } { 4 }
227       {#1} {#3} {#4} { \@contentsline@destination }
228       #3
229       \UseHookWithArguments { contentsline / text / after } { 4 }
230       {#1} {#3} {#4} { \@contentsline@destination }
231       \UseHookWithArguments { contentsline / page / before } { 4 }
232       {#1} {#3} {#4}
233       { \@contentsline@destination }
234       \UseHookWithArguments { contentsline / page / after } { 4 }
235       {#1} {#3} {#4}
236       { \@contentsline@destination }
237       \par
238       \group_end:
239       \vfil
240     }
241 }

```

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

```

242 \setcounter { tocdepth } { 2 }

```

1.4 Block environments

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

```

    quote (env.) 243 \NewDocumentEnvironment { description } { } { } { }
  quotation (env.) 244 \NewDocumentEnvironment { quote } { } { } { }
    verse (env.) 245 \NewDocumentEnvironment { quotation } { } { } { }
  stdquote (env.) 246 \NewDocumentEnvironment { verse } { } { } { }
stdquotation (env.) 247 \AddToHook { begindocument / before }
  stdverse (env.) 248 {
249   \clist_map_inline:nn { quote , quotation , verse }
250   {
251     \NewEnvironmentCopy { std #1 } {#1}
252     \RenewDocumentEnvironment {#1} { D <> { all } !O { } }
253     {
254       \__talk_action_begin:n {##1}
255       \begin { std #1 } [ {##2} ]
256       \ignorespaces
257     }
258     {
259       \end { std #1 }
260       \__talk_action_end:
261     }
262   }
263 }
```

`block (env.)`

```

264 \NewDocumentEnvironment { block } { D <> { all } m }
265 {
266   \__talk_action_begin:n {#1}
267   \par
268   \vbox_set:Nw \l__talk_tmp_box
269   \group_begin:
270     \medskip
271     \leavevmode
272     \normalfont \large \bfseries
273     \color { structure }
274     #2
275     \par
276     \medskip
277   \group_end:
278 }
279 {
280   \vbox_set_end:
281   \box_use:N \l__talk_tmp_box
282   \par
283   \__talk_action_end:
284 }
```

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

```

\__talk_item_parse_spec:w
\__talk_item_parse_spec:n
```

close the structure before them first, so if we test too early, we'd end up covering then uncovering straight away!

```

285 \AddToHook { begindocument / before }
286 {
287   \NewCommandCopy \stditem \item
288   \RenewDocumentCommand \item { d <> o }
289   {
290     \IfNoValueTF {#2}
291     { \stditem }
292     { \stditem [ {#2} ] }
293     \IfNoValueTF {#1}
294     {
295       \exp_after:wN \__talk_item_parse_spec:w
296       \l__talk_action_spec_str < all > \q_stop
297     }
298     { \__talk_item_parse_spec:n {#1} }
299   }
300 }

```

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.

```

301 \cs_new_protected:Npn \__talk_item_parse_spec:w #1 < #2 > #3 \q_stop
302 { \__talk_item_parse_spec:n {#2} }
303 \cs_new_protected:Npn \__talk_item_parse_spec:n #1
304 {
305   \bool_lazy_or:nnF
306   { \tl_if_blank_p:n {#1} }
307   { \str_if_eq_p:nn {#1} { all } }
308   {
309     \tl_set:Nx \l__talk_list_end_tl
310     {
311       \exp_not:N \int_compare:nNnT \tex_currentgrouplevel:D =
312       { \int_use:N \tex_currentgrouplevel:D + 1 }
313       {
314         \__talk_action_end:
315         \tl_clear:N \exp_not:N \l__talk_list_end_tl
316       }
317     }
318     \__talk_action_begin:n {#1}
319   }
320 }

```

(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`

```

321 \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.

```

322 \cs_gset_protected:Npn \__block_inter_item:
323 {
324   \legacy_if:nT { @inlabel }
325   { \indent \par }
326   \mode_if_horizontal:T
327   {
328     \__block_skip_remove_last:
329     \__block_skip_remove_last:
330     \par
331   }
332   \l__talk_list_end_tl
333   \__kernel_list_item_end:
334   \__kernel_list_item_begin:
335   \addpenalty \@itempenalty
336   \addvspace \itemsep
337 }

```

A rather long block done by expansion to avoid duplication in a patch.

```

338 \IfFormatAtLeastTF { 2026-06-01 }
339 { \cs_gset_protected:Npe \BlockEnvEnd }
340 { \cs_gset:Npe \endblockenv }
341 {
342   \exp_not:n
343   { \__block_debug_typeout:n { blockenv~common~ending \on@line } }
344   \cs_if_exist:NTF \l__block_transparent_level_bool
345   {
346     \exp_not:N \bool_if:NF
347     \exp_not:N \l__block_transparent_level_bool
348   }
349   {
350     \exp_not:N \bool_if:NT
351     \exp_not:N \l__block_level_incr_bool
352   }
353   { \int_gdecr:N \exp_not:N \g_block_nesting_depth_int }
354   \exp_not:n
355   {
356     \legacy_if:nT { @inlabel }
357     {
358       \mode_leave_vertical:
359       \legacy_if_gset_false:n { @inlabel }
360     }
361     \__block_if_list:T
362     { \legacy_if:nT { @newlist } { \@noitemerr } }
363     \mode_if_horizontal:TF
364     {
365       \__block_skip_remove_last:
366       \__block_skip_remove_last:
367       \par
368     }
369     { \@inmatherr { \end { \@currenvir } } }
370     \l__talk_list_end_tl
371     \__kernel_displayblock_end:
372     \__block_if_list:T { \legacy_if_gset_false:n { @newlist } }
373     \legacy_if_gset_false:n { @nobreak }
374     \legacy_if:nF { @nolist }

