

ltx-talk – A class for typesetting presentations*

Joseph Wright[†]

Released 2026-04-30

Contents

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

*This file describes v0.5.0, last revised 2026-04-30.

[†]E-mail: joseph@texdev.net

1	ltx-talk-frame implementation	18
1.1	Slides in frames	18
1.2	Counters	21
1.3	Frame options	22
1.4	Tagging for headers	23
1.5	Wallpaper	23
1.6	The <code>frame</code> environment	27
V	ltx-talk-frame – The structure of frames	32
1	ltx-talk-frame-structure implementation	32
1.1	Columns	32
1.2	Floats	34
1.3	Footnotes	36
VI	ltx-talk-mode – Modes	38
1	ltx-talk-mode implementation	38
VII	ltx-talk-overlay – Overlays	39
1	ltx-talk-overlay implementation	39
1.1	Utilities	39
1.2	Opacity utilities	40
1.3	Action commands and environments	40
1.4	Non-action commands and environments	44
1.5	Fixed-size areas	46
1.6	Adding overlays to existing commands	48
1.7	Overlay patching of third-party environments	50
VIII	ltx-talk-required – “Required” definitions	51
1	ltx-talk-required implementation	51
1.1	Standard design settings	51
1.2	List support	52
IX	ltx-talk-structure – Structural commands	53
1	ltx-talk-structure implementation	53
1.1	Frame title	53
1.2	Headings	54
1.3	Table of contents	58
1.4	Block environments	60
1.5	Lists	61
1.6	Theorems, <i>etc.</i>	65

X	ltx-talk-title – Title pages	66
1	ltx-talk-title implementation	66
	Index	70

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} {2026-04-30} {0.5.0}
4 {A class for typesetting presentations}
    Get the right type of message.
5 \prop_gput:Nnn \g_msg_module_name_prop { talk } { ltx-talk }
6 \prop_gput:Nnn \g_msg_module_type_prop { talk } { Class }
    Require the latest LATEX structures.
7 \IfFormatAtLeastF { 2025-11-01 }
8 {
9     \msg_new:nnnn { ltx-talk } { kernel-too-old }
10     { The~ltx-talk~class~requires~LaTeX~2025-11-01~or~later. }
11     {
12         You~have~tried~to~use~the~ltx-talk~class~with~a~LaTeX~kernel~release~
13         prior~to~2025-11-01;~the~required~functionality~is~missing.
14     }
15     \msg_fatal:nn { ltx-talk } { kernel-too-old }
16 }
17 \NeedsDocumentMetadata
    Warn if not an engine that is tested.
18 \bool_lazy_or:nnF
19 { \sys_if_engine luatex_p: }
20 { \sys_if_engine pdftex_p: }
21 {
22     \msg_new:nnn { ltx-talk } { unsupported-engine }
23     {
24         The~engine~"\c_sys_engine_str"~
25         is~not~supported~by~the~ltx-talk~class.
26     }
27     \msg_warning:nn { ltx-talk } { unsupported-engine }
28 }
```

1.2 Additions for expl3

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

```

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

```

Some things from `xbox` that would be useful.

```

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

```

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

```

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

```

```

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

```

1.3 Extra variants

```

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

```

1.4 Scratch space

__talk_tmp:w For one-off processing.

```

114 \cs_new_protected:Npn \__talk_tmp:w { }

```

(End of definition for __talk_tmp:w.)

\l__talk_tmp_box

```

115 \box_new:N \l__talk_tmp_box

```

(End of definition for \l__talk_tmp_box.)

\l__talk_tmp_tl

116 \tl_new:N \l__talk_tmp_tl

(End of definition for \l__talk_tmp_tl.)

1.5 Option handling

\l__talk_aspect_ratio_str

\l__talk_fontsize_dim

\l__talk_frame_title_bool

\l__talk_mode_str

117 \keys_define:nn { talk }

118 {

119 aspect-ratio .str_set:N =

120 \l__talk_aspect_ratio_str ,

121 font-size .dim_set:N =

122 \l__talk_fontsize_dim ,

123 frame-title-arg .bool_set:N =

124 \l__talk_frame_title_bool ,

125 handout .code:n =

126 { \str_set:Nn \l__talk_mode_str { handout } } ,

127 handout .value_forbidden:n = true ,

128 mode .choices:nn =

129 { handout , projector }

130 { \str_set:NV \l__talk_mode_str \l_keys_choice_tl }

131 }

132 \str_new:N \l__talk_mode_str

(End of definition for \l__talk_aspect_ratio_str and others.)

Scope for options.

133 \keys_define:nn { talk }

134 {

135 aspect-ratio .usage:n = load ,

136 font-size .usage:n = load ,

137 frame-title-arg .usage:n = load ,

138 mode .usage:n = load

139 }

Compatibility keys for classical font size setting.

140 \clist_map_inline:nn { 10pt , 11pt , 12pt }

141 {

142 \keys_define:nn { talk }

143 {

144 #1 .meta:n = { font-size = #1 } ,

145 #1 .value_forbidden:n = true ,

146 #1 .usage:n = load

147 }

148 }

Initial values.

149 \keys_set:nn { talk }

150 {

151 aspect-ratio = 16:9 ,

152 font-size = 11pt ,

153 frame-title-arg = false ,

154 mode = projector

155 }

156 \ProcessKeyOptions [talk]

1.6 Setting up

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

```

157 \file_if_exist_input:nF { size \dim_to_decimal:n \l__talk_fontsize_dim .clo }
158 {
159   \file_input:n { size10.clo }
160   \RequirePackage { relsize }
161   \hook_gput_code:nne { begindocument } { talk }
162   { \exp_not:N \relsize { \fp_eval:n { \l__talk_fontsize_dim / 10pt } } } }
163 }

```

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`.

```

164 \dim_const:Nn \c__talk_paper_height_dim { 100mm }
165 \use:e
166 {
167   \cs_set_protected:Npn \exp_not:N \__talk_tmp:w
168     #1 \tl_to_str:n { : } #2 \tl_to_str:n { : } #3 \exp_not:N \q_stop
169   {
170     \dim_const:Nn \exp_not:N \c__talk_paper_width_dim
171     {
172       \exp_not:N \fp_to_dim:n
173       { (#1 / #2) * \exp_not:N \c__talk_paper_height_dim }
174     }
175   }
176   \exp_not:N \__talk_tmp:w \l__talk_aspect_ratio_str
177   \tl_to_str:n { : } 100 \exp_not:N \q_stop
178 }
179 \use:e
180 {
181   \exp_not:N \RequirePackage
182   [
183     papersize =
184     {
185       \dim_use:N \c__talk_paper_width_dim ,
186       \dim_use:N \c__talk_paper_height_dim
187     } ,
188     tmargin    = 10mm ,
189     bmargin    = 8mm ,
190     lmargin    = 10mm ,
191     rmargin    = 10mm ,
192     headheight = 10mm ,
193     headsep    = 2mm ,
194     footskip   = 6mm
195   ]
196   { geometry }
197 }

```

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

Turn off justification

```

198 \raggedright

```

1.7 Math support

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

