

# ltx-talk – A class for typesetting presentations\*

Joseph Wright<sup>†</sup>

Released 2026-01-16

## Contents

<b>I</b>	<b>ltx-talk – Overall set up</b>	<b>1</b>
<b>1</b>	<b>ltx-talk implementation</b>	<b>1</b>
1.1	Set up . . . . .	1
1.2	Additions for expl3 . . . . .	1
1.3	Extra variants . . . . .	3
1.4	Scratch space . . . . .	3
1.5	Option handling . . . . .	3
1.6	Setting up . . . . .	4
1.7	Math support . . . . .	5
1.8	Font selection . . . . .	5
1.9	Hyperlinks . . . . .	6
1.10	Tagging . . . . .	6
<b>II</b>	<b>ltx-talk-color – Color definitions</b>	<b>7</b>
<b>1</b>	<b>ltx-talk-color implementation</b>	<b>7</b>
1.1	Existing definitions . . . . .	7
1.2	Document (and interface) commands . . . . .	7
1.3	Color definition . . . . .	9
1.4	Semantic colors . . . . .	9
<b>III</b>	<b>ltx-talk-decode – Decoding overlay specs</b>	<b>10</b>
<b>1</b>	<b>ltx-talk-decode implementation</b>	<b>10</b>
<b>IV</b>	<b>ltx-talk-frame – The structure of frames</b>	<b>17</b>

---

\*This file describes v0.3.11, last revised 2026-01-16.

<sup>†</sup>E-mail: [joseph@texdev.net](mailto:joseph@texdev.net)

<b>1</b>	<b>ltx-talk-frame implementation</b>	<b>17</b>
1.1	Slides in frames . . . . .	17
1.2	Counters . . . . .	20
1.3	Frame options . . . . .	21
1.4	Tagging for headers . . . . .	21
1.5	Wallpaper . . . . .	22
1.6	The <code>frame</code> environment . . . . .	26
<b>V</b>	<b>ltx-talk-frame – The structure of frames</b>	<b>29</b>
<b>1</b>	<b>ltx-talk-frame-structure implementation</b>	<b>29</b>
1.1	Columns . . . . .	29
1.2	Floats . . . . .	31
1.3	Footnotes . . . . .	33
<b>VI</b>	<b>ltx-talk-mode – Modes</b>	<b>34</b>
<b>1</b>	<b>ltx-talk-mode implementation</b>	<b>34</b>
<b>VII</b>	<b>ltx-talk-overlay – Overlays</b>	<b>35</b>
<b>1</b>	<b>ltx-talk-overlay implementation</b>	<b>35</b>
1.1	Utilities . . . . .	35
1.2	Opacity utilities . . . . .	36
1.3	Action commands and environments . . . . .	37
1.4	Non-action commands and environments . . . . .	40
1.5	Fixed-size areas . . . . .	42
1.6	Adding overlays to existing commands . . . . .	44
<b>VIII</b>	<b>ltx-talk-required – “Required” definitions</b>	<b>47</b>
<b>1</b>	<b>ltx-talk-required implementation</b>	<b>47</b>
1.1	Standard design settings . . . . .	47
1.2	List support . . . . .	48
<b>IX</b>	<b>ltx-talk-structure – Structural commands</b>	<b>49</b>
<b>1</b>	<b>ltx-talk-structure implementation</b>	<b>49</b>
1.1	Frame title . . . . .	49
1.2	Sectioning . . . . .	50
1.3	Table of contents . . . . .	52
1.4	Block environments . . . . .	54
1.5	Lists . . . . .	55
1.6	Theorems, <i>etc.</i> . . . . .	58

<b>X</b>	<b>ltx-talk-title – Title pages</b>	<b>59</b>
<b>1</b>	<b>ltx-talk-title implementation</b>	<b>59</b>
	<b>Index</b>	<b>63</b>

## 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} {2026-01-16} {0.3.11}
4 {A class for typesetting presentations}
    Get the right type of message.
5 \prop_gput:Nnn \g_msg_module_name_prop { talk } { ltx-talk }
6 \prop_gput:Nnn \g_msg_module_type_prop { talk } { Class }
    Require the latest LATEX structures.
7 \IfFormatAtLeastF{2025-11-01}
8 {
9     \msg_new:nnnn { ltx-talk } { kernel-too-old }
10     { The~ltx-talk~class~requires~LaTeX~2025-11-01~or~later. }
11     {
12         You~have~tried~to~use~the~ltx-talk~class~with~a~LaTeX~kernel~release~
13         prior~to~2025-11-01;~the~required~functionality~is~missing.
14     }
15     \msg_fatal:nn { ltx-talk } { kernel-too-old }
16 }
17 \NeedsDocumentMetadata
```

### 1.2 Additions for expl3

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

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

```

33         \tex_hsize:D \_box_dim_eval:n {#2}
34         \group_insert_after:N \_box_set_to_wd:
35         \color_group_begin:
36     }

```

Some things from xbox that would be useful.

```

37 \cs_gset_protected:Npn \rule:nnn #1#2#3
38 {
39     \tex_vrule:D
40     height \dim_eval:n {#2} \exp_stop_f:
41     depth  \dim_eval:n {#3} \exp_stop_f:
42     width  \dim_eval:n {#1} \exp_stop_f:
43     \scan_stop:
44 }

```

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

```

45 \cs_gset_protected:Npn \opacity_begin:n #1
46 { \_opacity_select:nN {#1} \_opacity_backend_begin:n }
47 \cs_gset_protected:Npn \opacity_end:
48 { \_opacity_backend_end: }
49 \AddToHook { begindocument }
50 {
51     \cs_gset_protected:Npe \_opacity_backend_begin:n #1
52     {
53         \bool_lazy_any:nTF
54         {
55             { \sys_if_engine_pdftex_p: }
56             { \sys_if_engine luatex_p: }
57             { \sys_if_engine_xetex_p: }
58         }
59         {
60             \tl_set:Nn \exp_not:N \l__opacity_backend_fill_tl {#1}
61             \tl_set:Nn \exp_not:N \l__opacity_backend_stroke_tl {#1}
62             \pdfmanagement_add:nnn { Page / Resources / ExtGState }
63             { opacity #1 }
64             { << /ca ~ #1 /CA ~ #1 >> }
65             \sys_if_engine_xetex:TF
66             { \_kernel_backend_literal_pdf:n }
67             {
68                 \_kernel_color_backend_stack_push:nn
69                 \exp_not:N \c__opacity_backend_stack_int
70             }
71             { /opacity #1 ~ gs }
72         }
73     }
74     \_opacity_backend:nnn {#1} { fill } { ca }
75     \_opacity_backend:nnn {#1} { stroke } { ca }
76 }
77 }
78 \cs_gset_protected:Npe \_opacity_backend_end:
79 {
80     \bool_lazy_any:nTF
81     {
82         { \sys_if_engine_pdftex_p: }

```

```

83         { \sys_if_engine luatex_p: }
84         { \sys_if_engine xetex_p: }
85     }
86     { \__opacity_backend_reset: }
87     {
88         \__opacity_backend_reset_fill:
89         \__opacity_backend_reset_stroke:
90     }
91 }
92 }

```

### 1.3 Extra variants

```

93 \cs_generate_variant:Nn \clist_set:Nn { cv }
94 \cs_generate_variant:Nn \hook_gput_code:nnn { nne }
95 \exp_args_generate:n { nVv }
96 \cs_generate_variant:Nn \color_select:n { V }
97 \cs_generate_variant:Nn \dim_compare:nNnTF { v }
98 \cs_generate_variant:Nn \dim_compare_p:nNn { vNv }
99 \cs_generate_variant:Nn \dim_max:nn { v }
100 \cs_generate_variant:Nn \str_replace_all:Nnn { NnV }
101 \cs_generate_variant:Nn \text_purify:n { v }
102 \cs_generate_variant:Nn \vbox_to_ht:nn { v }

```

### 1.4 Scratch space

`\__talk_tmp:w` For one-off processing.

```

103 \cs_new_protected:Npn \__talk_tmp:w { }

```

*(End of definition for \\_\_talk\_tmp:w.)*

`\l__talk_tmp_box`

```

104 \box_new:N \l__talk_tmp_box

```

*(End of definition for \l\_\_talk\_tmp\_box.)*

`\l__talk_tmp_tl`

```

105 \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
106 \keys_define:nn { talk }
107 {
108     aspect-ratio .str_set:N =
109     \l__talk_aspect_ratio_str ,
110     font-size .dim_set:N =
111     \l__talk_fontsize_dim ,
112     frame-title-arg .bool_set:N =
113     \l__talk_frame_title_bool ,
114     handout .code:n =
115     { \str_set:Nn \l__talk_mode_str { handout } } ,
116     handout .value_forbidden:n = true ,

```

```

117 mode .choices:nn =
118   { handout , projector }
119   { \str_set:NV \l__talk_mode_str \l_keys_choice_tl }
120 }

```

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

Scope for options.

```

121 \keys_define:nn { talk }
122 {
123   aspect-ratio .usage:n = load ,
124   font-size .usage:n = load ,
125   frame-title-arg .usage:n = load ,
126   mode .usage:n = load
127 }

```

Initial values.

```

128 \keys_set:nn { talk }
129 {
130   aspect-ratio = 16:9 ,
131   font-size = 11pt ,
132   frame-title-arg = false ,
133   mode = projector
134 }

```

```

135 \ProcessKeyOptions [ talk ]

```

## 1.6 Setting up

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

```

136 \file_if_exist_input:nF { size \dim_to_decimal:n \l__talk_fontsize_dim .clo }
137 {
138   \file_input:n { size10.clo }
139   \RequirePackage { relsize }
140   \hook_gput_code:nne { begindocument } { talk }
141   { \exp_not:N \relsize { \fp_eval:n { \l__talk_fontsize_dim / 10pt } } } }
142 }

```

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

```

143 \dim_const:Nn \c__talk_paper_height_dim { 100mm }
144 \use:e
145 {
146   \cs_set_protected:Npn \exp_not:N \__talk_tmp:w
147     #1 \tl_to_str:n { : } #2 \tl_to_str:n { : } #3 \exp_not:N \q_stop
148   {
149     \dim_const:Nn \exp_not:N \c__talk_paper_width_dim
150     {
151       \exp_not:N \fp_to_dim:n
152       { (#1 / #2) * \exp_not:N \c__talk_paper_height_dim }
153     }
154   }
155   \exp_not:N \__talk_tmp:w \l__talk_aspect_ratio_str
156   \tl_to_str:n { : } 100 \exp_not:N \q_stop
157 }

```

```

158 \use:e
159 {
160   \exp_not:N \RequirePackage
161   [
162     papersize =
163     {
164       \dim_use:N \c__talk_paper_width_dim ,
165       \dim_use:N \c__talk_paper_height_dim
166     } ,
167     tmargin    = 10mm ,
168     bmargin    = 8mm ,
169     lmargin    = 10mm ,
170     rmargin    = 10mm ,
171     headheight = 10mm ,
172     headsep    = 2mm ,
173     footskip   = 6mm
174   ]
175   { geometry }
176 }

```

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

Turn off justification