```

```

375     {
376       \__block_skip_set_to_last:N \l_tmpa_skip
377       \dim_compare:nNnT \l_tmpa_skip > \c_zero_dim
378       {
379         \skip_vertical:n { - \l_tmpa_skip }
380         \skip_vertical:n { \l_tmpa_skip + \parskip - \@outerparskip }
381       }
382       \addpenalty \@endparpenalty
383       \addvspace \l__block_topsepadd_skip
384     }
385     \socket_use:n { block / endpe }
386   }
387 }

```

(End of definition for `__block_inter_item:`, `\BlockEnvEnd`, and `\endblockenv`. These functions are documented on page ??.)

`itemize (env.)` Allow for the classical beamer syntax: currently two versions but that will only last until the 2026-06-01 release of L^AT_EX is out.

```

description (env.) 388 \AddToHook { begindocument / before }
389 {
390   \clist_map_inline:nn { itemize , enumerate , description }
391   {
392     \IfFormatAtLeastTF { 2026-06-01 }
393     {
394       \RenewDocumentEnvironment {#1} { = { action-spec } !0 { } }
395       { \SimpleBlockEnv {#1} {##1} }
396       { \BlockEnvEnd }
397     }
398     {
399       \RenewDocumentEnvironment {#1} { = { action-spec } !o }
400       {
401         \IfNoValueTF {##1}
402         { \UseInstance { blockenv } {#1} { } }
403         { \UseInstance { blockenv } {#1} {##1} }
404       }
405       { \endblockenv }
406     }
407   }
408 }

```

And add the structural color to item labels.

```

409 \AddToHook { begindocument / before }
410 {
411   \EditInstance { item } { basic }
412   { label-format = \color { structure } #1 }
413   \EditInstance { item } { description }
414   { label-format = \normalfont \bfseries \color { structure } #1 }
415 }

```

`\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. Currently two versions but that will only last until the 2026-06-01 release of L^AT_EX is out.

```

416 \IfFormatAtLeastTF { 2026-06-01 }
417 {
418   \keys_define:nn { template / block / std }
419   { action-spec .str_set:N = \l__talk_action_spec_str }
420 }
421 {
422   \keys_define:nn { template / block / display }
423   { action-spec .str_set:N = \l__talk_action_spec_str }
424 }

```

(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.

```

425 \NewCommandCopy \stdnewtheorem \newtheorem
426 \RenewDocumentCommand \newtheorem { m O {#1} m o }
427 {
428   \IfNoValueTF {#4}
429   { \stdnewtheorem {#1} [ {#2} ] {#3} }
430   { \stdnewtheorem {#1} [ {#2} ] {#3} [ {#4} ] }
431   \NewEnvironmentCopy { std #1 } {#1}
432   \RenewDocumentEnvironment {#1} { D <> { all } o }
433   {
434     \__talk_action_begin:n {##1}
435     \IfNoValueTF {##2}
436     { \begin { std #1 } }
437     { \begin { std #1 } [ {##2} ] }
438     \ignorespaces
439   }
440   {
441     \end { std #1 }
442     \__talk_action_end:
443   }
444 }