```
199 \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.

```
200 \sys_if_engine_opentype:TF
201 {
202   \RequirePackage { fontspec }
203   \RequirePackage { mathtools }
204   \sys_if_engine luatex:TF
205   {
206     \RequirePackage { lua-unicode-math }
207     \tagpdfsetup { math / mathml / luamml / load = true }
208   }
209   { \RequirePackage { unicode-math } }
210   \setmainfont { NewCMSans10-Regular.otf }
211   \setsansfont { NewCMSans10-Regular.otf }
212   \setmathfont { NewCMSansMath-Regular.otf }
213 }
214 {
215   \RequirePackage { sansmathfonts }
216   \RequirePackage [ nomath ] { lmodern }
217   \cs_set_eq:NN \rmdefault \sfdefault
218 }
```

1.9 Text scripts

Newer kernel releases allow us to use real text sub- and superscripts: we set up the appropriate data here.

```
\textsubscript@offset Offset values as in ConTeXt, no additional spacing added after script items.
\textsubscript@space
\textsuperscript@offset
\textsuperscript@space
219 \cs_set_nopar:Npn \textsubscript@offset { 0.48ex }
220 \cs_set_nopar:Npn \textsubscript@space { }
221 \cs_set_nopar:Npn \textsuperscript@offset { 0.86ex }
222 \cs_set_nopar:Npn \textsuperscript@space { }
```

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

1.10 Hyperlinks

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

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

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

A requirement.

```
224 \RequirePackage { hyperref }
```

```
225 \hypersetup { hidelinks }
```

1.11 Tagging

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

```
226 \tagpdfsetup
```

```
227 {
```

```
228   role / user-NS = ltx-talk      ,
```

```
229   role / new-tag = frame / Sect  ,
```

```
230   role / new-tag = frametitle / H4
```

```
231 }
```

```
232 \</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     \mode_leave_vertical:
33     \group_begin:
34     \__talk_if_overlay:nT {#1}
35     {
36         \IfNoValueTF {#2}
37         { \color_select:e {#3} }
38         { \color_select:ne {#2} {#3} }
39     }
40     #4
41     \group_end:
42 }

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

(End of definition for \set@color and \reset@color. These functions are documented on page ??.)

1.3 Color definition

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

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

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

1.4 Semantic colors

Pick up the standard colors from beamer.

```
71 \DeclareColor { alert } [ RGB ] { 200 , 0 , 0 }
72 \DeclareColor { example } { green!50!black }
73 \DeclareColor { structure } [ rgb ] { 0.2 , 0.2 , 0.7 }
74 \</class>
```

Part III

ltx-talk-decode – Decoding overlay specs

1 ltx-talk-decode implementation

Start the DocStrip guards.

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

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

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

(End of definition for \l__talk_decode_overlays_bool.)

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

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

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

`\l__talk_decode_modes_bool` Tracks whether at least one mode was given with no overlays: this means that the specification selects only included modes.

```
7 \bool_new:N \l__talk_decode_modes_bool
```

(End of definition for \l__talk_decode_modes_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 16 \cs_new_protected:Npn \__talk_decode_parse:n #1
\__talk_decode_parse_auxii:n 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_false:N \l__talk_decode_modes_bool
                             30       \exp_args:No \__talk_decode_parse_auxii:n { \l__talk_decode_arg_str }
                             31     }
                             32 }

```

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

```

33 \cs_new_protected:Npn \__talk_decode_parse_auxii:n #1
34 {
35   \bool_set_false:N \l__talk_decode_step_bool
36   \__talk_decode_parse:w #1 | \q_recursion_tail | \q_recursion_stop
37   \bool_if:NT \l__talk_decode_step_bool
38   {
39     \legacy_if:nF { measuring@ }
40     { \int_gincr:N \g__talk_pauses_int }
41   }
42 }

```

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

```

43 \cs_new_protected:Npn \__talk_decode_parse:w #1 |
44 {
45   \quark_if_recursion_tail_stop_do:nn {#1}
46   {
47     \bool_lazy_and:nnT
48       { \str_if_empty_p:N \l__talk_decode_overlays_str }
49       { ! \l__talk_decode_modes_bool }
50       { \bool_set_true:N \l__talk_decode_overlays_bool }

```

```

51     }
52     \exp_args:Ne \__talk_decode_mode:n
53     { \tl_trim_spaces:n {#1} }
54     \__talk_decode_parse:w
55 }

```

(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.

```

56 \clist_const:Ne \c__talk_modes_clist
57 {
58     \tl_to_str:n { article } ,
59     \tl_to_str:n { handout } ,
60     \tl_to_str:n { projector }
61 }

```

(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 *. Notice that if we get a hit here for the mode test, there is no overlay part: we therefore have a spec that excludes other modes.

```

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

```

```

95     \__talk_decode_action:n {#2}
96     \str_if_empty:NT \l__talk_decode_overlays_str
97     { \__talk_decode_overlays:nn { overlays } { * } }
98   }
99   {
100     \tl_if_blank:nT {#2}
101     { \bool_set_true:N \l__talk_decode_modes_bool }
102   }
103 }

```

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

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

__talk_decode_action:w

```

104 \cs_new_protected:Npe \__talk_decode_action:n #1
105 {
106   \exp_not:N \__talk_decode_action:w
107   #1 \tl_to_str:n { @ @ } \exp_not:N \q_stop
108 }
109 \use:e
110 {
111   \cs_new_protected:Npn \exp_not:N \__talk_decode_action:w
112   #1 \tl_to_str:n { @ } #2 \tl_to_str:n { @ } #3 \exp_not:N \q_stop
113 }
114 {
115   \tl_if_blank:nTF {#2}
116   {
117     \str_if_empty:NTF \l__talk_decode_action_str
118     { \__talk_decode_overlays:nn { overlays } {#1} }
119     {
120       \msg_error:nnV { talk } { misplaced-action-spec }
121       \l__talk_decode_arg_str
122     }
123   }
124   {
125     \cs_if_exist:cTF { __talk_action_ #1 :N }
126     {
127       \str_if_empty:NTF \l__talk_decode_action_str
128       {
129         \str_set:Nn \l__talk_decode_action_str {#1}
130         \tl_if_blank:nF {#2}
131         { \__talk_decode_overlays:nn { actions } {#2} }
132       }
133       {
134         \msg_error:nnV { talk } { duplicate-action-spec }
135         \l__talk_decode_arg_str
136       }
137     }
138     {
139       \msg_error:nnV { talk } { bad-action-spec }
140       \l__talk_decode_arg_str
141     }
142   }
143 }

```

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

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

```

__talk_decode_overlays:nn
__talk_decode_overlays:nN
  \@_decode_overlay_+:nw
__talk_decode_overlay_.:nw
  __talk_decode_overlay_aux:nN
  __talk_decode_overlay_offset:nNn
  __talk_decode_overlay_offset:nNn
144 \cs_new_protected:Npn __talk_decode_overlays:nn #1#2
145 {
146   \str_clear:c { l__talk_decode_ #1 _str }
147   __talk_decode_overlays:nN {#1} #2 \q_recursion_tail \q_recursion_stop
148   __talk_decode_check:n {#1}
149 }
150 \cs_new_protected:Npn __talk_decode_overlays:nN #1#2
151 {
152   \quark_if_recursion_tail_stop:N #2
153   \cs_if_exist_use:cF { __talk_decode_overlay_ #2 :nw }
154   {
155     \str_put_right:cn { l__talk_decode_ #1 _str } {#2}
156     __talk_decode_overlays:nN
157   }
158   {#1}
159 }
160 \cs_new_protected:cpn { __talk_decode_overlay_+:nw } #1
161 {
162   \bool_set_true:N \l__talk_decode_step_bool
163   __talk_decode_overlay_aux:nNn {#1} 1
164 }
165 \cs_new_protected:cpn { __talk_decode_overlay_.:nw } #1
166 { __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 `(`.