```

177 \raggedright

```

## 1.7 Math support

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

```

178 \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.

```

179 \sys_if_engine_opentype:TF
180 {
181   \RequirePackage { fontspec }
182   \RequirePackage { mathtools }
183   \sys_if_engine luatex:TF
184   {
185     \RequirePackage { lua-unicode-math }
186     \tagpdfsetup { math / mathml / luamml / load = true }
187   }
188   { \RequirePackage { unicode-math } }
189   \setmainfont { NewCMSans10-Regular.otf }
190   \setsansfont { NewCMSans10-Regular.otf }

```



```

191 \setmathfont { NewCMSansMath-Regular.otf }
192 }
193 {
194 \RequirePackage { sansmathfonts }
195 \RequirePackage [ nomath ] { lmodern }
196 \cs_set_eq:NN \rmdefault \sfdefault
197 }

```

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.

```

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

```

## 1.9 Hyperlinks

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

```

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

```

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

A requirement.

```

200 \RequirePackage { hyperref }
201 \hypersetup { hidelinks }

```

## 1.10 Tagging

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

```

202 \tagpdfsetup
203 {
204   role / user-NS = ltx-talk      ,
205   role / new-tag = frame / Sect  ,
206   role / new-tag = frametitle / H4
207 }
208 </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 ??.)

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

```

61 \cs_set_eq:NN \set@color \color_ensure_current:

```

(End of definition for `\set@color`. This function is 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.

```
62 \NewDocumentCommand \DeclareColor { m o m }
63 {
64   \IfNoValueTF {#2}
65     { \colorlet {#1} {#3} }
66     { \definecolor {#1} {#2} {#3} }
67 }
```

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

### 1.4 Semantic colors

Pick up the standard colors from beamer.

```
68 \DeclareColor { alert } [ RGB ] { 200 , 0 , 0 }
69 \DeclareColor { example } { green!50!black }
70 \DeclareColor { structure } [ rgb ] { 0.2 , 0.2 , 0.7 }
71 </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 }
35 \cs_new_protected:Npn \__talk_decode_parse_auxii:n #1
36 { \__talk_decode_parse:w #1 | \q_recursion_tail | \q_recursion_stop }

```

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

```

37 \cs_new_protected:Npn \__talk_decode_parse:w #1 |
38 {
39   \quark_if_recursion_tail_stop_do:nn {#1}
40   {
41     \bool_lazy_and:nnT
42       { \str_if_empty_p:N \l__talk_decode_overlays_str }
43       { ! \l__talk_decode_pure_bool }
44       { \bool_set_true:N \l__talk_decode_overlays_bool }
45   }
46   \exp_args:Ne \__talk_decode_mode:n
47   { \tl_trim_spaces:n {#1} }
48   \__talk_decode_parse:w
49 }

```

(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.

```

50 \clist_const:Ne \c__talk_modes_clist
51 {

```

```

52     \tl_to_str:n { handout } ,
53     \tl_to_str:n { projector }
54 }

```

(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
55 \cs_new_protected:Npe \__talk_decode_mode:n #1
56 {
57   \clist_if_in:NnTF \exp_not:N \c__talk_modes_clist {#1}
58   {
59     \exp_not:N \str_if_eq:VnT
60     \exp_not:N \l__talk_mode_str {#1}
61     { \bool_set_true:N \exp_not:N \l__talk_decode_overlays_bool }
62   }
63   {
64     \exp_not:N \__talk_decode_mode:w #1 \tl_to_str:n { : : }
65     \exp_not:N \q_stop
66   }
67 }
68 \use:e
69 {
70   \cs_new_protected:Npe \exp_not:N \__talk_decode_mode:w
71   #1 \token_to_str:N :
72   #2 \token_to_str:N :
73   #3 \exp_not:N \q_stop
74 }
75 {
76   \exp_not:N \tl_if_blank:nTF {#2}
77   {
78     \exp_not:N \__talk_decode_mode:nn
79     { \tl_to_str:n { projector } } {#1}
80   }
81   { \exp_not:N \__talk_decode_mode:nn {#1} {#2} }
82 }
83 \cs_new_protected:Npn \__talk_decode_mode:nn #1#2
84 {
85   \str_if_eq:VnTF \l__talk_mode_str {#1}
86   {
87     \__talk_decode_action:n {#2}
88     \str_if_empty:NT \l__talk_decode_overlays_str
89     { \__talk_decode_overlays:nn { overlays } { * } }
90   }
91   {
92     \tl_if_blank:nF {#2}
93     { \bool_set_false:N \l__talk_decode_pure_bool }
94   }
95 }

```

(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

```

96 \cs_new_protected:Npe \__talk_decode_action:n #1
97 {
98   \exp_not:N \__talk_decode_action:w
99   #1 \tl_to_str:n { @ @ } \exp_not:N \q_stop
100 }
101 \use:e
102 {
103   \cs_new_protected:Npn \exp_not:N \__talk_decode_action:w
104   #1 \tl_to_str:n { @ } #2 \tl_to_str:n { @ } #3 \exp_not:N \q_stop
105 }
106 {
107   \tl_if_blank:nTF {#2}
108   { \__talk_decode_overlays:nn { overlays } {#1} }
109   {
110     \cs_if_exist:cTF { __talk_action_ #1 :N }
111     {
112       \bool_set_false:N \l__talk_decode_pure_bool
113       \str_set:Nn \l__talk_decode_action_str {#1}
114       \tl_if_blank:nF {#2}
115       { \__talk_decode_overlays:nn { actions } {#2} }
116     }
117     {
118       \msg_error:nnV { talk } { bad-action-spec }
119       \l__talk_decode_arg_str
120     }
121   }
122 }

```

(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:nN

```

The loop here needs to replace all + and . characters by the current automatic value, allowing for any offsets. This step also needs to track whether to increment the automatic value: true if a + is seen, false otherwise. If the `amsmath \ifmeasuring@` flag is on, the overlay counter is not advanced.

```

123 \cs_new_protected:Npn \__talk_decode_overlays:nn #1#2
124 {
125   \bool_set_false:N \l__talk_decode_step_bool
126   \__talk_decode_overlays:nN {#1} #2 \q_recursion_tail \q_recursion_stop
127   \bool_if:NT \l__talk_decode_step_bool
128   {
129     \legacy_if:nF { measuring@ }
130     { \int_gincr:N \g__talk_pauses_int }
131   }
132   \__talk_decode_check:n {#1}
133 }
134 \cs_new_protected:Npn \__talk_decode_overlays:nN #1#2
135 {
136   \quark_if_recursion_tail_stop:N #2
137   \cs_if_exist_use:cF { __talk_decode_overlay_ #2 :nw }
138   {
139     \str_put_right:cn { l__talk_decode_ #1 _str } {#2}
140     \__talk_decode_overlays:nN
141   }
142   {#1}

```



```

143 }
144 \cs_new_protected:cpn { __talk_decode_overlay_+:nw } #1
145 {
146   \bool_set_true:N \l__talk_decode_step_bool
147   \__talk_decode_overlay_aux:nNN {#1} 1
148 }
149 \cs_new_protected:cpn { __talk_decode_overlay_.:nw } #1
150 { \__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 (.

```

151 \cs_new_protected:Npn \__talk_decode_overlay_aux:nNN #1#2#3
152 {
153   \quark_if_recursion_tail_stop_do:Nn #3
154   {
155     \__talk_decode_overlay_offset:nNn {#1} #2 { 0 }
156     \q_recursion_tail \q_recursion_stop
157   }
158   \token_if_eq_meaning:NNTF #3 ( % )
159   { \__talk_decode_overlay_offset:nNn {#1} #2 { } }
160   { \__talk_decode_overlay_offset:nNn {#1} #2 { 0 } #3 }
161 }

```

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.

```

162 \cs_new_protected:Npn \__talk_decode_overlay_offset:nNnN #1#2#3#4
163 {
164   \quark_if_recursion_tail_stop_do:Nn #4
165   {
166     \msg_error:nnV { talk } { bad-action-spec }
167     \l__talk_decode_arg_str
168   } % (
169   \token_if_eq_meaning:NNTF #4 )
170   { \__talk_decode_overlay_offset:nNn {#1} #2 {#3} }
171   { \__talk_decode_overlay_offset:nNn {#1} #2 {#3#4} }
172 }

```

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

```

173 \cs_new_protected:Npn \__talk_decode_overlay_offset:nNn #1#2#3
174 {
175   \str_put_right:ce { l__talk_decode_ #1 _str }
176   { \int_max:nn { 0 } { #3 + \g__talk_pauses_int + #2 } }
177   \__talk_decode_overlays:nN {#1}
178 }

```

*(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.

```

179 \cs_new_protected:Npn \__talk_decode_check:n #1
180 {
181   \clist_set:cv { l__talk_decode_ #1 _clist } { l__talk_decode_ #1 _str }
182   \clist_if_in:cnTF { l__talk_decode_ #1 _clist } { * }
183   { \bool_set_true:c { l__talk_decode_ #1 _bool } }
184   {
185     \clist_map_inline:cn { l__talk_decode_ #1 _clist }
186     { \__talk_decode_check:nw {#1} 0 ##1 - - \q_stop }
187   }
188 }

```

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.

```

189 \cs_new_protected:Npn \__talk_decode_check:nw #1#2 - #3 - #4 \q_stop
190 {
191   \tl_if_empty:nTF {#4}
192   { \__talk_decode_check_single:nn {#1} {#2} }
193   {
194     \tl_if_blank:nTF {#3}
195     { \__talk_decode_check_range:nnn {#1} {#2} { \c_max_int } }
196     { \__talk_decode_check_range:nnn {#1} {#2} {#3} }
197   }
198 }
199 \cs_new_protected:Npn \__talk_decode_check_single:nn #1#2
200 {
201   \int_compare:nNnTF \g__talk_slide_int = {#2}
202   { \bool_set_true:c { l__talk_decode_ #1 _bool } }
203   {
204     \int_compare:nNnT {#2} > \g__talk_slide_int
205     { \bool_gset_true:N \g__talk_slide_continue_bool }
206   }
207 }

```

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.

```

208 \cs_set_protected:Npn \__talk_decode_check_range:nnn #1#2#3
209 {
210   \int_compare:nNnF \g__talk_slide_int > {#3}
211   {
212     \int_compare:nNnTF \g__talk_slide_int < {#2}
213     { \bool_gset_true:N \g__talk_slide_continue_bool }
214     {
215       \bool_set_true:c { l__talk_decode_ #1 _bool }
216       \bool_lazy_and:nnT
217       { \int_compare_p:nNn \g__talk_slide_int < {#3} }
218       { \int_compare_p:nNn {#3} < \c_max_int }
219       { \bool_gset_true:N \g__talk_slide_continue_bool }
220     }
221   }
222 }
223 }

```

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

```
224 \msg_new:nnnn { talk } { bad-action-spec }
225 { Bad~overlay~specification~"#1". }
226 {
227   The~overlay~specification~given~doesn't~follow~the~pattern~described~in~
228   the~ltx-talk~manual:~it~has~been~ignored.
229 }
230 </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<sup>A</sup>T<sub>E</sub>X counter-style access.

`\theslide`

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

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

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

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

`\__talk_slide:nn`  
`\__talk_slide_aux:n`

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

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

```

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

```

65 }

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

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

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

`\g__talk_frame_struct_int`

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

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

(End of definition for `\g__talk_frame_struct_int`.)

`\__talk_slide_begin:`

`\__talk_slide_end:`

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

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

`\__talk_slide_align_bottom:n`

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

`\__talk_slide_align_center:n`

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

`\__talk_slide_align_stretch:n`

`\__talk_slide_align_top:n`

```

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

```

(End of definition for \\_\_talk\_slide\_align\_bottom:n and others.)

## 1.2 Counters

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

```

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

```

(End of definition for \l\_\_talk\_cnt\_reset\_seq.)

\\_\_talk\_cnt\_save: A simple save-and-restore pair.

\\_\_talk\_cnt\_restore:

```

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

```

(End of definition for \\_\_talk\_cnt\_save: and \\_\_talk\_cnt\_restore:.)