```

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

```

445 </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	<u>131</u>
\\	<u>315</u>
A	
\abovecaptionskip	144, <u>146</u>
\action	<u>125</u>
actionenv (env.)	<u>125</u>
\addcontentsline	<u>120</u>
\addpenalty	335, <u>382</u>
\AddToHook	54, 60, 66, <u>159</u> , 218, 247, 265, 285, 330, 388, 400, 409
\addtolength	48
\addvspace	336, <u>383</u>
\alert	40, <u>103</u>
alertenv (env.)	<u>110</u>
\alt	<u>183</u>
\and	<u>131</u>
\arabic	<u>397</u>
\arraycolsep	25
\arrayrulewidth	27
\AssignSocketPlug	<u>182</u>
\AssignTaggingSocketPlug	<u>149</u>
\author	<u>15</u>
B	
\begin ...	116, 117, 130, 144, 255, 436, 437
\begingroup	151, 152, <u>377</u>
\belowcaptionskip	145, <u>147</u>
\bfseries	21, 39, 175, 272, <u>414</u>
\bigskipamount	18
block (env.)	<u>264</u>
block commands:	
\g_block_nesting_depth_int	353
block internal commands:	
__block_debug_typeout:n	343
__block_if_list:TF	361, <u>372</u>
__block_inter_item:	322, <u>322</u>
\l_block_level_incr_bool	351
__block_list_end:	57
__block_list_item_end:	57
__block_skip_remove_last:	328, 329, 365, <u>366</u>
__block_skip_set_to_last:N	376
\l_block_topsepadd_skip	383
\l_block_transparent_level_bool	344, <u>347</u>
\BlockEnvEnd	<u>322</u> , 396
bool commands:	
\bool_do_while:Nn	27
\bool_gset_false:N	30, 36, 40, <u>422</u>
\bool_gset_true:N	207, 215, 221, <u>414</u>
\bool_if:NTF	6, 39, 44, 44, 46, 50, 58, 66, 71, 83, 86, 95, 141, 157, 170, 172, 211, 256, 346, 350, 428
\bool_lazy_and:nnTF	21, 49, 218, 303
\bool_lazy_any:nTF	64, 91
\bool_lazy_or:nnTF	5, 18, 21, 305
\bool_lazy_or_p:nn	24
\bool_new:N	3, 3, 7, 8, 13, 121, 123, 124, 391, 392, 393
\bool_set_eq:NN	149, 153
\bool_set_false:N	27, 28, 37, 101, 120, 142, 434, 454
\bool_set_true:N	24, 29, 52, 69, 140, 148, 185, 204, 217, 407, 442, 462
\c_false_bool	15
\c_true_bool	14, 166
box commands:	
\box_dp:N	36
\box_gclear:N	77
\box_ht:N	63, 261, 286
\box_if_empty:NTF	92
\box_move_down:nn	61
\box_new:N	4, 115, 159
\box_use:N	281
\box_use_drop:N	26, 239, 266, 270, 272, 326
\box_wd:N	41
box internal commands:	
__box_dim_eval:n	33, 36, 41, 44
__box_set_to_wd:	40, 45
C	
\clearpage	98
clist commands:	
\clist_const:Nn	58
\clist_if_in:NnTF	65, 184
\clist_map_break:	222
\clist_map_inline:Nn	132, 187, 313
\clist_map_inline:nn	3, 89, 126, 139, 249, 390
\clist_new:N	10, 14
\clist_pop:NNTF	309
\clist_set:Nn	104, 183

<code>\color</code>	4, 11 , 56, 175, 273, 412, 414
color commands:	
<code>\color_ensure_current:</code> . .	63, 182, 192
<code>\color_group_begin:</code>	34, 46
<code>\color_group_end:</code>	34
<code>\color_math:nn</code>	9, 26
<code>\color_math:nnn</code>	10, 27
<code>\color_select:n</code>	7, 16, 35, 37, 45, 46, 69, 107, 210, 254, 307, 348
<code>\color_select:nn</code>	8, 17, 38
color internal commands:	
<code>__color_backend_reset:</code>	64
<code>\colorlet</code>	68
<code>column</code> (env.)	72
<code>columns</code> (env.)	11
<code>\columnwidth</code>	20, 81, 116
cs commands:	
<code>\cs_generate_variant:Nn</code>	7, 8, 9, 10, 104, 105, 107, 108, 109, 110, 111, 112, 113, 181
<code>\cs_gset:Npe</code>	340
<code>\cs_gset:Npn</code>	78, 79, 80
<code>\cs_gset_eq:NN</code>	56
<code>\cs_gset_nopar:Npn</code>	49, 57, 64
<code>\cs_gset_protected:Npe</code> . . .	62, 89, 339
<code>\cs_gset_protected:Npn</code>	29, 38, 48, 56, 58, 148, 150, 310, 322
<code>\cs_if_exist:NTF</code>	118, 256, 344
<code>\cs_if_exist_p:N</code>	304
<code>\cs_if_exist_use:NTF</code> . . .	139, 159, 174
<code>\cs_new:Npn</code> 6, 7, 34, 35, 36, 37, 38, 39, 40, 41, 42, 219, 245, 336, 396, 397, 402	
<code>\cs_new_eq:NN</code>	5, 6, 147, 183, 395
<code>\cs_new_nopar:Npn</code>	3, 379
<code>\cs_new_protected:Npe</code>	63, 78, 104