```

167 \cs_new_protected:Npn __talk_decode_overlay_aux:nNn #1#2#3
168 {
169   \quark_if_recursion_tail_stop_do:Nn #3
170   {
171     __talk_decode_overlay_offset:nNn {#1} #2 { 0 }
172     \q_recursion_tail \q_recursion_stop
173   }
174   \token_if_eq_meaning:NNTF #3 ( % )
175   { __talk_decode_overlay_offset:nNn {#1} #2 { } }
176   { __talk_decode_overlay_offset:nNn {#1} #2 { 0 } #3 }
177 }

```

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.

```

178 \cs_new_protected:Npn __talk_decode_overlay_offset:nNnN #1#2#3#4
179 {
180   \quark_if_recursion_tail_stop_do:Nn #4
181   {
182     \msg_error:nnV { talk } { bad-action-spec }
183     \l__talk_decode_arg_str

```

```

184     } % (
185     \token_if_eq_meaning:NNTF #4 )
186     { \__talk_decode_overlay_offset:nNn {#1} #2 {#3} }
187     { \__talk_decode_overlay_offset:nNnN {#1} #2 {#3#4} }
188 }

```

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

```

189 \cs_new_protected:Npn \__talk_decode_overlay_offset:nNn #1#2#3
190 {
191   \str_put_right:ce { l__talk_decode_ #1 _str }
192   { \int_max:nn { 0 } { #3 + \g__talk_pauses_int + #2 } }
193   \__talk_decode_overlays:nN {#1}
194 }

```

(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.

```

195 \cs_new_protected:Npn \__talk_decode_check:n #1
196 {
197   \clist_set:cv { l__talk_decode_ #1 _clist } { l__talk_decode_ #1 _str }
198   \clist_if_in:cnTF { l__talk_decode_ #1 _clist } { * }
199   { \bool_set_true:c { l__talk_decode_ #1 _bool } }
200   {
201     \clist_map_inline:cn { l__talk_decode_ #1 _clist }
202     { \__talk_decode_check:nw {#1} 0 ##1 - - \q_stop }
203   }
204 }

```

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.

```

205 \cs_new_protected:Npn \__talk_decode_check:nw #1#2 - #3 - #4 \q_stop
206 {
207   \tl_if_empty:nTF {#4}
208   { \__talk_decode_check_single:nn {#1} {#2} }
209   {
210     \tl_if_blank:nTF {#3}
211     { \__talk_decode_check_range:nnn {#1} {#2} { \c_max_int } }
212     { \__talk_decode_check_range:nnn {#1} {#2} {#3} }
213   }
214 }
215 \cs_new_protected:Npn \__talk_decode_check_single:nn #1#2
216 {
217   \int_compare:nNnTF \g__talk_slide_int = {#2}
218   { \bool_set_true:c { l__talk_decode_ #1 _bool } }
219   {
220     \int_compare:nNnT {#2} > \g__talk_slide_int
221     { \bool_gset_true:N \g__talk_slide_continue_bool }
222   }
223 }

```

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.

```

224 \cs_set_protected:Npn \__talk_decode_check_range:nnn #1#2#3
225 {
226   \int_compare:nNnF \g__talk_slide_int > {#3}
227   {
228     \int_compare:nNnTF \g__talk_slide_int < {#2}
229     { \bool_gset_true:N \g__talk_slide_continue_bool }
230     {
231       \bool_set_true:c { l__talk_decode_ #1 _bool }
232       \bool_lazy_and:nnT
233       { \int_compare_p:nNn \g__talk_slide_int < {#3} }
234       { \int_compare_p:nNn {#3} < \c_max_int }
235       { \bool_gset_true:N \g__talk_slide_continue_bool }
236       \clist_map_break:
237     }
238   }
239 }

```

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

```

240 \msg_new:nnnn { talk } { bad-action-spec }
241 { Bad~overlay~specification~"#1". }
242 {
243   The~overlay~specification~given~doesn't~follow~the~pattern~described~in~
244   the~ltx-talk-manual:~it~has~been~ignored.
245 }
246 \msg_new:nnnn { talk } { duplicate-action-spec }
247 { Duplicate~action~in~overlay~specification~"#1". }
248 {
249   The~overlay~specification~contains~more~than~one~action:
250   ltx-talk-only~supports~a~single~action~in~one~overlay.
251 }
252 \msg_new:nnnn { talk } { misplaced-action-spec }
253 { Misplaced~action~in~overlay~specification~"#1". }
254 {
255   The~overlay~specification~contains~an~action~before~the~overlay~
256   part(s):~this~is~not~supported.
257 }
258 </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.

```
\l__talk_slide_continue_bool
```

- 3 `\bool_new:N \g__talk_slide_continue_bool`
- 4 `\bool_new:N \l__talk_slide_continue_bool`

(End of definition for `\g__talk_slide_continue_bool` and `\l__talk_slide_continue_bool`.)

```
\l__talk_slide_box
```

- 5 `\box_new:N \l__talk_slide_box`

(End of definition for `\l__talk_slide_box`.)

```
\g__talk_slide_int
```

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

```
\c@slide
```

We also provide L^AT_EX counter-style access.

```
\theslide
```

- 6 `\int_new:N \g__talk_slide_int`
- 7 `\cs_new_eq:NN \c@slide \g__talk_slide_int`
- 8 `\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.

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

```
\__talk_slide:nn
```

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.

```
\__talk_slide_aux:n
```

- 10 `\cs_new_protected:Npn __talk_slide:nn #1#2`

```

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

```

```

65     \use:e
66     {
67         \group_end:
68         \tl_retokenize:n {#1}
69     }
70 }

```

(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.

```

71 \AddToHook { enddocument / afterlastpage }
72 {
73     \property_record:ee { frame . \int_use:N \g__talk_frame_int }
74     { slides }
75 }

```

`\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).

```

76 \int_new:N \g__talk_frame_struct_int

```

(End of definition for `\g__talk_frame_struct_int`.)

```

\__talk_slide_begin:
\__talk_slide_end:

```

```

77 \cs_new_protected:Npn \__talk_slide_begin:
78 {
79     \int_gzero:N \g__talk_pauses_int
80     \tl_gclear:N \g__talk_frame_title_tl
81     \tl_gclear:N \g__talk_frame_subtitle_tl
82     \box_gclear:N \g__talk_footnote_box
83     \__talk_cnt_save:
84     \vbox_set:Nw \l__talk_slide_box
85     \tl_gclear:N \g__talk_onslide_tl
86 }
87 \cs_new_protected:Npn \__talk_slide_end:
88 {
89     \tl_use:N \g__talk_onslide_tl
90     \vbox_set_end:
91     \bool_if:NT \g__talk_slide_continue_bool
92     { \__talk_cnt_restore: }
93     \vbox_to_ht:nn { \textheight }
94     {
95         \use:c { __talk_slide_align_ \l__talk_frame_alignment_tl :n }
96         { \vbox_unpack_drop:N \l__talk_slide_box }
97         \box_if_empty:NF \g__talk_footnote_box
98         {
99             \footnoterule
100             \vbox_unpack_drop:N \g__talk_footnote_box
101         }
102     }
103     \clearpage
104 }

```

(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 105 \cs_new_protected:Npn \__talk_slide_align_bottom:n #1
    \__talk_slide_align_stretch:n 106 {
\__talk_slide_align_top:n 107   \skip_vertical:n { Opt~plus~1fil }
108   #1
109   \skip_vertical:n { Opt }
110 }
111 \cs_new_protected:Npn \__talk_slide_align_center:n #1
112 {
113   \skip_vertical:n { Opt~plus~0.5fil }
114   #1
115   \skip_vertical:n { Opt~plus~0.5fil }
116 }
117 \cs_new_protected:Npn \__talk_slide_align_stretch:n #1
118 {
119   \skip_vertical:n { Opt }
120   #1
121   \skip_vertical:n { Opt }
122 }
123 \cs_new_protected:Npn \__talk_slide_align_top:n #1
124 {
125   \skip_vertical:n { Opt }
126   #1
127   \skip_vertical:n { Opt~plus~1fil }
128 }

```

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

1.2 Counters

`\g__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`.

```

129 \seq_new:N \g__talk_cnt_reset_seq
130 \seq_gset_from_clist:Nn \g__talk_cnt_reset_seq
131 {
132   equation      ,
133   footnote      ,
134   mpfootnote    ,
135   parentequation
136 }
137 \seq_map_inline:Nn \g__talk_cnt_reset_seq
138 {
139   \int_new:c { g__talk_saved_ #1 _int }
140   \int_gset_eq:cc { g__talk_saved_ #1 _int } { c@ #1 }
141 }

```

(End of definition for `\g__talk_cnt_reset_seq`.)

`__talk_cnt_save:` A simple save-and-restore pair.

```

\__talk_cnt_restore: 142 \cs_new_protected:Npn \__talk_cnt_save:
143 {
144   \seq_map_inline:Nn \g__talk_cnt_reset_seq

```

```

145     { \int_gset_eq:cc { g__talk_saved_ ##1 _int } { c@ ##1 } }
146   }
147   \cs_new_protected:Npn \__talk_cnt_restore:
148   {
149     \seq_map_inline:Nn \g__talk_cnt_reset_seq
150     { \int_gset_eq:cc { c@ ##1 } { g__talk_saved_ ##1 _int } }
151   }

```

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

```

\@definecounter Track all counters for resetting.
\std@definecounter
152 \cs_new_eq:NN \std@definecounter \@definecounter
153 \cs_gset_protected:Npn \@definecounter #1
154   {
155     \std@definecounter {#1}
156     \int_new:c { g__talk_saved_ #1 _int }
157     \seq_gput_right:Nn \g__talk_cnt_reset_seq {#1}
158   }

```

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

1.3 Frame options

\l__talk_frame_alignment_tl

```

159 \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_name_str
\l__talk_frame_tagging_str

```

160 \keys_define:nn { talk / frame }
161   {
162     action-spec .str_set:N =
163       \l__talk_action_spec_str ,
164     label .meta:n =
165       { name = {#1} } ,
166     name .str_set:N =
167       \l__talk_frame_name_str ,
168     tag-slides .str_set:N =
169       \l__talk_frame_tagging_str ,
170     vertical-alignment .choices:nn =
171       { bottom , center , stretch , top }
172       {
173         \tl_set_eq:NN \l__talk_frame_alignment_tl
174         \l_keys_value_tl
175       }
176   }
177 \keys_set:nn { talk / frame }
178   {
179     action-spec      =      ,
180     name             =      ,
181     tag-slides       = n    ,
182     vertical-alignment = center
183   }

```

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

1.4 Tagging for headers

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

```

\__talk_header_tag_begin:n
\__talk_header_tag_begin:e
  \__talk_header_tag_end:
184 \cs_new_protected:Npn \__talk_header_tag_begin:n #1
185   {
186     \tag_resume:n { header }
187     \tag_mc_end:
188     \tag_struct_begin:n {#1}
189     \tag_mc_begin:n { }
190   }
191 \cs_generate_variant:Nn \__talk_header_tag_begin:n { e }
192 \cs_new_protected:Npn \__talk_header_tag_end:
193   {
194     \tag_mc_end:
195     \tag_struct_end:
196     \tag_mc_begin:n { artifact }
197     \tag_suspend:n { header }
198   }

```

(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
199 \NewTemplateType { footer-element } { 1 }
200 \DeclareTemplateInterface { footer-element } { talk } { 1 }
201   {
202     color      : tokenlist ,
203     font       : tokenlist = ,
204     left-hspace : length = 0em ,
205     right-hspace : length = 0em
206   }
207 \DeclareTemplateCode { footer-element } { talk } { 1 }
208   {
209     color      = \l__talk_footelem_color_tl ,
210     font       = \l__talk_footelem_font_tl ,
211     left-hspace = \l__talk_footelem_left_skip ,
212     right-hspace = \l__talk_footelem_right_skip
213   }
214   {
215     \tl_if_empty:nF {#1}
216     {
217       \hspace { \l__talk_footelem_left_skip }
218       \hbox:n
219       {
220         \tl_if_empty:NF \l__talk_footelem_color_tl
221         { \color_select:V \l__talk_footelem_color_tl }
222         \l__talk_footelem_font_tl
223         #1
224       }
225       \hspace { \l__talk_footelem_right_skip }
226     }
227   }

```

```

228 \DeclareInstance { footer-element } { date } { talk } { }
229 \DeclareInstance { footer-element } { author } { talk } { }
230 \DeclareInstance { footer-element } { title } { talk } { }
231 \DeclareInstance { footer-element } { subtitle } { talk } { }
232 \DeclareInstance { footer-element } { institute } { talk } { }
233 \DeclareInstance { footer-element } { framenumbers } { talk } { }
234 \DeclareInstance { footer-element } { totalframes } { talk } { }

```

(End of definition for \l__talk_footelem_left_skip and others.)

\l__talk_header_bg_tl Templates for the header area. The background always covers the full width, but the text 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.