```

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

```

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

### 1.3 Frame options

```

\l__talk_frame_alignment_tl
148 \tl_new:N \l__talk_frame_alignment_tl

(End of definition for \l__talk_frame_alignment_tl.)

```

```

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

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

```

### 1.4 Tagging for headers

```

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

```



```

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

```

(End of definition for \\_\_talk\_header\_tag\_begin:n and \\_\_talk\_header\_tag\_end:.)

## 1.5 Wallpaper

```

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

```

(End of definition for \l\_\_talk\_footelem\_left\_skip and others.)

```

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

```

Templates for the header area. The background always covers the full width, but the text area may be narrower. The setup here aims to avoid repeated kerns but also dealing with

complex conditionals, hence we always move to the edge of the paper first then adjust as required.

```

218 \NewTemplateType { header } { 0 }
219 \DeclareTemplateInterface { header } { talk } { 0 }
220 {
221     background-color : tokenlist,
222     color             : tokenlist = structure ,
223     font              : tokenlist = \normalfont ,
224     height            : length = \Gm@tmargin + \headsep ,
225     left-hspace       : skip = \Gm@lmargin ,
226     print-frame-title : boolean = true ,
227     right-hspace      : skip = \Gm@rmargin
228 }
229 \DeclareTemplateCode { header } { talk } { 0 }
230 {
231     background-color = \l__talk_header_bg_tl ,
232     color            = \l__talk_header_fg_tl ,
233     font             = \l__talk_header_font_tl ,
234     height           = \l__talk_header_ht_dim ,
235     left-hspace      = \l__talk_header_left_skip ,
236     print-frame-title = \l__talk_header_frametitle_bool ,
237     right-hspace     = \l__talk_header_right_skip
238 }
239 {
240     \noindent
241     \__talk_wallpaper_hruler:Nnn
242     \l__talk_header_bg_tl
243     { \l__talk_header_ht_dim - \headsep }
244     { \headsep }
245     \skip_horizontal:n { \l__talk_header_left_skip }
246     \group_begin:
247         \tl_if_empty:NF \l__talk_header_fg_tl
248         { \color_select:V \l__talk_header_fg_tl }
249         \l__talk_header_font_tl
250         \bool_if:NT \l__talk_header_frametitle_bool
251         {
252             \ExpandArgs { nnV }
253             \UseInstance { frametitle } { header }
254             \g__talk_frame_title_tl
255         }
256     \group_end:
257 }
258 \DeclareInstance { header } { std } { talk } { }
259 \AddToHook { begindocument }
260 {
261     \DeclareInstanceCopy { header } { wallpaper } { std }
262     \EditInstance { header } { wallpaper }
263     { print-frame-title = false }
264 }

```

*(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.

```

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

```

```

318         \hfil
319     }
320     \box_use_drop:N \l__talk_tmp_box
321     \skip_horizontal:n { \l__talk_footer_right_skip - \Gm@rmargin }
322 }
323 \DeclareInstance { footer } { std } { talk } { }
324 \AddToHook { begindocument }
325 {
326     \DeclareInstanceCopy { footer } { wallpaper } { std }
327     \EditInstance { footer } { wallpaper }
328         { element-order = }
329 }

```

(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.

```

330 \cs_new:Npn \__talk_metadata_name:n #1
331 {
332     \tl_if_exist:cTF { @ short #1 }
333         { short #1 }
334         {#1}
335 }

```

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

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

```

336 \cs_new_protected:Npn \__talk_wallpaper_hrule:Nnn #1#2#3
337 {
338     \skip_horizontal:n { -\Gm@lmargin }
339     \tl_if_empty:NF #1
340     {
341         \group_begin:
342         \color_select:V #1
343         \rule:nnn { \paperwidth } {#2} {#3}
344         \skip_horizontal:n { -\paperwidth }
345         \group_end:
346     }
347 }

```

(End of definition for `\__talk_wallpaper_hrule:Nnn`.)

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

```

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

```

```

356   \cs_set_eq:NN \@evenfoot \@oddfoot
357 }
358 \cs_set_nopar:Npn \ps@wallpaper
359 {
360   \cs_set_nopar:Npn \@oddhead
361   {
362     \UseInstance { header } { wallpaper }
363     \hfil
364   }
365   \cs_set_nopar:Npn \@oddfoot
366   {
367     \UseInstance { footer } { wallpaper }
368     \hfil
369   }
370   \cs_set_eq:NN \@evenhead \@oddhead
371   \cs_set_eq:NN \@evenfoot \@oddfoot
372 }
373 \cs_new_nopar:Npn \ps@talk
374 {
375   \cs_set_nopar:Npn \@oddhead
376   {
377     \UseInstance { header } { std }
378     \hfil
379   }
380   \cs_set_nopar:Npn \@oddfoot { \UseInstance { footer } { std } }
381   \cs_set_eq:NN \@evenhead \@oddhead
382   \cs_set_eq:NN \@evenfoot \@oddfoot
383 }
384 \pagestyle { talk }

```

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

## 1.6 The frame environment

<pre> \l__talk_frame_bool </pre>	<p>To track whether we are inside a frame or not.</p> <pre> 385 \bool_new:N \l__talk_frame_bool </pre> <p>(End of definition for \l__talk_frame_bool.)</p>
<pre> \g__talk_frame_tag_bool </pre>	<p>To track when a frame is being tagged: mainly needed for the header (and as a result global).</p> <pre> 386 \bool_new:N \g__talk_frame_tag_bool </pre> <p>(End of definition for \g__talk_frame_tag_bool.)</p>
<pre> \l__talk_frame_verb_bool </pre>	<p>Indicates that material was collected verbatim (and thus needs rescanning).</p> <pre> 387 \bool_new:N \l__talk_frame_verb_bool </pre> <p>(End of definition for \l__talk_frame_verb_bool.)</p>
<pre> \g__talk_frame_int \c@frame \theframe \@framenumbers </pre>	<p>The overall frame number, including L<sup>A</sup>T<sub>E</sub>X counter-like access.</p> <pre> 388 \int_new:N \g__talk_frame_int 389 \cs_new_eq:NN \c@frame \g__talk_frame_int 390 \cs_new:Npn \theframe { \@arabic \c@frame } 391 \cs_new:Npn \@framenumbers { \arabic { frame } } </pre>

*(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.

```

392 \property_new:nnnn { totalframes } { shipout } { -1 }
393   { \int_use:N \g__talk_frame_int }
394 \AddToHook { enddocument / afterlastpage }
395   { \property_record:nn { lastpage } { totalframes } }
396 \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.

```

397 \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.

```

398 \cs_new_protected:Npn \__talk_frame_process:nn #1#2
399   {
400     \int_gincr:N \g__talk_frame_int
401     \bool_set_true:N \l__talk_frame_bool
402     \__talk_slide:nn {#1} {#2}
403   }

```

*(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.

```

404 \cs_new_protected:Npn \__talk_frame_tag:n #1
405   {
406     \tag_struct_begin:n { tag = frame }
407     \int_gset:Nn \g__talk_frame_struct_int { \tag_get:n { struct_num } }
408     \bool_gset_true:N \g__talk_frame_tag_bool
409     #1
410     \tag_struct_end:
411   }

```

*(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.

```

412 \cs_new_protected:Npn \__talk_frame_notag:n #1
413   {
414     \tag_mc_begin:n { artifact }
415     \tag_suspend:n { frame }
416     \bool_gset_false:N \g__talk_frame_tag_bool
417     #1
418     \par
419     \tag_resume:n { frame }
420     \tag_mc_end:
421   }

```

*(End of definition for \\_\_talk\_frame\_notag:n.)*

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

```

422 \bool_if:NTF \l__talk_frame_title_bool
423 {
424   \RenewDocumentEnvironment { frame }
425     { D <> { all } = { action-spec } 0 { } +m +b }
426     {
427       \keys_set:nn { talk / frame } {#2}
428       \bool_set_false:N \l__talk_frame_verb_bool
429       \__talk_frame_process:nn {#1} { \frametitle {#3} #4 }
430     }
431   { }
432   \NewDocumentEnvironment { frame* }
433     { D <> { all } = { action-spec } 0 { } +m c }
434     {
435       \keys_set:nn { talk / frame } {#2}
436       \bool_set_true:N \l__talk_frame_verb_bool
437       \tl_gset:Nn \g__talk_frame_title_tl {#3}
438       \exp_args:Nne \__talk_frame_process:nn {#1}
439         { \tl_to_str:n { \frametitle } \exp_not:n { {#3} #4 } }
440     }
441   { }
442 }
443 {
444   \RenewDocumentEnvironment { frame }
445     { !D <> { all } = { action-spec } !0 { } +b }
446     {
447       \keys_set:nn { talk / frame } {#2}
448       \bool_set_false:N \l__talk_frame_verb_bool
449       \__talk_frame_process:nn {#1} {#3}
450     }
451   { }
452   \NewDocumentEnvironment { frame* }
453     { !D <> { all } = { action-spec } !0 { } c }
454     {
455       \keys_set:nn { talk / frame } {#2}
456       \bool_set_true:N \l__talk_frame_verb_bool
457       \__talk_frame_process:nn {#1} {#3}
458     }
459   { }
460 }

```

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

461 </class>

## Part V

# ltx-talk-frame – The structure of frames

## 1 ltx-talk-frame-structure implementation

Start the DocStrip guards.

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

### 1.1 Columns

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

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

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

*(End of definition for `\l__talk_columns_wd_tl`.)*

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

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



`\l__talk_column_alignment_tl`

```

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

```

(End of definition for `\l__talk_column_alignment_tl`.)

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

Based on ideas in the highly experimental `xbox`.