<code>\cs_new_protected:Npn</code>	9, 11, 16, 18, 18, 35, 38, 40, 42, 44, 45, 48, 51, 52, 53, 54, 55, 56, 56, 62, 64, 69, 70, 72, 81, 82, 87, 91, 100, 101, 106, 111, 112, 114, 118, 121, 131, 136, 136, 137, 138, 142, 146, 146, 151, 153, 164, 168, 174, 174, 175, 176, 181, 182, 186, 191, 194, 197, 201, 204, 209, 219, 226, 275, 301, 301, 303, 342, 343, 375, 404, 410, 418
<code>\cs_set:Npn</code>	14, 15, 59
<code>\cs_set_eq:NN</code>	61, 62, 63, 64, 216, 361, 362, 376, 377, 387, 388
<code>\cs_set_nopar:Npn</code>	118, 354, 356, 360, 364, 366, 371, 381, 386
<code>\cs_set_protected:Npn</code>	40, 131, 148, 166, 210
<code>\cs_to_str:N</code>	333, 338
<code>\csname</code>	155, 161, 162
D	
<code>\date</code>	15
<code>\day</code>	20
<code>\DeclareColor</code>	65 , 71, 72, 73
<code>\DeclareInstance</code>	43, 79, 80, 81, 83, 85, 87, 120, 217, 218, 219, 220, 221, 222, 223, 264, 329
<code>\DeclareInstanceCopy</code>	267, 332
<code>\DeclareTemplateCode</code>	23, 54, 77, 104, 109, 197, 235, 282
<code>\DeclareTemplateInterface</code>	16, 45, 75, 95, 99, 190, 225, 272
<code>\definecolor</code>	69
<code>description</code> (env.)	243 , 388
dim commands:	
<code>\dim_compare:nNnTF</code>	108, 259, 377
<code>\dim_compare_p:nNn</code>	109, 306
<code>\dim_const:Nn</code>	163, 169
<code>\dim_eval:n</code>	51, 52, 53
<code>\dim_max:nn</code>	110, 285
<code>\dim_set:Nn</code>	19, 80
<code>\dim_set_eq:NN</code>	20, 81
<code>\dim_to_decimal:n</code>	156
<code>\dim_use:N</code>	184, 185
<code>\c_zero_dim</code>	377
<code>\DocumentMetadata</code>	8
<code>\doublerulesep</code>	28
E	
<code>\EditInstance</code>	268, 333, 411, 413
<code>\emph</code>	318
<code>\end</code>	123, 124, 137, 146, 259, 369, 441
<code>\endblockenv</code>	322 , 405
<code>\endcsname</code>	155, 161, 162
<code>\endfloatenv</code>	121 , 134
<code>\endgroup</code>	157, 166, 390
<code>enumerate</code> (env.)	388
environments:	
<code>actionenv</code>	125
<code>alertenv</code>	110
<code>block</code>	264
<code>column</code>	72
<code>columns</code>	11
<code>description</code>	243 , 388
<code>enumerate</code>	388
<code>figure</code>	126
<code>invisibleenv</code>	98
<code>itemize</code>	388
<code>onlyenv</code>	118
<code>overlayarea</code>	233
<code>overprint</code>	250
<code>quotation</code>	243
<code>quote</code>	243
<code>stdquotation</code>	243

stdquote	243		
stdverse	243		
table	126		
uncoverenv	98		
verse	243		
visibleenv	98		
exp commands:			
\exp_after:wN	206, 295		
\exp_args:Ne	17, 54, 137, 223		
\exp_args:Nne	444		
\exp_args:No	32		
\exp_args:NV	129, 176		
\exp_args_generate:n	106		
\exp_not:N	65, 67, 68, 69, 71, 72, 72, 73, 78, 80, 81, 84, 86, 87, 89, 89, 90, 92, 94, 95, 96, 98, 100, 101, 101, 103, 106, 107, 110, 111, 112, 114, 117, 119, 120, 122, 123, 125, 126, 142, 161, 166, 167, 169, 171, 172, 175, 176, 180, 292, 304, 311, 315, 337, 338, 339, 346, 347, 350, 351, 353, 360		
\exp_not:n	294, 342, 354, 362, 363, 445		
\exp_stop_f:	51, 52, 53		
\expandafter	161		
\ExpandArgs	91, 98, 134, 258, 311, 318, 333, 334		
		F	
\fboxrule	32		
\fboxsep	31		
\fi	18, 153		
figure (env.)	126		
\figurename	137		
file commands:			
\file_if_exist_input:nTF	156		
\file_input:n	158		
fnote commands:			
\fnote_makefn_text:n	183		
\footins	30		
\footnote	161, 162		
\footnoterule	94		
\footnotesize	193		
\footskip	296, 297		
fp commands:			
\fp_eval:n	161		
\fp_to_dim:n	171		
\frame	28, 26, 403		
frame	428		
frame*	428		
\framesubtitle	10		
\frametitle	5, 435, 445		
		G	
\gdef	292		
\geometry	5		
group commands:			
\group_begin:	11, 33, 35, 58, 60, 67, 105, 127, 138, 170, 208, 223, 252, 269, 347		
\c_group_begin_token	43		
\group_end:	40, 40, 54, 62, 63, 75, 108, 131, 172, 181, 213, 238, 262, 277, 351		
\group_insert_after:N	45		
		H	
hbox commands:			
\hbox:n	59		
\hbox_set_end:	25		
\hbox_set_to_wd:Nnw	18		
\headsep	230, 249, 250		
\hfil	77, 280, 324, 358, 369, 374, 384		
hook commands:			