```

\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
235 \NewTemplateType { header } { 0 }
236 \DeclareTemplateInterface { header } { talk } { 0 }
237 {
238   background-color : tokenlist,
239   color             : tokenlist = structure ,
240   font              : tokenlist = \normalfont ,
241   height            : length = \Gm@tmargin + \headsep ,
242   left-hspace       : skip = \Gm@lmargin ,
243   print-frame-title : boolean = true ,
244   right-hspace      : skip = \Gm@rmargin
245 }
246 \DeclareTemplateCode { header } { talk } { 0 }
247 {
248   background-color = \l__talk_header_bg_tl ,
249   color            = \l__talk_header_fg_tl ,
250   font             = \l__talk_header_font_tl ,
251   height           = \l__talk_header_ht_dim ,
252   left-hspace      = \l__talk_header_left_skip ,
253   print-frame-title = \l__talk_header_frametitle_bool ,
254   right-hspace     = \l__talk_header_right_skip
255 }
256 {
257   \noindent
258   \__talk_wallpaper_hruler:Nnn
259     \l__talk_header_bg_tl
260     { \l__talk_header_ht_dim - \headsep }
261     { \headsep }
262   \skip_horizontal:n { \l__talk_header_left_skip }
263   \group_begin:
264     \tl_if_empty:NF \l__talk_header_fg_tl
265       { \color_select:V \l__talk_header_fg_tl }
266     \l__talk_header_font_tl
267     \bool_if:NT \l__talk_header_frametitle_bool
268       {
269         \ExpandArgs { nnV }
270         \UseInstance { frametitle } { header }
271         \g__talk_frame_title_tl
272       }
273   \group_end:
274 }

```

```

275 \DeclareInstance { header } { std } { talk } { }
276 \AddToHook { begindocument }
277 {
278   \DeclareInstanceCopy { header } { wallpaper } { std }
279   \EditInstance { header } { wallpaper }
280     { print-frame-title = false }
281 }

```

(End of definition for \l__talk_header_bg_tl and others.)

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

```

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

```

```

323         { @ \_talk\_metadata\_name:n { \_talk\_tmp\_tl } }
324     \clist\_map\_inline:Nn \_talk\_footer\_order\_clist
325     {
326         \tl\_if\_empty:cF { @ \_talk\_metadata\_name:n { ##1 } }
327         {
328             \\_talk\_footer\_sep\_tl
329             \ExpandArgs { nnn }
330             \UseInstance { footer-element } {##1}
331             { @ \_talk\_metadata\_name:n { ##1 } }
332         }
333     }
334 }
335 \hfil
336 }
337 \box\_use\_drop:N \_talk\_tmp\_box
338 \skip\_horizontal:n { \_talk\_footer\_right\_skip - \Gm@rmargin }
339 }
340 \DeclareInstance { footer } { std } { talk } { }
341 \AddToHook { begindocument }
342 {
343     \DeclareInstanceCopy { footer } { wallpaper } { std }
344     \EditInstance { footer } { wallpaper }
345     { element-order = }
346 }

```

(End of definition for _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.

```

347 \cs\_new:Npn \_talk\_metadata\_name:n #1
348 {
349     \tl\_if\_exist:cTF { @ short #1 }
350     { short #1 }
351     {##1}
352 }

```

(End of definition for _talk_metadata_name:n.)

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

```

353 \cs\_new\_protected:Npn \_talk\_wallpaper\_hrule:Nnn #1#2#3
354 {
355     \skip\_horizontal:n { -\Gm@lmargin }
356     \tl\_if\_empty:NF #1
357     {
358         \group\_begin:
359         \color\_select:V #1
360         \rule:nnn { \paperwidth } {##2} {##3}
361         \skip\_horizontal:n { -\paperwidth }
362         \group\_end:
363     }
364 }

```

(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
`\ps@wallpaper` used for frames without “wallpaper” which still need core links, *etc.* We also provide a
`\ps@talk` version that only shows the visual elements: this is deliberately using the same settings
as the main templates.

```

365 \cs_set_nopar:Npn \ps@plain
366 {
367   \cs_set_nopar:Npn \@oddhead
368   {
369     \hfil
370   }
371   \cs_set_nopar:Npn \@oddfoot { }
372   \cs_set_eq:NN \@evenhead \@oddhead
373   \cs_set_eq:NN \@evenfoot \@oddfoot
374 }
375 \cs_set_nopar:Npn \ps@wallpaper
376 {
377   \cs_set_nopar:Npn \@oddhead
378   {
379     \UseInstance { header } { wallpaper }
380     \hfil
381   }
382   \cs_set_nopar:Npn \@oddfoot
383   {
384     \UseInstance { footer } { wallpaper }
385     \hfil
386   }
387   \cs_set_eq:NN \@evenhead \@oddhead
388   \cs_set_eq:NN \@evenfoot \@oddfoot
389 }
390 \cs_new_nopar:Npn \ps@talk
391 {
392   \cs_set_nopar:Npn \@oddhead
393   {
394     \UseInstance { header } { std }
395     \hfil
396   }
397   \cs_set_nopar:Npn \@oddfoot { \UseInstance { footer } { std } }
398   \cs_set_eq:NN \@evenhead \@oddhead
399   \cs_set_eq:NN \@evenfoot \@oddfoot
400 }
401 \pagestyle { talk }

```

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

1.6 The frame environment

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

```

402 \bool_new:N \l__talk_frame_bool

```

(End of definition for `\l__talk_frame_bool`.)

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

```

403 \bool_new:N \g__talk_frame_tag_bool
(End of definition for \g__talk_frame_tag_bool.)

\l__talk_frame_verb_bool Indicates that material was collected verbatim (and thus needs rescanning).
404 \bool_new:N \l__talk_frame_verb_bool
(End of definition for \l__talk_frame_verb_bool.)

\g__talk_frame_int The overall frame number, including LATEX counter-like access.
\c@frame 405 \int_new:N \g__talk_frame_int
\theframe 406 \cs_new_eq:NN \c@frame \g__talk_frame_int
\@framenumber 407 \cs_new:Npn \theframe { \@arabic \c@frame }
408 \cs_new:Npn \@framenumber { \arabic { frame } }

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

\@totalframes The total frames can be handled using the kernel properties.
409 \property_new:nnnn { totalframes } { shipout } { -1 }
410 { \int_use:N \g__talk_frame_int }
411 \AddToHook { enddocument / afterlastpage }
412 { \property_record:nn { lastpage } { totalframes } }
413 \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.
414 \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.
415 \cs_new_protected:Npn \__talk_frame_process:nn #1#2
416 {
417 \int_gincr:N \g__talk_frame_int
418 \bool_set_true:N \l__talk_frame_bool
419 \str_if_empty:NF \l__talk_frame_name_str
420 {
421 \tl_if_exist:cT { g__talk_frame_ \l__talk_frame_name_str _tl }
422 {
423 \msg_error:nnV { talk } { duplicate-frame-name }
424 \l__talk_frame_name_str
425 }
426 \tl_clear_new:c { g__talk_frame_ \l__talk_frame_name_str _tl }
427 \tl_gset:ce { g__talk_frame_ \l__talk_frame_name_str _tl }
428 {
429 {
430 \bool_if:NTF \l__talk_frame_verb_bool
431 { true }
432 { false }
433 }
434 { \exp_not:n {#2} }
435 }
436 }
437 \__talk_slide:nn {#1} {#2}
438 }

```

(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.

```

439 \cs_new_protected:Npn \_talk\_frame\_tag:n #1
440 {
441   \tag_struct_begin:n { tag = frame }
442   \int_gset:Nn \g\_talk\_frame\_struct\_int { \tag_get:n { struct\_num } }
443   \bool_gset_true:N \g\_talk\_frame\_tag\_bool
444   #1
445   \tag_struct_end:
446 }

```

(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.