```

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

```

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

`column (env.)` A cut-down version of a minipage: we want to be clear on the semantic meaning. the action is applied inside the box after starting horizontal mode to avoid spacing issues when switching whatsits in and out.

```

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

```

```

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

```

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

```

82 {
83   \par
84   \__talk_action_end:
85   \vbox_set_end:
86   \use:c { __talk_column_align_ \l__talk_column_alignment_tl :n }
87   { \vbox_unpack_drop:N \l__talk_tmp_box }
88   \hfil
89   \par
90   \@ignoretrue
91 }

```

## 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.

```

92 \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.

```

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

```

```

108 }
109 {
110   \SetTemplateKeys { floatenv } { talk } {#1}
111   \begin { minipage } { \columnwidth }
112     \begin { \l__talk_float_alignment_tl }
113       \cs_set_nopar:Npn \@captype {#2}
114     }
115   \DeclareInstance { floatenv } { std } { talk } { horizontal-alignment = left }
\endfloatenv And the common end function.
116 \cs_new_protected:Npn \endfloatenv
117 {
118   \end { \l__talk_float_alignment_tl }
119   \end { minipage }
120 }

```

(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.)**

```

121 \clist_map_inline:nn { figure , table }
122 {
123   \NewDocumentEnvironment {#1} { D <> { all } = { float-placement } 0 { } }
124   {
125     \__talk_action_begin:n {##1}
126     \UseInstance { floatenv } { std } {##2} {#1}
127   }
128   {
129     \endfloatenv
130     \__talk_action_end:
131   }

```

**\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**

**\thetable**

**\figurename**

**\tablename**

**\fnum@figure**

**\fnum@table**

```

132 \newcounter {#1}
133 \tl_new:c { #1 name }
134 \tl_set:ce { #1 name } { \text_titlecase_first:n {#1} }
135 \tl_new:c { fnum@ #1 }
136 \tl_set:ce { fnum@ #1 }
137 { \exp_not:c { #1 name } \exp_not:N \nobreakspace \exp_not:c { the #1 } }
138 }

```

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

The spacing values needed for the standard function.

```

139 \newlength \abovecaptionskip
140 \newlength \belowcaptionskip
141 \setlength \abovecaptionskip { 7pt }
142 \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.

```

143 \cs_set_protected:Npn \@caption #1 [ #2 ] #3
144 {
145   \par

```

```

146 \begingroup
147 \parboxrestore
148 \if@minipage \setminipage \fi
149 \normalsize
150 \@makecaption { \csname fnum@ #1 \endcsname } { \ignorespaces #3 }
151 \par
152 \endgroup
153 }

```

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

### 1.3 Footnotes

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

```

154 \cs_new_protected:Npn \@makefnmark #1
155 {
156   \parindent 1em
157   \noindent
158   \hb@xt@ 1.8em { \hss \@makefnmark }
159   #1
160 }

```

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

```

161 \endclass

```

## 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.

\g\_\_talk\_opacity\_seq For tracking.

```

38 \seq_new:N \g__talk_opacity_seq
39 \seq_gpush:Nn \g__talk_opacity_seq { 1 }

```

*(End of definition for \g\_\_talk\_opacity\_seq.)*

\\_\_talk\_opacity\_begin:n Simply tracking wrappers.

```

\__talk_opacity_end:
40 \cs_new_protected:Npn \__talk_opacity_begin:n #1
41 {
42   \seq_gpush:Nn \g__talk_opacity_seq {#1}
43   \__talk_shuffle_skip:n { \opacity_begin:n {#1} }
44 }
45 \cs_new_protected:Npn \__talk_opacity_end:
46 {
47   \__talk_shuffle_skip:n { \opacity_end: }
48   \seq_gpop:NN \g__talk_opacity_seq \l__talk_tmp_tl
49 }

```

*(End of definition for \\_\_talk\_opacity\_begin:n and \\_\_talk\_opacity\_end:.)*

\\_\_talk\_opacity\_reapply: Simply tracking wrappers.

```

50 \cs_new_protected:Npn \__talk_opacity_reapply:
51 {
52   \seq_get:NN \g__talk_opacity_seq \l__talk_tmp_tl
53   \exp_args:NV \__talk_opacity_begin:n \l__talk_tmp_tl
54 }

```

*(End of definition for \\_\_talk\_opacity\_reapply:.)*

### 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_:N`  
`\_talk\_action\_end:N`

The fallback action. This is needed for two reasons. First, it ensures that spacing is correct in terms of whatsits. Second, it deals with the need to keep opacity in-track in things like lists.

```
55 \cs_new_protected:Npn \_talk\_action_:N #1 { \_talk\_opacity\_reapply: }
56 \cs_new_protected:Npn \_talk\_action\_end:N #1 { \_talk\_opacity\_end: }
```

*(End of definition for `\_talk\_action_:N` and `\_talk\_action\_end:N`.)*

`\_talk\_action\_alert:N`

At present a color selection.

```
57 \cs_new_protected:Npn \_talk\_action\_alert:N #1
58 {
59   \bool_if:NTF #1
60     { \color\_select:n { alert } }
61     { \color\_select:n { . } }
62 }
```

*(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`

Simply (un)hide unconditionally, overwriting any previous opacity.

```
63 \cs_new_protected:Npn \_talk\_action\_invisible:N #1
64 {
65   \bool_if:NTF #1
66     { \_talk\_opacity\_begin:n { 0 } }
67     { \_talk\_opacity\_begin:n { 1 } }
68 }
69 \cs_new_protected:Npn \_talk\_action\_invisible\_end:N #1
70 { \_talk\_opacity\_end: }
71 \cs_new_protected:Npn \_talk\_action\_visible:N #1
72 {
73   \bool_if:NTF #1
74     { \_talk\_opacity\_begin:n { 1 } }
75     { \_talk\_opacity\_begin:n { 0 } }
76 }
77 \cs_new_protected:Npn \_talk\_action\_visible\_end:N #1
78 { \_talk\_opacity\_end: }
```

*(End of definition for `\_talk\_action\_invisible:N` and others.)*

`\_talk\_action\_only:N`  
`\_talk\_action\_only\_end:N`

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

```
79 \cs_new_protected:Npn \_talk\_action\_only:N #1
80 {
81   \bool_if:NF #1
82     { \vbox\_set:Nw \l\_talk\_tmp\_box }
83 }
84 \cs_new_protected:Npn \_talk\_action\_only\_end:N #1
85 {
86   \bool_if:NF #1
87     { \vbox\_set\_end: }
88 }
```



(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.

```

89 \NewTemplateType { hidden } { 0 }
90 \DeclareTemplateInterface { hidden } { talk } { 0 }
91   { opacity : real = 0 }
92 \DeclareTemplateCode { hidden } { talk } { 0 }
93   { opacity = \l__talk_uncover_hidden_fp }
94   { \__talk_opacity_begin:n { \l__talk_uncover_hidden_fp } }
95 \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.

```

96 \cs_new_protected:Npn \__talk_action_uncover:N #1
97   {
98     \bool_if:NTF #1
99       { \__talk_opacity_begin:n { 1 } }
100       { \UseInstance { hidden } { std } }
101   }
102 \cs_new_protected:Npn \__talk_action_uncover_end:N #1
103   { \__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 104 \clist_map_inline:nm { invisible , uncover , visible }
\visible 105   {
106     \ExpandArgs { cne } \NewDocumentCommand { #1 }
107       { > { \__talk_overlay_arg:n } D <> { all } +m }
108       {
109         \exp_not:c { __talk_action_ #1 :N } ##1
110         ##2
111         \exp_not:c { __talk_action_ #1 _end:N } ##1
112       }
113   }
```

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

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

```

\uncoverenv (env.) 113 \ExpandArgs { nnee } \NewDocumentEnvironment { #1 env }
\visibleenv (env.) 114   { > { \__talk_overlay_arg:n } D <> { all } }
115   { \exp_not:c { __talk_action_ #1 :N } ##1 }
116   { \exp_not:c { __talk_action_ #1 _end:N } ##1 }
117   }
```

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

```

118 \NewDocumentCommand \alert { > { \__talk_overlay_arg:n } D <> { all } +m }
119   {
120     \group_begin:
121       \__talk_action_alert:N #1
122       #2
123     \group_end:
124   }
```

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

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

```

125 \NewDocumentEnvironment { alertenv } { > { \_talk_overlay_arg:n } D <> { all } }
126   { \_talk_action_alert:N #1 }
127   { }

```

**\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).

```

128 \NewDocumentCommand \only { D <> { all } +m }
129   {
130     \_talk_if_overlay:nT {#1}
131     {#2}
132   }

```

(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.

```

133 \NewDocumentEnvironment { onlyenv } { > { \_talk_overlay_arg:n } D <> { all } }
134   { \_talk_action_only:N #1 }
135   { \_talk_action_only_end:N #1 }

```

```

\_talk_saved_overlays_bool
\_talk_saved_action_str
\_talk_saved_actions_bool
136 \bool_new:N \_talk_saved_overlays_bool
137 \str_new:N \_talk_saved_action_str
138 \bool_new:N \_talk_saved_actions_bool

```

(End of definition for \\_talk\_saved\_overlays\_bool, \\_talk\_saved\_action\_str, and \\_talk\_saved\_actions\_bool.)

```

\_talk_overlay_all_bool
139 \bool_new:N \_talk_overlay_all_bool

```

(End of definition for \\_talk\_overlay\_all\_bool.)

**actionenv** (*env.*) 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<sup>A</sup>T<sub>E</sub>X 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:
140 \NewDocumentCommand \action { D <> { all } +m }
141   {
142     \group_begin:
143       \_talk_action_begin:n {#1}
144       #2
145       \_talk_action_end:
146     \group_end:
147   }
148 \NewDocumentEnvironment { actionenv } { D <> { all } }
149   { \_talk_action_begin:n {#1} }

```

```

150 { \__talk_action_end: }
151 \cs_new_protected:Npn \__talk_action_begin:n #1
152 {
153   \group_begin:
154   \str_if_eq:nnTF {#1} { all }
155     { \bool_set_true:N \l__talk_overlay_all_bool }
156     {
157       \bool_set_false:N \l__talk_overlay_all_bool
158       \__talk_action_begin_aux:n {#1}
159     }
160 }
161 \cs_new_protected:Npn \__talk_action_begin_aux:n #1
162 {
163   \__talk_decode_parse:n {#1}
164   \bool_set_eq:NN \l__talk_saved_overlays_bool
165     \l__talk_decode_overlays_bool
166   \str_set_eq:NN \l__talk_saved_action_str
167     \l__talk_decode_action_str
168   \bool_set_eq:NN \l__talk_saved_actions_bool
169     \l__talk_decode_actions_bool
170   \tl_if_empty:NTF \g__talk_onslide_tl
171     {
172       \bool_if:NTF \l__talk_decode_overlays_bool
173         {
174           \use:c { __talk_action_ \l__talk_decode_action_str :N }
175           \l__talk_decode_actions_bool
176         }
177         { \UseInstance { hidden } { std } }
178       }
179     { \__talk_action_invisible:N \c_true_bool }
180 }
181 \cs_new_protected:Npn \__talk_action_end:
182 {
183   \bool_if:NF \l__talk_overlay_all_bool
184   {
185     \bool_if:NTF \l__talk_saved_overlays_bool
186     {
187       \cs_if_exist_use:cF
188       { __talk_action_ \l__talk_saved_action_str _end:N }
189       { \use_none:n }
190       \l__talk_saved_actions_bool
191     }
192     { \__talk_opacity_end: }
193   }
194   \group_end:
195 }

```

(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.