\hook_gput_code:nnn	105, 160, 298		
\hook_new:n	132		
\hook_use:n	130		
\hspace	207, 214		
\hypersetup	221		
		I	
\IfBooleanT	361		
\ifcase	5		
\IfFormatAtLeastF	7		
\IfFormatAtLeastTF	338, 392, 416		
\IfNoValueF	362, 363		
\IfNoValueTF	15, 25, 36, 47, 67, 165, 220, 290, 293, 401, 428, 435		
\ignorespaces	19, 21, 86, 155, 192, 231, 256, 438		
\immediate	162		
\includegraphics	352		
\indent	325		
\insertsection	81		
\insertsubsection	81		
\insertsubsubsection	81		
\institute	34		
int commands:			
\int_compare:nNnTF	76, 122, 199, 203, 206, 211, 212, 214, 221, 311		
\int_compare_p:nNn	219, 220		
\int_eval:n	223		
\int_gdecr:N	353		
\int_gincr:N	29, 42, 75, 221, 406		
\int_gset:Nn	222, 413		
\int_gset_eq:NN	27, 135, 140, 145		
\int_gzero:N	16, 25, 74		
\int_incr:N	254		

<code>\int_max:nn</code>	178	<code>\leftmarginii</code>	44, 59
<code>\int_new:N</code>		<code>\leftmarginiii</code>	45, 66
... 4, 5, 9, 10, 71, 134, 151, 244, 394		<code>\leftskip</code>	181, 191
<code>\int_set_eq:NN</code>	15	legacy commands:	
<code>\int_to_Roman:n</code>	247	<code>\legacy_if:nTF</code>	
<code>\int_to_roman:n</code>	248	. 41, 157, 218, 288, 324, 356, 362, 374	
<code>\int_use:N</code> .. 8, 14, 52, 56, 68, 312, 399		<code>\legacy_if_gset_false:n</code> 359, 372, 373	
<code>\c_max_int</code>	197, 220		
<code>\invisible</code>	89		
<code>invisibleenv (env.)</code>	98		
iow commands:			
<code>\iow_now:Nn</code>	290		
<code>\item</code>	59, 285		
<code>itemize (env.)</code>	388		
<code>\itemsep</code>	54, 62, 69, 336		
	J		
<code>\jobname</code>	155, 162		
	K		
kernel internal commands:			
<code>__kernel_backend_literal_pdf:n</code> . 77			
<code>__kernel_color_backend_stack_-</code>			
<code>push:nn</code>	79		
<code>__kernel_displayblock_end:</code>	371		
<code>__kernel_list_item_begin:</code>	334		
<code>__kernel_list_item_end:</code>	333		
keys commands:			
<code>\l_keys_choice_tl</code>	130		
<code>\keys_define:nn</code>	3,		
5, 6, 31, 117, 132, 141, 155, 418, 422			
<code>\keys_set:nn</code> .. 7, 17, 17, 24, 29, 36,			
41, 49, 78, 148, 168, 433, 441, 453, 461			
<code>\keys_set_known:nn</code>	20, 32		
<code>\l_keys_value_tl</code>	46, 165		
	L		
<code>\label</code>	368		
<code>\labelenumi</code>	34		
<code>\labelenumii</code>	35		
<code>\labelenumiii</code>	36		
<code>\labelenumiv</code>	37		
<code>\labelitemfont</code>	38, 39, 40, 41, 42		
<code>\labelitemi</code>	38		
<code>\labelitemii</code>	39		
<code>\labelitemiii</code>	40		
<code>\labelitemiv</code>	41		
<code>\labelsep</code>	29, 33, 46, 48		
<code>\labelwidth</code>	47, 48		
<code>\Large</code>	21, 90		
<code>\large</code>	272		
<code>\leavevmode</code>	83, 271		
<code>\leftmargin</code>	51, 59, 66		
<code>\leftmargini</code>	43, 47, 51		
		M	
		<code>\makeatletter</code>	153
		<code>\maketitle</code>	139
		<code>\mathcolor</code>	5, 11
		<code>\mbox</code>	346
		<code>\medskip</code>	270, 276
		<code>\mode</code>	36, 11
		mode commands:	
		<code>\mode_if_horizontal:TF</code> .. 31, 326, 363	
		<code>\mode_if_horizontal_p:</code>	25
		<code>\mode_if_math:TF</code>	345
		<code>\mode_if_vertical_p:</code>	26
		<code>\mode_leave_vertical:</code> ... 34, 348, 358	
		<code>\month</code>	5
		msg commands:	
		<code>\msg_error:nnn</code>	126, 168
		<code>\msg_fatal:nn</code>	15
		<code>\g_msg_module_name_prop</code>	5
		<code>\g_msg_module_type_prop</code>	6
		<code>\msg_new:nnn</code>	22, 313
		<code>\msg_new:nnnn</code>	9, 226
		<code>\msg_warning:nn</code>	27, 309
		N	
		<code>\NeedsDocumentMetadata</code>	17
		<code>\NewCommandCopy</code>	
		. 4, 5, 6, 161, 287, 333, 353, 403, 425	
		<code>\newcounter</code>	21, 72, 73, 74, 137
		<code>\NewDocumentCommand</code>	5,
		10, 11, 34, 39, 65, 87, 91, 92, 103,	
		113, 125, 139, 168, 183, 189, 206, 216	
		<code>\NewDocumentEnvironment</code>	11,