```

447 \cs_new_protected:Npn \_talk\_frame\_notag:n #1
448 {
449   \tag_mc_begin:n { artifact }
450   \tag_suspend:n { frame }
451   \bool_gset_false:N \g\_talk\_frame\_tag\_bool
452   #1
453   \par
454   \tag_resume:n { frame }
455   \tag_mc_end:
456 }

```

(End of definition for _talk_frame_notag:n.)

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

```

457 \bool_if:NTF \l\_talk\_frame\_title\_bool
458 {
459   \RenewDocumentEnvironment { frame }
460     { D <> { all } = { action-spec } 0 { } +m +b }
461     {
462       \str_clear:N \l\_talk\_frame\_name\_str
463       \keys_set:nn { talk / frame } {#2}
464       \bool_set_false:N \l\_talk\_frame\_verb\_bool
465       \_talk\_frame\_process:nn {#1} { \frametitle {#3} #4 }
466     }
467     { }
468   \NewDocumentEnvironment { frame* }
469     { D <> { all } = { action-spec } 0 { } +m c }
470     {
471       \str_clear:N \l\_talk\_frame\_name\_str
472       \keys_set:nn { talk / frame } {#2}
473       \bool_set_true:N \l\_talk\_frame\_verb\_bool
474       \tl_gset:Nn \g\_talk\_frame\_title\_tl {#3}
475       \exp_args:Nne \_talk\_frame\_process:nn {#1}
476         { \tl_to_str:n { \frametitle } \exp_not:n { {#3} #4 } }
477     }
478     { }

```

```

479 }
480 {
481   \RenewDocumentEnvironment { frame }
482   { !D <> { all } = { action-spec } !0 { } +b }
483   {
484     \str_clear:N \l__talk_frame_name_str
485     \keys_set:nn { talk / frame } {#2}
486     \bool_set_false:N \l__talk_frame_verb_bool
487     \__talk_frame_process:nn {#1} {#3}
488   }
489   { }
490   \NewDocumentEnvironment { frame* }
491   { !D <> { all } = { action-spec } !0 { } c }
492   {
493     \str_clear:N \l__talk_frame_name_str
494     \keys_set:nn { talk / frame } {#2}
495     \bool_set_true:N \l__talk_frame_verb_bool
496     \__talk_frame_process:nn {#1} {#3}
497   }
498   { }
499 }

```

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

```

\__talk_frame_reuse:nn Reusing a frame is largely a question of passing the correct stored information along after
\__talk_frame_reuse_aux:nn setting the flag for re-scanning.
\__talk_frame_reuse_aux:nnn
\reuseframe
\againframe
500 \cs_new_protected:Npn \__talk_frame_reuse:nn #1#2
501 {
502   \tl_if_exist:cTF { g__talk_frame_ #2 _tl }
503   {
504     \exp_args:Nv \__talk_frame_reuse_aux:nn
505     { g__talk_frame_ #2 _tl } {#1}
506   }
507   { \msg_error:nnn { talk } { unknown-frame-name } {#2} }
508 }
509 \cs_new_protected:Npn \__talk_frame_reuse_aux:nn #1#2
510 { \__talk_frame_reuse_aux:nnn #1 {#2} }
511 \cs_new_protected:Npn \__talk_frame_reuse_aux:nnn #1#2#3
512 {
513   \use:c { bool_set_ #1 :N } \l__talk_frame_verb_bool
514   \__talk_frame_process:nn {#3} {#2}
515 }
516 \NewDocumentCommand \reuseframe { D <> { all } = { action-spec } 0 { } m }
517 {
518   \group_begin:
519   \keys_set:nn { talk / frame } {#2}
520   \str_clear:N \l__talk_frame_name_str
521   \__talk_frame_reuse:nn {#1} {#3}
522   \group_end:
523 }
524 \NewCommandCopy \againframe \reuseframe

```

(End of definition for *__talk_frame_reuse:nn* and others. These functions are documented on page ??.)

```

525 \msg_new:nnnn { talk } { unknown-frame-name }
526   { Unknown~frame~name~"#1"! }
527   {
528     You~asked~to~reuse~the~contents~of~a~frame~with~the~name~"#1",~
529     but~that~name~was~never~defined.
530   }
531 \msg_new:nnnn { talk } { duplicate-frame-name }
532   { Duplicate~frame~name~"#1"! }
533   {
534     You~asked~to~save~the~contents~of~this~frame~with~the~name~"#1",~
535     but~that~name~has~already~been~used.
536   }
537 </class>

```

Part V

ltx-talk-frame – The structure of frames

1 ltx-talk-frame-structure implementation

Start the DocStrip guards.

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

1.1 Columns

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

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

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

(End of definition for \l__talk_columns_wd_tl.)

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

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

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

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

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

```

28   \par
29   \__talk_action_end:
30 }

```

\l__talk_column_alignment_tl

```

31 \keys_define:nn { talk / column }
32 {
33   b .meta:n =
34     { vertical-alignment = bottom } ,
35   b .value_forbidden:n = true ,
36   c .meta:n =
37     { vertical-alignment = center } ,
38   c .value_forbidden:n = true ,
39   t .meta:n =
40     { vertical-alignment = top } ,
41   t .value_forbidden:n = true ,
42   vertical-alignment .choices:nn =
43     { bottom , center , top }
44     {
45       \tl_set_eq:NN \l__talk_column_alignment_tl
46       \l_keys_value_tl
47     }
48 }
49 \tl_new:N \l__talk_column_alignment_tl
50 \keys_set:nn { talk / column }
51 {
52   vertical-alignment = center
53 }

```

(End of definition for \l__talk_column_alignment_tl.)

__talk_column_align_bottom:n
 __talk_column_align_center:n
 __talk_column_align_top:n

Most of this will appear in the kernel in due course.

```

54 \cs_new_protected:Npn \__talk_column_align_bottom:n #1
55 { \vbox:n {#1} }
56 \cs_new_protected:Npn \__talk_column_align_center:n #1
57 {
58   \check@mathfonts
59   \vbox_set:Nn \l__talk_tmp_box {#1}
60   \box_set_ht:Nn \l__talk_tmp_box
61   {
62     0.5 \box_ht:N \l__talk_tmp_box
63     + 0.5 \box_dp:N \l__talk_tmp_box
64     + ( \l__talk_vcenter_offset_tl )
65   }
66   \box_set_dp:Nn \l__talk_tmp_box
67   {
68     \box_ht:N \l__talk_tmp_box
69     - ( \l__talk_vcenter_offset_tl ) * 2
70   }
71   \box_use_drop:N \l__talk_tmp_box
72 }
73 \cs_new_protected:Npn \__talk_column_align_top:n #1
74 { \vbox_top:n {#1} }

```

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

`\l__talk_vcenter_offset_tl` Vertical offset based on text mode values: done as a variable `tl` so that a reset to math mode parameters is possible.

```

75 \tl_new:N \l__talk_vcenter_offset_tl
76 \tl_set:Nn \l__talk_vcenter_offset_tl
77   { ( \fontcharht \font `\< - \fontchardp \font `\< ) / 2 }

```

(End of definition for `\l__talk_vcenter_offset_tl`.)

`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.