```

196 \NewDocumentCommand \alt { D <> { all } +m +m }
197 {
198   \__talk_if_overlay:nTF {#1}
199     {#2}
200     {#3}
201 }

```

*(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  
**\\_\_talk\_onslide:n** tabular cells.

```

202 \NewDocumentCommand \onslide { D <> { all } }
203 {
204   \__talk_onslide:n {#1}
205   \ignorespaces
206 }
207 \cs_new_protected:Npn \__talk_onslide:n #1
208 {
209   \tl_use:N \g__talk_onslide_tl
210   \tl_gclear:N \g__talk_onslide_tl
211   \__talk_if_overlay:nF {#1}
212   {
213     \__talk_opacity_begin:n { 0 }
214     \tl_gput_right:Nn \g__talk_onslide_tl
215       { \__talk_opacity_end: }
216   }
217 }

```

*(End of definition for \onslide and \\_\_talk\_onslide:n. This function is documented on page ??.)*

**\g\_\_talk\_onslide\_tl**

```

218 \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.

```

219 \NewDocumentCommand \temporal { D <> { all } +m +m +m }
220 {
221   \__talk_if_overlay:nTF {#1}
222     {#3}
223     {
224       \bool_if:NTF \g__talk_slide_continue_bool
225         {#4}
226         {#2}
227     }
228 }

```

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

**\pause** A thin wrapper.

```

229 \NewDocumentCommand \pause { o }
230 {
231   \legacy_if:nF { measuring@ }
232   {

```

```

233     \IfNoValueTF {#1}
234     { \int_gincr:N \g__talk_pauses_int }
235     { \int_gset:Nn \g__talk_pauses_int {#1} }
236     \exp_args:Ne \__talk_onslide:n { \int_eval:n { \g__talk_pauses_int + 1 } - }
237   }
238 }

```

(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`.

```

239 \cs_new_protected:Npn \__talk_overprint_begin:n #1
240 {
241   \par
242   \vbox_set_to_wd:Nnw \l__talk_tmp_box {#1}
243   \raggedright
244   \ignorespaces
245 }

```

(End of definition for \\_\_talk\_overprint\_begin:n.)

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

```

246 \NewDocumentEnvironment { overlayarea } { m m }
247 { \__talk_overprint_begin:n {#1} }
248 {
249   \vbox_set_end:
250   \vbox_to_ht:nn {#2}
251   {
252     \box_use_drop:N \l__talk_tmp_box
253     \vfil
254   }
255   \par
256 }

```

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

```

257 \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.

```

258 \cs_new:Npn \__talk_frame_overprint:
259 {
260   \int_to_Roman:n \g__talk_frame_int
261   \int_to_roman:n \l__talk_overprint_int
262 }

```

(End of definition for \\_\_talk\_frame\_overprint:.)

`\_talk_overprint@_t1
 \_talk_overprint_check_ht:n`

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.

```

263 \NewDocumentEnvironment { overprint } { 0 { \textwidth } }
264 { \_talk_overprint_begin:n {#1} }
265 {
266   \vbox_set_end:
267   \int_incr:N \l__talk_overprint_int
268   \_talk_overprint_save_ht:
269   \cs_if_exist:cTF
270     { overprint@ \_talk_frame_overprint: }
271     {
272       \dim_compare:vNnTF
273         { overprint@ \_talk_frame_overprint: }
274         > { \box_ht:N \l__talk_tmp_box }
275         {
276           \vbox_to_ht:vn
277             { overprint@ \_talk_frame_overprint: }
278             {
279               \box_use_drop:N \l__talk_tmp_box
280               \vfil
281             }
282           }
283         { \box_use_drop:N \l__talk_tmp_box }
284       }
285     { \box_use_drop:N \l__talk_tmp_box }
286   \par
287 }

```

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.

```

288 \cs_new_protected:Npn \_talk_overprint_save_ht:
289 {
290   \tl_if_exist:cF { g__talk_overprint_ \_talk_frame_overprint: _t1 }
291   {
292     \tl_new:c { g__talk_overprint_ \_talk_frame_overprint: _t1 }
293     \tl_gset:cn { g__talk_overprint_ \_talk_frame_overprint: _t1 }
294       { Opt }
295   }
296   \tl_gset:ce { g__talk_overprint_ \_talk_frame_overprint: _t1 }
297   {
298     \dim_max:vn { g__talk_overprint_ \_talk_frame_overprint: _t1 }
299     { \box_ht:N \l__talk_tmp_box }
300   }
301   \legacy_if:nT { @files }
302   {
303     \iow_now:Ne \@auxout
304     {
305       \gdef \exp_not:c { overprint@ \_talk_frame_overprint: }
306       {
307         \exp_not:v { g__talk_overprint_ \_talk_frame_overprint: _t1 }
308       }
309     }
310   }

```

```

311 \hook_gput_code:nne { enddocument / afterlastpage } { talk }
312 { \__talk_overprint_check_ht:n { \__talk_frame_overprint: } }
313 }
314 \cs_new_protected:Npn \__talk_overprint_check_ht:n #1
315 {
316 \bool_lazy_and:nnF
317 { \exp_not:N \cs_if_exist_p:c { overprint@ #1 } }
318 {
319 \dim_compare_p:vNv { overprint@ #1 } = { g__talk_overprint_ #1 _tl }
320 }
321 {
322 \msg_warning:nn { talk } { overprint-ht }
323 \cs_gset_protected:Npn \__talk_overprint_check_ht:n ##1 { }
324 }
325 }
326 \msg_new:nnn { talk } { overprint-ht }
327 {
328 Overprint~area~height~has~changed:\\
329 rerun~LaTeX.
330 }

```

(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.

```

\textnormal 331 \tl_map_inline:nn
\textrm      332 {
\textsc      333 \textbf
\textsf      334 \textit
\textsl      335 \textmd
\texttt      336 \textnormal
\textup      337 \textrm
\emph        338 \textsc
\stdtextbf   339 \textsf
\stdtextit   340 \textsl
\stdtextmd   341 \texttt
\stdtextnormal 342 \textup
\stdtextrm   343 \emph
\stdtextsc   344 }
\stdtextsf   345 {
\stdtextsl   346 \ExpandArgs { c } \NewCommandCopy { std \cs_to_str:N #1 } #1
\stdtexttt   347 \ExpandArgs { Nne } \RenewDocumentCommand #1
\stdtextup   348 { D <> { all } +m }
\stdemph     349 {
350 \exp_not:N \__talk_if_overlay:nTF {##1}
351 { \exp_not:c { std \cs_to_str:N #1 } }
352 { \exp_not:N \__talk_textcmd_equiv:n }
353 {##2}
354 }
355 }
356 \cs_new_protected:Npn \__talk_textcmd_equiv:n #1

```

```

357 {
358   \mode_if_math:TF
359     { { \mbox {#1} } }
360     {
361       \mode_leave_vertical:
362       {#1}
363     }
364 }

```

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

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

```

\stdincludegraphics
365 \RequirePackage { graphicx }
366 \NewCommandCopy \stdincludegraphics \includegraphics
367 \RenewDocumentCommand \includegraphics { s D <> { all } o o m }
368 {
369   \__talk_if_overlay:nT {#2}
370   {
371     \use:e
372     {
373       \exp_not:N \stdincludegraphics
374       \IfBooleanT #1 { * }
375       \IfNoValueF {#3} { [ \exp_not:n { {#3} } ] }
376       \IfNoValueF {#4} { [ \exp_not:n { {#4} } ] }
377     }
378     {#5}
379   }
380 }

```

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

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

```

\__talk_label:n
381 \RenewDocumentCommand \label { D <> { 1 } m }
382 {
383   \@bsphack
384   \__talk_if_overlay:nT {#1}
385   { \__talk_label:n {#2} }
386   \@esphack
387 }
388 \cs_new_protected:Npn \__talk_label:n #1
389 {
390   \begingroup
391   \UseHookWithArguments { label } { 1 } {#1}
392   \protected@write \@auxout { }
393   {
394     \string \newlabel {#1}
395     {
396       { \@currentlabel }
397       { \thepage }
398       { \@currentlabelname }

```



```

399         { \@currentHref }
400         { \@kernel@reserved@label@data }
401     }
402 }
403 \endgroup
404 }

```

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

```

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

```

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

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

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

```

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

```

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

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

```

81 \seq_set_from_clist:Nn \l_tmpa_seq
82   { section , subsection , subsubsection }
83 \seq_map_indexed_inline:Nn \l_tmpa_seq
84   {
85     \use:e
86     {
87       \NewDocumentCommand \exp_not:c { insert #2 } { { }
88         {
89           \exp_not:N \tl_use:N
90           \exp_not:c { l__talk_ #2 _tl }
91         }
92       \NewDocumentCommand \exp_not:c {#2}
93       { s D <> { all } 0 {##4} m }
94       {
95         \exp_not:N \refstepcounter {#2}
96         \UseTaggingSocket { sec / end } { \use:c { toplevel@ #2 } }
97         \UseTaggingSocket { sec / begin }
98         {
99           { \use:c { toplevel@ #2 } }
100          {
101            tag =
102            \exp_not:N \UseStructureName
103            { sec / \use:c { toplevel@ #2 } }
104          }
105        }
106        \tl_set:Nn \exp_not:c { l__talk_ #2 _tl } {##4}
107        \UseTaggingSocket { talk / sec / title } {#2}
108        \str_if_eq:nnT {#2} { section }
109          { \tl_clear:N \exp_not:N \l__talk_subsection_tl }
110        \str_if_eq:nnF {#2} { subsubsection }
111          { \tl_clear:N \exp_not:N \l__talk_subsubsection_tl }
112        \exp_not:N \addcontentsline { toc } {#2}

```

```

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

```

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

```

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

```

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

### 1.3 Table of contents

`\@starttoc` The standard kernel implementation here deliberately overwrites the file as soon as it's read. That's no good for us as the table of contents can be read multiple times. So we modify the code: we start from the tagging-aware version (this may need to be revisited). We retain the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> code as much as possible.