		72, 98, 110, 118, 128, 133, 233,	
		243, 244, 245, 246, 250, 264, 438, 458	
		<code>\NewEnvironmentCopy</code>	251, 431
		<code>\newlabel</code>	381
		<code>\newlength</code>	144, 145
		<code>\NewSocketPlug</code>	169
		<code>\NewTaggingSocket</code>	135
		<code>\NewTaggingSocketPlug</code>	136
		<code>\NewTemplateType</code>	
		. 15, 44, 74, 94, 98, 189, 224, 271	
		<code>\newtheorem</code>	425
		<code>\newwrite</code>	161
		<code>\nobreakspace</code>	142
		<code>\noindent</code>	31, 224, 246, 293

<code>\normalfont</code>	42, 50, 229, 272, 414	property commands:	
<code>\normalsize</code>	154	<code>\property_new:nnnn</code>	8, 398
<code>\number</code>	20, 22	<code>\property_record:nn</code>	52, 68, 401
<code>\numberline</code>	125	<code>\property_ref:nn</code>	14, 402
O			
<code>\obeyedline</code>	18, 59	<code>\protect</code>	125
<code>\only</code>	44, 113	<code>\ProvidesExplClass</code>	3
<code>onlyenv (env.)</code>	118	<code>\put</code>	57
<code>\onslide</code>	41, 189	Q	
opacity commands:		<code>\quad</code>	131
<code>\opacity_begin:n</code>	39, 56, 214	quark commands:	
<code>\opacity_end:</code>	41, 58, 216	<code>\quark_if_recursion_tail_stop:N</code>	138
opacity internal commands:		<code>\quark_if_recursion_tail_stop_-</code>	
<code>__opacity_backend:nnn</code>	85, 86	<code>do:Nn</code>	155, 166
<code>__opacity_backend_begin:n</code> ..	57, 62	<code>\quark_if_recursion_tail_stop_-</code>	
<code>__opacity_backend_end:</code>	59, 89	<code>do:nn</code>	47
<code>\l__opacity_backend_fill_tl</code>	71	<code>\q_recursion_stop</code>	38, 133, 158
<code>__opacity_backend_reset:</code>	97	<code>\q_recursion_tail</code>	38, 133, 158
<code>__opacity_backend_reset_fill:</code> ..	99	<code>\q_stop</code>	73, 81, 107, 112,
<code>__opacity_backend_reset_stroke:</code>	100		167, 176, 188, 191, 207, 209, 296, 301
<code>\c__opacity_backend_stack_int</code> ...	80	quotation (env.)	243
<code>\l__opacity_backend_stroke_tl</code> ...	72	quote (env.)	243
<code>__opacity_select:nN</code>	57	R	
<code>\openout</code>	162	<code>\raggedright</code>	84, 197, 230
<code>\or</code> ..	5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16	<code>\refstepcounter</code>	103
<code>overlayarea (env.)</code>	233	<code>\relax</code>	162
<code>overprint (env.)</code>	250	<code>\relsize</code>	161
P			
<code>\pagecolor</code>	43	<code>\RenewCommandCopy</code>	26
<code>\pagestyle</code>	390	<code>\RenewDocumentCommand</code> ..	11, 15, 21, 22,
<code>\paperheight</code>	57, 58		27, 30, 43, 162, 288, 334, 354, 368, 426
<code>\paperwidth</code>	58, 301, 349, 350	<code>\RenewDocumentEnvironment</code>	
<code>\par</code>	30, 32, 14, 28, 34, 39, 73,		252, 394, 399, 430, 432, 450
	74, 89, 94, 150, 156, 228, 237, 242,	<code>\RequirePackage</code>	3, 159, 180, 198,
	267, 273, 275, 282, 325, 330, 367, 424		201, 202, 205, 208, 214, 215, 220, 352
<code>\parsep</code>	53, 61, 62, 68, 69	<code>\rmdefault</code>	216
<code>\parskip</code>	380	<code>\rule</code>	58
<code>\partopsep</code>	71	rule commands:	
<code>\pause</code>	41, 216	<code>\rule:nnn</code>	48, 349
pdfmanagement commands:		S	
<code>\pdfmanagement_add:nnn</code>	73	scan commands:	
prg commands:		<code>\scan_stop:</code>	54
<code>\prg_generate_conditional_-</code>		<code>\scriptsize</code>	84
<code>variant:Nnn</code>	10	<code>\section</code>	72, 81
<code>\prg_new_protected_conditional:Npnn</code>		seq commands:	
	3, 3	<code>\seq_gpop_left:NN</code>	174
<code>\prg_return_false:</code>	8, 9	<code>\seq_gpush:Nn</code>	164
<code>\prg_return_true:</code>	7, 8	<code>\seq_gput_right:Nn</code>	152
<code>\ProcessedArgument</code>	14, 15	<code>\seq_gremove_all:Nn</code>	75, 76, 77
<code>\ProcessKeyOptions</code>	155	<code>\seq_map_indexed_inline:Nn</code>	83
prop commands:		<code>\seq_map_inline:Nn</code>	132, 139, 144