```

78 \NewDocumentEnvironment { column } { D <> { all } 0 { } m }
79   {
80     \par
81     \int_gincr:N \g__talk_column_int
82     \int_compare:nNnF \g__talk_column_int = 1
83       { \hfil }
84     \keys_set:nn { talk / column } {#2}
85     \vbox_set_to_wd:Nnw \l__talk_tmp_box {#3}
86     \dim_set:Nn \textwidth {#3}
87     \dim_set_eq:NN \columnwidth \textwidth
88     \@parboxrestore
89     \leavevmode
90     \raggedright
91     \__talk_action_begin:n {#1}
92     \ignorespaces
93   }

```

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.

```

94   {
95     \par
96     \__talk_action_end:
97     \vbox_set_end:
98     \use:c { __talk_column_align_ \l__talk_column_alignment_tl :n }
99     { \vbox_unpack_drop:N \l__talk_tmp_box }
100    \par
101    \@ignoretrue
102  }

```

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.

```

103 \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.

```

104 \NewTemplateType { floatenv } { 2 }
105 \DeclareTemplateInterface { floatenv } { talk } { 2 }
106 {
107   float-placement : tokenlist ,
108   horizontal-alignment : choice { left , center , right } = left
109 }
110 \DeclareTemplateCode { floatenv } { talk } { 2 }
111 {
112   float-placement = \l__talk_tmp_tl ,
113   horizontal-alignment =
114   {
115     left = \tl_set:Nn \l__talk_float_alignment_tl { flushleft } ,
116     center = \tl_set:Nn \l__talk_float_alignment_tl { center } ,
117     right = \tl_set:Nn \l__talk_float_alignment_tl { flushright }
118   }
119 }

```

The use of `\@skiphyperreftrue` here is needed as we do not want targets creating for “floats”. We may need a better interface for this: the switch is essentially internal. The structure role shuffle is needed so that the entire unit is treated as a `Float` for tagging, but we do not lose other information.

```

120 {
121   \SetTemplateKeys { floatenv } { talk } {#1}
122   \begin { minipage } { \columnwidth }
123     \use:e
124     {
125       \AssignStructureRole { para / semantic } { float }
126       \begin { \l__talk_float_alignment_tl }
127       \AssignStructureRole{ para / semantic }
128       { \UseStructureName { para / semantic } }
129     }
130     \cs_set_nopar:Npn \@capttype {#2}
131     \@skiphyperreftrue
132   }
133 \DeclareInstance { floatenv } { std } { talk } { horizontal-alignment = left }

```

`\endfloatenv` And the common end function.

```

134 \cs_new_protected:Npn \endfloatenv
135 {
136   \exp_args:Ne \end { \l__talk_float_alignment_tl }
137   \end { minipage }
138 }

```

(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.) 139 \clist_map_inline:nn { figure , table }
140 {
141   \NewDocumentEnvironment {#1} { D <> { all } = { float-placement } 0 { } }

```

```

142     {
143       \__talk_action_begin:n {##1}
144       \UseInstance { floatenv } { std } {##2} {#1}
145     }
146     {
147       \endfloatenv
148       \__talk_action_end:
149     }

```

`\c@figure` The standard variables needed to make captions work (nothing for list of floats, as at present those are not offered). For the “number”, we currently only show the name: “floats” in presentations really should not be numbered.

```

\thetable 150 \newcounter {#1}
\figurename 151 \tl_new:c { #1 name }
\tableename 152 \tl_set:ce { #1 name } { \text_titlecase_first:n {#1} }
\fnum@figure 153 \tl_new:c { fnum@ #1 }
\fnum@table 154 \tl_set:eq:cc { fnum@ #1 } { #1 name }
155 }

```

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

The spacing values needed for the standard function.

```

156 \newlength \abovecaptionskip
157 \newlength \belowcaptionskip
158 \setlength \abovecaptionskip { 7pt }
159 \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.

```

160 \cs_set_protected:Npn \@caption #1 [ #2 ] #3
161 {
162   \par
163   \begingroup
164     \@parboxrestore
165     \if@minipage \@setminipage \fi
166     \normalsize
167     \@makecaption { \csname fnum@ #1 \endcsname } { \ignorespaces #3 }
168   \par
169   \endgroup
170 }

```

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

1.3 Footnotes

`\g__talk_footnote_box` Holds footnotes as they are constructed.

```

171 \box_new:N \g__talk_footnote_box

```

(End of definition for \g__talk_footnote_box.)

`\g__talk_footnote_overlay_seq` For tracking the overlays to apply.

```

172 \seq_new:N \g__talk_footnote_overlay_seq

```

(End of definition for \g__talk_footnote_overlay_seq.)

\stdfootnote

```
173 \NewCommandCopy \stdfootnote \footnote
```

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

\footnote Sort-of overlay aware!

```
174 \RenewDocumentCommand \footnote { D <> { all } o +m }
175 {
176   \seq_gpush:Nn \g__talk_footnote_overlay_seq {#1}
177   \IfNoValueTF {#2}
178     { \stdfootnote {#3} }
179     { \stdfootnote [ {#2} ] {#3} }
180 }
```

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

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

```
181 \NewSocketPlug { fntext / process } { talk }
182 {
183   \vbox_gset:Nn \g__talk_footnote_box
184   {
185     \vbox_unpack:N \g__talk_footnote_box
186     \seq_gpop_left:NN \g__talk_footnote_overlay_seq
187     \l__talk_tmp_tl
188     \exp_args:NV \__talk_decode_parse:n \l__talk_tmp_tl
189     \__talk_action_uncover:N \l__talk_decode_overlays_bool
190     #1
191     \__talk_action_uncover_end:N \l__talk_decode_overlays_bool
192   }
193 }
194 \AssignSocketPlug { fntext / process } { talk }
```

\@makefntext Use a copy of the standard setup.

```
195 \cs_new_eq:NN \@makefntext \fnote_makefntext:n
```

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

```
196 \</class>
```

Part VI

ltx-talk-mode – Modes

1 ltx-talk-mode implementation

Start the DocStrip guards.

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

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

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

(End of definition for __talk_mode:nT.)

`\mode`

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

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

```
13 </class>
```

Part VII

ltx-talk-overlay — Overlays

1 ltx-talk-overlay implementation

Start the DocStrip guards.

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

1.1 Utilities

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

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

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

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

\l__talk_shuffle_skip For tracking.

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

(End of definition for \l__talk_shuffle_skip.)

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

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

```

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

```

(End of definition for __talk_shuffle_skip:n.)

1.2 Opacity utilities

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

```

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

```

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

1.3 Action commands and environments

Commands that can be used as actions all have a common form (with one exception). The common internal structure is used to enable them to be used as actions by looking for the name __talk_action_⟨name⟩:N.

```

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

```

(End of definition for __talk_action_alert:N.)

```

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

```

```

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

```

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

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

```

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

```

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

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

```

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

```

(End of definition for \l__talk_uncover_hidden_fp.)