```

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

```

```

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

```

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

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

```

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

```

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

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

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

```

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

```



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

```

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

```

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

```

235 \setcounter { tocdepth } { 2 }

```

## 1.4 Block environments

<code>description (env.)</code>	Stub logical environments: needed as the tagging setup expects these to exist.
<code>quote (env.)</code>	236 \NewDocumentEnvironment { description } { } { } { }
<code>quotation (env.)</code>	237 \NewDocumentEnvironment { quote } { } { } { }
<code>verse (env.)</code>	238 \NewDocumentEnvironment { quotation } { } { } { }
<code>stdquote (env.)</code>	239 \NewDocumentEnvironment { verse } { } { } { }
<code>stdquotation (env.)</code>	
<code>stdverse (env.)</code>	

```

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

```

`block (env.)`

```

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

```

## 1.5 Lists

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

```

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

```

```

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

```

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

```

294 \cs_new_protected:Npn \__talk_item_parse_spec:w #1 < #2 > #3 \q_stop
295 { \__talk_item_parse_spec:n {#2} }
296 \cs_new_protected:Npn \__talk_item_parse_spec:n #1
297 {
298     \bool_lazy_or:nnF
299     { \tl_if_blank_p:n {#1} }
300     { \str_if_eq_p:nn {#1} { all } }
301     {
302         \tl_set:Nx \l__talk_list_end_tl
303         {
304             \exp_not:N \int_compare:nNnT \tex_currentgrouplevel:D =
305             { \int_use:N \tex_currentgrouplevel:D + 1 }
306             {
307                 \__talk_action_end:
308                 \tl_clear:N \exp_not:N \l__talk_list_end_tl
309             }
310         }
311         \__talk_action_begin:n {#1}
312     }
313 }

```

*(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`

```

314 \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.

```

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

```

```

323     \par
324   }
325   \l__talk_list_end_tl
326   \__kernel_list_item_end:
327   \__kernel_list_item_begin:
328   \addpenalty \@itempenalty
329   \addvspace \itemsep
330 }
331 \cs_gset:Npn \endblockenv
332 {
333   \__block_debug_typeout:n { blockenv~common~ending \on@line }
334   \bool_if:NT \l__block_level_incr_bool
335     { \int_gdecr:N \g_block_nesting_depth_int }
336   \legacy_if:nT { @inlabel }
337     {
338       \mode_leave_vertical:
339       \legacy_if_gset_false:n { @inlabel }
340     }
341   \__block_if_list:T
342     { \legacy_if:nT { @newlist } { \@noitemerr } }
343   \mode_if_horizontal:TF
344     {
345       \__block_skip_remove_last:
346       \__block_skip_remove_last:
347       \par
348     }
349     { \@inmatherr { \end { \@currenvir } } }
350   \l__talk_list_end_tl
351   \__kernel_displayblock_end:
352   \__block_if_list:T { \legacy_if_gset_false:n { @newlist } }
353   \legacy_if:nF { @nparlist }
354     {
355       \__block_skip_set_to_last:N \l_tmpa_skip
356       \dim_compare:nNnT \l_tmpa_skip > \c_zero_dim
357         {
358           \skip_vertical:n { - \l_tmpa_skip }
359           \skip_vertical:n { \l_tmpa_skip + \parskip - \@outerparskip }
360         }
361       \addpenalty \@endparpenalty
362       \addvspace \l__block_topsepadd_skip
363     }
364   \socket_use:n { block / endpe }
365 }

```

(End of definition for \\_\_block\_inter\_item: and \endblockenv. This function is documented on page ??.)

```

itemize (env.) Allow for the classical beamer syntax.
enumerate (env.) 366 \AddToHook { begindocument / before }
description (env.) 367 {
368   \clist_map_inline:nn { itemize , enumerate , description }
369   {
370     \RenewDocumentEnvironment {#1} { = { action-spec } !o }
371     {

```

```

372         \IfNoValueTF {##1}
373         { \UseInstance { blockenv } {#1} { } }
374         { \UseInstance { blockenv } {#1} {##1} }
375     }
376     { \endblockenv }
377 }
378 }

```

And add the structural color to item labels.

```

379 \AddToHook { begindocument / before }
380 {
381     \EditInstance { item } { basic }
382     { label-format = \color { structure } #1 }
383     \EditInstance { item } { description }
384     { label-format = \normalfont \bfseries \color { structure } #1 }
385 }

```

`\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.

```

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

```

(End of definition for `\l__talk_action_spec_str`.)

## 1.6 Theorems, *etc.*

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

```

388 \NewCommandCopy \stdnewtheorem \newtheorem
389 \RenewDocumentCommand \newtheorem { m O {#1} m o }
390 {
391     \IfNoValueTF {#4}
392     { \stdnewtheorem {#1} [ {#2} ] {#3} }
393     { \stdnewtheorem {#1} [ {#2} ] {#3} [ {#4} ] }
394     \NewEnvironmentCopy { std #1 } {#1}
395     \RenewDocumentEnvironment {#1} { D <> { all } o }
396     {
397         \__talk_action_begin:n {##1}
398         \IfNoValueTF {##2}
399         { \begin { std #1 } }
400         { \begin { std #1 } [ {##2} ] }
401         \ignorespaces
402     }
403     {
404         \end { std #1 }
405         \__talk_action_end:
406     }
407 }