<code>\prop_gput:Nnn</code>	5, 6	<code>\seq_new:N</code>	124, 160
		<code>\seq_set_from_clist:Nn</code>	81, 125

\l__talk_aspect_ratio_str .	117 , 175	__talk_decode_parse_auxii:n ...	16 , 32 , 35
\l__talk_cnt_reset_seq		\l__talk_decode_pure_bool	7 , 29 , 51 , 101 , 120
.....	75 , 76 , 77 , 124 , 139 , 144 , 152	\l__talk_decode_step_bool	8 , 37 , 39 , 148
__talk_cnt_restore:	87 , 137 , 142	\l__talk_float_alignment_tl	97 , 109 , 110 , 111 , 117 , 123
__talk_cnt_save:	78 , 137 , 137	\l__talk_fontsize_dim ..	117 , 156 , 161
__talk_column_align_bottom:n	53 , 53	\l__talk_footelem_color_tl	189
__talk_column_align_center:n	53 , 55	\l__talk_footelem_font_tl	189
__talk_column_align_top:n ..	53 , 70	\l__talk_footelem_left_skip	189
\l__talk_column_alignment_tl .	31 , 92	\l__talk_footelem_right_skip ...	189
\g__talk_column_int 9 , 15 , 16 , 27 , 75 , 76		\l__talk_footer_bg_tl	271
\l__talk_column_int	9 , 15 , 27	\l__talk_footer_fg_tl	271
\l__talk_columns_wd_tl	5 , 18 , 19	\l__talk_footer_font_tl	271
__talk_decode_action:n .	95 , 104 , 104	\l__talk_footer_left_skip	271
__talk_decode_action:w	104 , 106 , 111	\l__talk_footer_order_clist	271
\l__talk_decode_action_str	12 , 20 , 121 , 152 , 160	\l__talk_footer_right_skip	271
\l__talk_decode_actions_bool ...	13 , 27 , 154 , 162	\l__talk_footer_sep_tl	271
\l__talk_decode_actions_clist ...	13	\g__talk_footnote_box	77 , 92 , 95 , 159 , 171 , 173
\l__talk_decode_actions_str ..	13 , 31	\g__talk_footnote_overlay_seq ..	160 , 164 , 174
\l__talk_decode_arg_str	9 , 26 , 32 , 127 , 169	\l__talk_frame_alignment_tl	90 , 94 , 154 , 164
__talk_decode_check:n .	134 , 181 , 181	\l__talk_frame_bool .	95 , 141 , 391 , 407
__talk_decode_check:nw	181 , 188 , 191	\g__talk_frame_int	14 , 52 , 68 , 247 , 394 , 399 , 406
__talk_decode_check_range:nnn .	181 , 197 , 198 , 210	__talk_frame_notag:n ...	41 , 418 , 418
__talk_decode_check_single:nn .	181 , 194 , 201	__talk_frame_overprint:	245 , 245 , 257 , 260 , 264 , 277 , 279 , 280 , 283 , 285 , 292 , 294 , 299
__talk_decode_mode:n	54 , 63 , 63	__talk_frame_process:nn	404 , 404 , 435 , 444 , 455 , 463
__talk_decode_mode:nn ...	86 , 89 , 91	\g__talk_frame_struct_int	56 , 71 , 413
__talk_decode_mode:w	63 , 72 , 78	\g__talk_frame_subtitle_tl	3 , 13 , 76
__talk_decode_mode_aux:n	63	__talk_frame_tag:n	37 , 410 , 410
__talk_decode_overlay_:nw	131	\g__talk_frame_tag_bool	46 , 392 , 414 , 422
__talk_decode_overlay_aux:nNn .	131 , 149 , 152 , 153	\l__talk_frame_tagging_str	17 , 18 , 20 , 22 , 34 , 155
__talk_decode_overlay_offset:nNn	131 , 157 , 162 , 172 , 175	__talk_frame_title:n	15 , 38 , 44
__talk_decode_overlay_offset:nNnN	131 , 161 , 164 , 173	\l__talk_frame_title_bool .	117 , 428
__talk_decode_overlays:nN	131 , 133 , 136 , 142 , 179	__talk_frame_title_tagged:n ...	15 , 47 , 51
__talk_decode_overlays:nn	97 , 116 , 123 , 131 , 131	\g__talk_frame_title_tl	3 , 8 , 58 , 75 , 260 , 443
\l__talk_decode_overlays_bool .	3 , 6 , 24 , 28 , 52 , 69 , 150 , 157 , 177 , 179	\l__talk_frame_verb_bool	44 , 393 , 434 , 442 , 454 , 462
\l__talk_decode_overlays_clist ..	10	\l__talk_frametitle_after_skip .	25 , 41
\l__talk_decode_overlays_str ...	10 , 30 , 50 , 96	\l__talk_frametitle_before_skip	26 , 32
__talk_decode_parse:n	5 , 16 , 16 , 148 , 176		
__talk_decode_parse:w .	16 , 38 , 45 , 56		
__talk_decode_parse_auxi:n	16 , 17 , 18		

<code>\l__talk_frametitle_color_tl ...</code>	<code>\l__talk_saved_action_str</code>
..... 27, 34, 35 121, 151, 175
<code>\l__talk_frametitle_font_tl .. 28, 36</code>	<code>\l__talk_saved_actions_bool</code>
<code>\l__talk_header_bg_tl</code> 121, 153, 177
<code>\l__talk_header_fg_tl</code>	<code>\l__talk_saved_overlays_bool ...</code>
<code>\l__talk_header_font_tl</code> 121, 149, 172
<code>\l__talk_header_frametitle_bool 224</code>	<code>__talk_sect_section:Nnn</code>