```

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

```

408 \</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 ..... 123  
 \\ ..... 328

## A

\abovecaptionskip ..... 139, 141  
 \action ..... 140  
 actionenv (env.) ..... 140  
 \addcontentsline ..... 112  
 \addpenalty ..... 328, 361  
 \AddToHook ..... 49, 54, 66, 151,  
                   198, 240, 259, 278, 324, 366, 379, 394  
 \addtolength ..... 48  
 \addvspace ..... 329, 362  
 \alert ..... 38, 118  
 alertenv (env.) ..... 125  
 \alt ..... 196  
 \and ..... 131  
 \arabic ..... 391  
 \arraycolsep ..... 25  
 \arrayrulewidth ..... 27  
 \AssignTaggingSocketPlug ..... 141  
 \author ..... 15

## B

\begin ... 111, 112, 130, 144, 248, 399, 400  
 \begingroup ..... 144, 146, 390  
 \belowcaptionskip ..... 140, 142  
 \bfseries ..... 21, 39, 167, 265, 384  
 \bigskipamount ..... 18  
 block (env.) ..... 257  
 block commands:  
   \g\_block\_nesting\_depth\_int ..... 335  
 block internal commands:  
   \\_\_block\_debug\_typeout:n ..... 333  
   \\_\_block\_if\_list:TF ..... 341, 352  
   \\_\_block\_inter\_item: ..... 315, 315  
   \l\_block\_level\_incr\_bool ..... 334  
   \\_\_block\_list\_end: ..... 56  
   \\_\_block\_list\_item\_end: ..... 56  
   \\_\_block\_skip\_remove\_last: .....  
     ..... 321, 322, 345, 346  
   \\_\_block\_skip\_set\_to\_last:N ..... 355  
   \l\_block\_topsepadd\_skip ..... 362  
 bool commands:  
   \bool\_do\_while:Nn ..... 27  
   \bool\_gset\_false:N ... 30, 36, 40, 416

\bool\_gset\_true:N . 205, 213, 219, 408  
 \bool\_if:NnTF ..... 6, 44,  
                   46, 59, 65, 73, 81, 85, 86, 98, 127,  
                   141, 172, 183, 185, 224, 250, 334, 422  
 \bool\_lazy\_and:nTF ... 21, 41, 216, 316  
 \bool\_lazy\_any:nTF ..... 53, 80  
 \bool\_lazy\_or:nTF ..... 5, 21, 298  
 \bool\_lazy\_or\_p:n ..... 24  
 \bool\_new:N ..... 3,  
                   3, 7, 8, 13, 136, 138, 139, 385, 386, 387  
 \bool\_set\_eq:N ..... 164, 168  
 \bool\_set\_false:N .....  
                   ... 27, 28, 93, 112, 125, 157, 428, 448  
 \bool\_set\_true:N .. 24, 29, 44, 61,  
                   146, 155, 183, 202, 215, 401, 436, 456  
 \c\_false\_bool ..... 15  
 \c\_true\_bool ..... 14, 179  
 box commands:  
   \box\_dp:N ..... 25  
   \box\_ht:N ..... 60, 274, 299  
   \box\_move\_down:n ..... 58  
   \box\_new:N ..... 4, 104  
   \box\_use:N ..... 274  
   \box\_use\_drop:N .....  
     ..... 24, 252, 279, 283, 285, 320  
   \box\_wd:N ..... 30  
 box internal commands:  
   \\_\_box\_dim\_eval:n ..... 22, 25, 30, 33  
   \\_\_box\_set\_to\_wd: ..... 29, 34

## C

\clearpage ..... 92  
 clist commands:  
   \clist\_const:Nn ..... 50  
   \clist\_if\_in:NnTF ..... 57, 182  
   \clist\_map\_break: ..... 220  
   \clist\_map\_inline:Nn ... 132, 185, 307  
   \clist\_map\_inline:nn .....  
     ..... 3, 104, 121, 242, 368  
   \clist\_new:N ..... 10, 14  
   \clist\_pop:NNTF ..... 303  
   \clist\_set:Nn ..... 93, 181  
 \color 4, 11, 56, 167, 174, 184, 266, 382, 384  
 color commands:  
   \color\_ensure\_current: ..... 61  
   \color\_group\_begin: ..... 23, 35  
   \color\_group\_end: ..... 23  
   \color\_math:nn ..... 9, 26





<b>K</b>	
kernel internal commands:	
\__kernel_backend_literal_pdf:n .	66
\__kernel_color_backend_stack_-	
push:nn	68
\__kernel_displayblock_end:	351
\__kernel_list_item_begin:	327
\__kernel_list_item_end:	326
keys commands:	
\l_keys_choice_tl	119
\keys_define:nn	
.....	3, 5, 6, 28, 106, 121, 149, 386
\keys_set:nn ..	7, 13, 17, 24, 29, 36,
41, 46, 72, 128, 162, 427, 435, 447, 455	
\keys_set_known:nn	20, 32
\l_keys_value_tl	43, 159
<b>L</b>	
\label	381
\labelenumi	34
\labelenumii	35
\labelenumiii	36
\labelenumiv	37
\labelitemfont	38, 39, 40, 41, 42
\labelitemi	38
\labelitemii	39
\labelitemiii	40
\labelitemiv	41
\labelsep	29, 33, 46, 48
\labelwidth	47, 48
\Large	21, 90
\large	265
\leavevmode	77, 264
\leftmargin	51, 59, 66
\leftmargini	43, 47, 51
\leftmarginii	44, 59
\leftmarginiii	45, 66
\leftskip	173, 183
legacy commands:	
\legacy_if:nTF	
129, 149, 231, 301, 317, 336, 342, 353	
\legacy_if_gset_false:n	339, 352
<b>M</b>	
\makeatletter	145
\maketitle	139
\mathcolor	5, 11
\mbox	359
\medskip	263, 269
\mode	34, 11
mode commands:	
\mode_if_horizontal:TF ..	31, 319, 343
\mode_if_horizontal_p:	25
\mode_if_math:TF	358
\mode_if_vertical_p:	26
\mode_leave_vertical:	34, 338, 361
\month	5
msg commands:	
\msg_error:nnn	118, 166
\msg_fatal:nn	15
\g_msg_module_name_prop	5
\g_msg_module_type_prop	6
\msg_new:nnn	326
\msg_new:nnnn	9, 224
\msg_warning:nn	322
<b>N</b>	
\NeedsDocumentMetadata	17
\NewCommandCopy	
.....	4, 5, 6, 280, 346, 366, 388, 397
\newcounter	20, 72, 73, 74, 132
\NewDocumentCommand	5,
10, 11, 34, 39, 62, 87, 92, 106, 118,	
128, 139, 140, 160, 196, 202, 219, 229	
\NewDocumentEnvironment	9,
69, 113, 123, 125, 133, 148, 236,	
237, 238, 239, 246, 257, 263, 432, 452	
\NewEnvironmentCopy	244, 394
\newlabel	394
\newlength	139, 140
\NewTaggingSocket	127
\NewTaggingSocketPlug	128
\NewTemplateType	
.....	15, 44, 89, 93, 94, 183, 218, 265
\newtheorem	388
\newwrite	153
\nobreakspace	137
\noindent	31, 157, 217, 240, 287
\normalfont	42, 50, 223, 265, 384
\normalsize	149
\number	20, 22
\numberline	117
<b>O</b>	
\obeyedline	17, 59
\only	43, 128
onlyenv (env.)	133
\onslide	39, 202
opacity commands:	
\opacity_begin:n	43, 45
\opacity_end:	47, 47
\opacity_select:n	207
opacity internal commands:	
\__opacity_backend:nnn	74, 75
\__opacity_backend_begin:n ..	46, 51
\__opacity_backend_end:	48, 78
\l_opacity_backend_fill_tl	60
\__opacity_backend_reset:	86



<code>\skip_vertical:n</code> . . . . .	33, 96, 98, 102, 104, 108, 110, 114, 116, 358, 359
<code>\l_tmpa_skip</code> . . . . .	355, 356, 358, 359
socket commands:	
<code>\socket_use:n</code> . . . . .	364
<code>\space</code> . . . . .	19, 21
<code>\stdcolor</code> . . . . .	4
<code>\stdemph</code> . . . . .	331
<code>\stdincludegraphics</code> . . . . .	365
<code>\stditem</code> . . . . .	280, 284, 285
<code>\stdmathcolor</code> . . . . .	4
<code>\stdnewtheorem</code> . . . . .	388
<code>stdquotation (env.)</code> . . . . .	236
<code>stdquote (env.)</code> . . . . .	236
<code>\stdtextbf</code> . . . . .	331
<code>\stdtextcolor</code> . . . . .	4
<code>\stdtextit</code> . . . . .	331
<code>\stdtextmd</code> . . . . .	331
<code>\stdtextnormal</code> . . . . .	331
<code>\stdtextrm</code> . . . . .	331
<code>\stdtextsc</code> . . . . .	331
<code>\stdtextsf</code> . . . . .	331
<code>\stdtextsl</code> . . . . .	331
<code>\stdtexttt</code> . . . . .	331
<code>\stdtextup</code> . . . . .	331
<code>stdverse (env.)</code> . . . . .	236
<code>\stepcounter</code> . . . . .	20
str commands:	
<code>\str_clear:N</code> . . . . .	20, 30, 31
<code>\str_if_empty:N</code> . . . . .	88
<code>\str_if_empty_p:N</code> . . . . .	42
<code>\str_if_eq:nnTF</code> . . . . .	17, 59, 85, 108, 110, 154
<code>\str_if_eq_p:nn</code> . . . . .	6, 7, 23, 300
<code>\str_new:N</code> . . . . .	9, 11, 12, 15, 137
<code>\str_put_right:Nn</code> . . . . .	139, 175
<code>\str_replace_all:Nnn</code> . . . . .	20, 22, 100
<code>\str_set:Nn</code> . . . . .	18, 26, 113, 115, 119
<code>\str_set_eq:NN</code> . . . . .	166
<code>\string</code> . . . . .	394
<code>\subsection</code> . . . . .	72, 81
<code>\subsubsection</code> . . . . .	72, 81
<code>\subsubtitle</code> . . . . .	34
sys commands:	
<code>\sys_if_engine luatex:TF</code> . . . . .	183
<code>\sys_if_engine luatex_p:</code> . . . . .	56, 83
<code>\sys_if_engine opentype:TF</code> . . . . .	179
<code>\sys_if_engine pdftex_p:</code> . . . . .	55, 82
<code>\sys_if_engine xetex:TF</code> . . . . .	65
<code>\sys_if_engine xetex_p:</code> . . . . .	57, 84
<b>T</b>	
<code>\tabbingsep</code> . . . . .	29
<code>\tabcolsep</code> . . . . .	26
<code>table (env.)</code> . . . . .	121
<code>\tablename</code> . . . . .	132
<code>\tableofcontents</code> . . . . .	160
tag commands:	
<code>\tag_get:n</code> . . . . .	407
<code>\tag_mc_begin:n</code> . . . . .	173, 180, 414
<code>\tag_mc_end:</code> . . . . .	171, 178, 420
<code>\tag_resume:n</code> . . . . .	170, 419
<code>\tag_struct_begin:n</code> . . . . .	91, 132, 172, 406
<code>\tag_struct_end:</code> . . . . .	92, 139, 179, 410
<code>\tag_suspend:n</code> . . . . .	181, 415
tag internal commands:	
<code>\g__tag_title_author_tl</code> . . . . .	19
<code>\g__tag_title_title_tl</code> . . . . .	31
<code>\tagpdfparaOff</code> . . . . .	61
<code>\tagpdfsetup</code> . . . . .	186, 202
talk internal commands:	
<code>__talk_action:N</code> . . . . .	55, 55
<code>__talk_action_end:N</code> . . . . .	55, 56
<code>__talk_action_alert:N</code> . . . . .	57, 57, 121, 126
<code>__talk_action_begin:n</code> . . . . .	11, 79, 125, 140, 143, 149, 151, 247, 259, 311, 397
<code>__talk_action_begin_aux:n</code> . . . . .	140, 158, 161
<code>__talk_action_end:</code> . . . . .	31, 26, 84, 130, 140, 145, 150, 181, 253, 276, 307, 405
<code>__talk_action_invisible:N</code> . . . . .	63, 63, 179
<code>__talk_action_invisible_end:N</code> . . . . .	63, 69
<code>__talk_action_only:N</code> . . . . .	79, 79, 134
<code>__talk_action_only_end:N</code> . . . . .	79, 84, 135
<code>\l__talk_action_spec_str</code> . . . . .	149, 289, 386
<code>__talk_action_uncover:N</code> . . . . .	96, 96
<code>__talk_action_uncover_end:N</code> . . . . .	96, 102
<code>__talk_action_visible:N</code> . . . . .	63, 71
<code>__talk_action_visible_end:N</code> . . . . .	63, 77
<code>\l__talk_aspect_ratio_str</code> . . . . .	106, 155
<code>\l__talk_cnt_reset_seq</code> . . . . .	75, 76, 77, 118, 133, 138, 146
<code>__talk_cnt_restore:</code> . . . . .	86, 131, 136
<code>__talk_cnt_save:</code> . . . . .	77, 131, 131
<code>__talk_column_align_bottom:n</code> . . . . .	50, 50
<code>__talk_column_align_center:n</code> . . . . .	50, 52
<code>__talk_column_align_top:n</code> . . . . .	50, 67
<code>\l__talk_column_alignment_tl</code> . . . . .	28, 86
<code>\l__talk_columns_wd_tl</code> . . . . .	5, 14, 15
<code>__talk_decode_action:n</code> . . . . .	87, 96, 96
<code>__talk_decode_action:w</code> . . . . .	96, 98, 103
<code>\l__talk_decode_action_str</code> . . . . .	12, 20, 113, 167, 174
<code>\l__talk_decode_actions_bool</code> . . . . .	13, 27, 169, 175
<code>\l__talk_decode_actions_clist</code> . . . . .	13
<code>\l__talk_decode_actions_str</code> . . . . .	13, 31

\l__talk_decode_arg_str .....	\l__talk_frame_bool ....
..... 9, 26, 32, 119, 167	141, 385, 401
\__talk_decode_check:n . 132, 179, 179	\g__talk_frame_int .....
\__talk_decode_check:nw 179, 186, 189	..... 14, 52, 68, 260, 388, 393, 400
\__talk_decode_check_range:nnn .	\__talk_frame_notag:n ... 41, 412, 412
..... 179, 195, 196, 208	\__talk_frame_overprint: .....
\__talk_decode_check_single:nn .	..... 258, 258, 270, 273, 277,
..... 179, 192, 199	290, 292, 293, 296, 298, 305, 307, 312
\__talk_decode_mode:n .... 46, 55, 55	\__talk_frame_process:nn .....
\__talk_decode_mode:nn ... 78, 81, 83	..... 398, 398, 429, 438, 449, 457
\__talk_decode_mode:w .... 55, 64, 70	\g__talk_frame_struct_int 56, 71, 407
\__talk_decode_mode_aux:n .....	\g__talk_frame_subtitle_tl 3, 13, 76
55	\__talk_frame_tag:n .... 37, 404, 404
\__talk_decode_overlay.:nw .... 123	\g__talk_frame_tag_bool .....
\__talk_decode_overlay_aux:nNn .	..... 46, 386, 408, 416
..... 123, 147, 150, 151	\l__talk_frame_tagging_str ....
\__talk_decode_overlay_offset:nNn	..... 17, 18, 20, 22, 34, 149
..... 123, 155, 160, 170, 173	\__talk_frame_title:n .... 15, 38, 44
\__talk_decode_overlay_offset:nNnN	\l__talk_frame_title_bool . 106, 422
..... 123, 159, 162, 171	\__talk_frame_title_tagged:n ...
\__talk_decode_overlays:nN .....	..... 15, 47, 51
..... 123, 126, 134, 140, 177	\g__talk_frame_title_tl .....
\__talk_decode_overlays:nn ....	..... 3, 8, 58, 75, 254, 437
..... 89, 108, 115, 123, 123	\l__talk_frame_verb_bool .....
\l__talk_decode_overlays_bool ..	..... 44, 387, 428, 436, 448, 456
..... 3, 6, 24, 28, 44, 61, 165, 172	\l__talk_frametitle_after_skip .
\l__talk_decode_overlays_clist .. 10	..... 25, 41
\l__talk_decode_overlays_str ...	\l__talk_frametitle_before_skip
..... 10, 30, 42, 88	..... 26, 32
\__talk_decode_parse:n . 5, 16, 16, 163	\l__talk_frametitle_color_tl ...
\__talk_decode_parse:w . 16, 36, 37, 48	..... 27, 34, 35
\__talk_decode_parse_auxi:n ....	\l__talk_frametitle_font_tl .. 28, 36
..... 16, 17, 18	\l__talk_header_bg_tl .....
\__talk_decode_parse_auxii:n ...	218
..... 16, 32, 35	\l__talk_header_fg_tl .....
\l__talk_decode_pure_bool .....	218
..... 7, 29, 43, 93, 112	\l__talk_header_frametitle_bool 218
\l__talk_decode_step_bool .....	\l__talk_header_ht_dim .....
..... 8, 125, 127, 146	218
\l__talk_float_alignment_tl ....	\l__talk_header_left_skip .....
..... 92, 104, 105, 106, 112, 118	218
\l__talk_fontsize_dim .. 106, 136, 141	\l__talk_header_right_skip .... 218
\l__talk_footelem_color_tl .... 183	\__talk_header_tag_begin:n ....
\l__talk_footelem_font_tl .... 183	..... 53, 168, 168, 175
\l__talk_footelem_left_skip .... 183	\__talk_header_tag_end: . 64, 168, 176
\l__talk_footelem_right_skip ... 183	\__talk_if_overlay:n .....
\l__talk_footer_bg_tl .....	..... 3, 7, 12, 13, 13, 23, 31, 32,
265	34, 45, 130, 198, 211, 221, 350, 369, 384
\l__talk_footer_fg_tl .....	\__talk_item_parse_spec:n .....
265	..... 278, 291, 295, 296
\l__talk_footer_font_tl .....	\__talk_item_parse_spec:w .....
265	..... 278, 288, 294
\l__talk_footer_left_skip .....	\__talk_label:n .....
265	381, 385, 388
\l__talk_footer_order_clist .... 265	\__talk_latex_frame:n .. 26, 397, 397
\l__talk_footer_right_skip .... 265	\l__talk_list_end_tl .....
\l__talk_footer_sep_tl .....	..... 302, 308, 314, 325, 350
265	
\l__talk_frame_alignment_tl ....	
..... 89, 94, 148, 158	



\__talk_metadata_name:n	306, 309, 314, 330, 330
\__talk_mode:n	3
\__talk_mode:nTF	3, 12
\l_talk_mode_str	7, 60, 85, 106
\c_talk_modes_clist	50, 57
\__talk_onslide:n	202, 204, 207, 236
\g_talk_onslide_tl	79, 83, 170, 209, 210, 214, 218
\__talk_opacity_begin:n	40, 40, 53, 66, 67, 74, 75, 94, 99, 213
\__talk_opacity_end:	40, 45, 56, 70, 78, 103, 192, 215
\__talk_opacity_reapply:	50, 50, 55
\g_talk_opacity_seq	38, 42, 48, 52
\l_talk_overlay_all_bool	139, 155, 157, 183
\__talk_overlay_arg:n	3, 11, 107, 114, 118, 125, 133
\__talk_overprint_begin:n	239, 239, 247, 264
\__talk_overprint_check_ht:n	263, 312, 314, 323
\l_talk_overprint_int	257, 261, 267
\__talk_overprint_save_ht:	263, 268, 288
\__talk_pagecolor:n	43, 48, 49, 52
\c_talk_paper_height_dim	143
\c_talk_paper_width_dim	143
\g_talk_pauses_int	10, 4, 74, 130, 176, 234, 235, 236
\l_talk_saved_action_str	136, 166, 188
\l_talk_saved_actions_bool	136, 168, 190
\l_talk_saved_overlays_bool	136, 164, 185
\__talk_sect_tag:nn	127, 129, 130
\g_talk_section_tl	66
\l_talk_section_tl	66
\l_talk_shuffle_skip	17, 20, 22, 34
\__talk_shuffle_skip:n	18, 18, 43, 47
\__talk_slide:nn	9, 9, 402
\__talk_slide_align_bottom:n	94, 94
\__talk_slide_align_center:n	94, 100
\__talk_slide_align_stretch:n	94, 106
\__talk_slide_align_top:n	94, 112
\__talk_slide_aux:n	9, 45, 56
\__talk_slide_begin:	33, 72, 72
\l_talk_slide_box	4, 78, 90
\g_talk_slide_continue_bool	3, 27, 30, 36, 40, 85, 205, 213, 219, 224
\__talk_slide_end:	49, 72, 81
\g_talk_slide_int	5, 8, 25, 29, 201, 204, 210, 212, 217
\g_talk_subsection_tl	66
\l_talk_subsection_tl	66, 109
\g_talk_subsubsection_tl	66
\l_talk_subsubsection_tl	66, 111
\__talk_textcmd_equiv:n	331, 352, 356
\l_talk_titlelem_after_skip	44
\l_talk_titlelem_before_skip	44
\l_talk_titlelem_color_tl	44
\l_talk_titlelem_font_tl	44
\l_talk_titlelem_tag_begin_tl	44
\l_talk_titlelem_tag_end_tl	44
\l_talk_titlepage_alignment_tl	94
\l_talk_titlepage_framestyle_tl	94
\l_talk_titlepage_order_clist	94
\__talk_tmp:w	103, 103, 146, 155
\l_talk_tmp_box	14, 24, 60, 73, 82, 87, 104, 242, 252, 261, 274, 274, 279, 283, 285, 293, 299, 320
\l_talk_tmp_tl	12, 18, 21, 23, 48, 52, 53, 101, 105, 303, 305, 306
\__talk_toc_aux:nnnn	166, 167, 170, 180, 189
\__talk_toc_dest:n	166, 193, 196
\__talk_toc_dest:w	166, 198, 201
\__talk_toc_level:nnnn	166, 194, 212
\l_talk_uncover_hidden_fp	89
\__talk_wallpaper_hrulerule:Nnn	241, 288, 336, 336
talk/sec/title	127
\temporal	219
TeX and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:	
\@arabic	6, 7, 78, 79, 80, 199, 390
\@author	3, 18, 19
\@auxout	303, 392
\@bsphack	383
\@caption	33, 143
\@capttype	113
\@contentsline@destination	54, 198, 220, 223, 226, 229
\@currentHref	399
\@currentlabel	396
\@currentlabelname	398
\@currenvir	349
\@date	3, 25
\@definecounter	141
\@endparpenalty	361
\@esphack	386
\@evenfoot	356, 371, 382
\@evenhead	355, 370, 381
\@framenum	388
\@ignore	31
\@ignoretrue	90

<code>\@inmatherr</code>	349	<code>\protected@write</code>	392
<code>\@input</code>	147	<code>\ps@plain</code>	348
<code>\@institute</code>	3, 37	<code>\ps@talk</code>	348
<code>\@itempenalty</code>	328	<code>\ps@wallpaper</code>	348
<code>\@kernel@reserved@label@data</code>	400	<code>\set@color</code>	61
<code>\@listI</code>	56	<code>\std@definecounter</code>	141
<code>\@listi</code>	49, 56	tex commands:	
<code>\@listii</code>	57	<code>\tex_currentgrouplevel:D</code>	304, 305
<code>\@listiii</code>	64	<code>\tex_fontdimen:D</code>	61
<code>\@makecaption</code>	150	<code>\tex_hsize:D</code>	22, 33
<code>\@makefnmark</code>	158	<code>\tex_lastskip:D</code>	20
<code>\@makefntext</code>	33, 154	<code>\tex_setbox:D</code>	20, 31
<code>\@mpfootins</code>	30	<code>\tex_textfont:D</code>	61
<code>\@nobreakfalse</code>	157	<code>\tex_unskip:D</code>	29
<code>\@noitemerr</code>	342	<code>\tex_vbox:D</code>	20, 31
<code>\@oddfnfoot</code>	354, 356, 365, 371, 380, 382	<code>\tex_vrule:D</code>	39
<code>\@oddfnhead</code>	350, 355, 360, 370, 375, 381	text commands:	
<code>\@outerparskip</code>	359	<code>\text_purify:n</code>	58, 101, 129
<code>\@parboxrestore</code>	76, 147	<code>\text_titlecase_first:n</code>	134
<code>\@setminipage</code>	148	<code>\textasteriskcentered</code>	40
<code>\@shortauthor</code>	3	<code>\textbf</code>	331
<code>\@shortdate</code>	3	<code>\textbullet</code>	38
<code>\@shortinstitute</code>	3	<code>\textcolor</code>	6, 11
<code>\@shortsubtitle</code>	3	<code>\textendash</code>	39
<code>\@shorttitle</code>	3	<code>\textheight</code>	87
<code>\@starttoc</code>	142, 163	<code>\textit</code>	331
<code>\@subtitle</code>	3, 42	<code>\textmd</code>	331
<code>\@title</code>	3, 30, 31	<code>\textnormal</code>	331
<code>\@totalframes</code>	392	<code>\textperiodcentered</code>	41
<code>\c@figure</code>	132	<code>\textrm</code>	331
<code>\c@frame</code>	388	<code>\textsc</code>	331
<code>\c@page</code>	199	<code>\textsf</code>	331
<code>\c@pauses</code>	4	<code>\textsl</code>	331
<code>\c@section</code>	78	<code>\texttt</code>	331
<code>\c@slide</code>	5	<code>\textup</code>	331
<code>\c@subsection</code>	79	<code>\textwidth</code>	29, 8, 15, 16, 74, 75, 263
<code>\c@subsubsection</code>	80	<code>\theenumi</code>	34
<code>\c@table</code>	132	<code>\theenumii</code>	35
<code>\check@mathfonts</code>	198	<code>\theenumiii</code>	36
<code>\currentgrouplevel</code>	56	<code>\theenumiv</code>	37
<code>\fnum@figure</code>	132	<code>\thefigure</code>	132
<code>\fnum@table</code>	132	<code>\theframe</code>	388
<code>\Gm@bmargin</code>	291	<code>\thepage</code>	6, 199, 397
<code>\Gm@lmargin</code>	225, 272, 338	<code>\thepauses</code>	4
<code>\Gm@rmargin</code>	227, 273, 321	<code>\thesection</code>	72
<code>\Gm@tmargin</code>	224	<code>\theslide</code>	5
<code>\hb@xt@</code>	158	<code>\thesubsection</code>	72
<code>\if@minipage</code>	148	<code>\thesubsubsection</code>	72
<code>\ifmeasuring@</code>	13	<code>\thetable</code>	132
<code>\ignorespaces</code>	31	<code>\thispagestyle</code>	129
<code>\l@section</code>	166	<code>\tiny</code>	271
<code>\l@subsection</code>	166	<code>\title</code>	15
<code>\l@subsubsection</code>	166	tl commands:	
<code>\on@line</code>	333	<code>\tl_clear:N</code>	109, 111, 308