<code>\l__talk_header_ht_dim</code> 81
<code>\l__talk_header_left_skip</code>	<code>__talk_sect_subsection:Nnn</code>
<code>\l__talk_header_right_skip</code> 81
<code>__talk_header_tag_begin:n</code>	<code>__talk_sect_tag:nn</code>
..... 53, 174, 174, 181 135, 137, 138
<code>__talk_header_tag_end: . 64, 174, 182</code>	<code>\g__talk_section_tl</code>
<code>__talk_if_overlay:n</code> 66
..... 3, 10	<code>\l__talk_section_tl</code>
<code>__talk_if_overlay:nTF</code> 66
..... 3,	<code>\l__talk_shuffle_skip .. 17, 20, 22, 34</code>
7, 12, 13, 13, 23, 31, 32, 34, 45,	<code>__talk_shuffle_skip:n . 18, 18, 39, 41</code>
97, 115, 185, 198, 208, 337, 356, 371	<code>__talk_slide:nn</code>
<code>__talk_item_parse_spec:n</code> 9, 9, 408
..... 285, 298, 302, 303	<code>__talk_slide_align_bottom:n 100, 100</code>
<code>__talk_item_parse_spec:w</code>	<code>__talk_slide_align_center:n 100, 106</code>
..... 285, 295, 301	<code>__talk_slide_align_stretch:n ..</code>
<code>__talk_label:n</code> 100, 112
..... 368, 372, 375	<code>__talk_slide_align_top:n . 100, 118</code>
<code>__talk_latex_frame:n .. 26, 403, 403</code>	<code>__talk_slide_aux:n</code>
<code>\l__talk_list_end_tl</code> 9, 45, 56
..... 309, 315, 321, 332, 370	<code>__talk_slide_begin:</code>
<code>__talk_metadata_name:n</code> 33, 72, 72
..... 312, 315, 320, 336, 336	<code>\l__talk_slide_box</code>
<code>__talk_mode:n</code> 4, 79, 91
..... 3	<code>\g__talk_slide_continue_bool .. 3,</code>
<code>__talk_mode:nTF</code>	27, 30, 36, 40, 86, 207, 211, 215, 221
..... 3, 12	<code>__talk_slide_end:</code>
<code>\l__talk_mode_str</code>	49, 72, 82
..... 7, 68, 93, 117	<code>\g__talk_slide_int</code>
<code>\c__talk_modes_clist</code> 5, 8, 25, 29, 203, 206, 212, 214, 219
..... 58, 65	<code>\g__talk_subsection_tl</code>
<code>__talk_onslide:n . 189, 191, 194, 223</code> 66
<code>\g__talk_onslide_tl</code>	<code>\l__talk_subsection_tl</code>
..... 80, 84, 155, 196, 197, 201, 205 66, 117
<code>__talk_opacity_begin:n</code>	<code>\g__talk_subsubsection_tl</code>
..... 38, 38, 51, 52, 59, 60, 79, 84, 200 66
<code>__talk_opacity_end:</code>	<code>\l__talk_subsubsection_tl .. 66, 119</code>
..... 38, 40, 55, 63, 88, 179, 202	<code>__talk_textcmd_equiv:n . 318, 339, 343</code>
<code>\l__talk_overlay_all_bool</code>	<code>\l__talk_titlelem_after_skip 44</code>
..... 124, 140, 142, 170	<code>\l__talk_titlelem_before_skip ... 44</code>
<code>__talk_overlay_arg:n</code>	<code>\l__talk_titlelem_color_tl</code>
..... 3, 11, 92, 99, 103, 110, 118 44
<code>__talk_overprint_begin:n</code>	<code>\l__talk_titlelem_font_tl</code>
..... 226, 226, 234, 251 44
<code>__talk_overprint_check_ht:n ...</code>	<code>\l__talk_titlelem_tag_begin_tl .. 44</code>
..... 250, 299, 301, 310	<code>\l__talk_titlelem_tag_end_tl 44</code>
<code>\l__talk_overprint_int . 244, 248, 254</code>	<code>\l__talk_titlepage_alignment_tl . 94</code>
<code>__talk_overprint_save_ht:</code>	<code>\l__talk_titlepage_framestyle_tl 94</code>
..... 250, 255, 275	<code>\l__talk_titlepage_order_clist .. 94</code>
<code>__talk_pagecolor:n 43, 48, 49, 52</code>	<code>__talk_tmp:w</code>
<code>\c__talk_paper_height_dim</code>	114, 114, 166, 175
..... 163	<code>\l__talk_tmp_box</code>
<code>\c__talk_paper_width_dim</code>	18, 26,
..... 163	63, 67, 79, 93, 115, 229, 239, 261,
<code>\g__talk_pauses_int</code>	266, 268, 270, 272, 281, 286, 299, 326
..... 11, 4, 42, 74, 178, 221, 222, 223	<code>\l__talk_tmp_tl</code>
	12, 18,
	21, 23, 106, 116, 175, 176, 309, 311, 312
	<code>__talk_toc_aux:nmmn</code>
 174, 175, 178, 188, 197
	<code>__talk_toc_dest:n 174, 201, 204</code>
	<code>__talk_toc_dest:w 174, 206, 209</code>
	<code>__talk_toc_level:nmmn . 174, 202, 219</code>
	<code>\l__talk_uncover_hidden_fp</code>
	74