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

```

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

```

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

\invisible All generated automatically using the above implementations.

```

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

```

```

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

```

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

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

```

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

```

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

```

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

```

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

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

```

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

```

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

```

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

```

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

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

```

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

```

```

\l__talk_saved_overlays_bool
\l__talk_saved_action_str 121 \bool_new:N \l__talk_saved_overlays_bool
\l__talk_saved_actions_bool 122 \str_new:N \l__talk_saved_action_str
123 \bool_new:N \l__talk_saved_actions_bool

```

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

\l__talk_overlay_all_bool

124 \bool_new:N \l__talk_overlay_all_bool

(End of definition for \l__talk_overlay_all_bool.)

actionenv

As we need data on not just overlays but also actions at the end of the environment, this has to be done manually. To allow working with environments but also items, the code needs to save data for the end function. The group is needed for cases where we are not in a L^AT_EX environment group. When an \onslide/\pause is active, it takes priority: sorted by applying up-front. Actions can be skipped entirely if the overlay spec is simply all, as there will never be any spacing issues, *etc.*

__talk_action_begin:n

__talk_action_begin:w

__talk_action_begin_auxi:n

__talk_action_begin_auxii:n

__talk_action_end:

125 \NewDocumentCommand \action { d <> +m }

126 {

127 \group_begin:

128 __talk_action_begin:n {#1}

129 #2

130 __talk_action_end:

131 \group_end:

132 }

133 \NewDocumentEnvironment { actionenv } { d <> }

134 { __talk_action_begin:n {#1} }

135 { __talk_action_end: }

136 \cs_new_protected:Npn __talk_action_begin:n #1

137 {

138 \group_begin:

139 \tl_if_novalue:nTF {#1}

140 {

141 \exp_after:wN __talk_action_begin:w

142 \l__talk_action_spec_str < all > \q_stop

143 }

144 { __talk_action_begin_auxi:n {#1} }

145 }

146 \cs_new_protected:Npn __talk_action_begin:w #1 < #2 > #3 \q_stop

147 { __talk_action_begin_auxi:n {#2} }

148 \cs_new_protected:Npn __talk_action_begin_auxi:n #1

149 {

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

151 { \bool_set_true:N \l__talk_overlay_all_bool }

152 {

153 \bool_set_false:N \l__talk_overlay_all_bool

154 __talk_action_begin_auxii:n {#1}

155 }

156 }

157 \cs_new_protected:Npn __talk_action_begin_auxii:n #1

158 {

159 __talk_decode_parse:n {#1}

160 \bool_set_eq:NN \l__talk_saved_overlays_bool

161 \l__talk_decode_overlays_bool

162 \str_set_eq:NN \l__talk_saved_action_str

163 \l__talk_decode_action_str

164 \bool_set_eq:NN \l__talk_saved_actions_bool

165 \l__talk_decode_actions_bool

166 \tl_if_empty:NTF \g__talk_onslide_tl

167 {

```

168         \bool_if:NTF \l__talk_decode_overlays_bool
169         {
170             \cs_if_exist_use:cF
171             { __talk_action_ \l__talk_decode_action_str :N }
172             { \use_none:n }
173             \l__talk_decode_actions_bool
174         }
175         { \UseInstance { hidden } { std } }
176     }
177     { \__talk_action_invisible:N \c_true_bool }
178 }
179 \cs_new_protected:Npn \__talk_action_end:
180 {
181     \bool_if:NF \l__talk_overlay_all_bool
182     {
183         \tl_if_empty:NTF \g__talk_onslide_tl
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         { \__talk_action_invisible_end:N \c_true_bool }
195     }
196     \group_end:
197 }

```

(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.

```

198 \NewDocumentCommand \alt { D <> { all } +m +m }
199 {
200     \__talk_if_overlay:NTF {#1}
201     {#2}
202     {#3}
203 }

```

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

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

```

204 \NewDocumentCommand \onslide { D <> { all } }
205 {
206     \__talk_onslide:n {#1}
207     \ignorespaces

```

```

208 }
209 \cs_new_protected:Npn \__talk_onslide:n #1
210 {
211   \tl_use:N \g__talk_onslide_tl
212   \tl_gclear:N \g__talk_onslide_tl
213   \__talk_if_overlay:nF {#1}
214   {
215     \__talk_opacity_begin:n { 0 }
216     \tl_gput_right:Nn \g__talk_onslide_tl
217       { \__talk_opacity_end: }
218   }
219 }

```

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

\g__talk_onslide_tl

```

220 \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.

```

221 \NewDocumentCommand \temporal { D <> { all } +m +m +m }
222 {
223   \__talk_if_overlay:nTF {#1}
224     {#3}
225     {
226       \bool_if:NTF \g__talk_slide_continue_bool
227         {#4}
228         {#2}
229     }
230 }

```

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

\pause A thin wrapper.

```

231 \NewDocumentCommand \pause { o }
232 {
233   \legacy_if:nF { measuring@ }
234   {
235     \IfNoValueTF {#1}
236       { \int_gincr:N \g__talk_pauses_int }
237       { \int_gset:Nn \g__talk_pauses_int {#1} }
238     \exp_args:Ne \__talk_onslide:n
239       { \int_eval:n { \g__talk_pauses_int + 1 } - }
240   }
241 }

```

(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`.

```

242 \cs_new_protected:Npn \__talk_overprint_begin:n #1
243 {
244   \par
245   \vbox_set_to_wd:Nnw \l__talk_tmp_box {#1}
246   \raggedright
247   \ignorespaces
248 }
```

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

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

```

249 \NewDocumentEnvironment { overlayarea } { m m }
250 { \__talk_overprint_begin:n {#1} }
251 {
252   \vbox_set_end:
253   \vbox_to_ht:nn {#2}
254   {
255     \box_use_drop:N \l__talk_tmp_box
256     \vfil
257   }
258   \par
259 }
```

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

```

260 \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.

```

261 \cs_new:Npn \__talk_frame_overprint:
262 {
263   \int_to_Roman:n \g__talk_frame_int
264   \int_to_roman:n \l__talk_overprint_int
265 }
```

(End of definition for `__talk_frame_overprint:.`)

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

```

266 \NewDocumentEnvironment { overprint } { 0 { \textwidth } }
267 { \__talk_overprint_begin:n {#1} }
268 {
269   \vbox_set_end:
270   \int_incr:N \l__talk_overprint_int
271   \__talk_overprint_save_ht:
272   \cs_if_exist:cTF
273     { overprint@ \__talk_frame_overprint: }
274     {
```

```

275 \dim_compare:vNnTF
276 { overprint@ \_talk_frame_overprint: }
277 > { \box_ht:N \l\_talk_tmp_box }
278 {
279 \vbox_to_ht:vn
280 { overprint@ \_talk_frame_overprint: }
281 {
282 \box_use_drop:N \l\_talk_tmp_box
283 \vfil
284 }
285 }
286 { \box_use_drop:N \l\_talk_tmp_box }
287 }
288 { \box_use_drop:N \l\_talk_tmp_box }
289 \par
290 }

```

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.

```

291 \cs_new_protected:Npn \_talk_overprint_save_ht:
292 {
293 \tl_if_exist:cF { g\_talk_overprint_ \_talk_frame_overprint: _tl }
294 {
295 \tl_new:c { g\_talk_overprint_ \_talk_frame_overprint: _tl }
296 \tl_gset:cn { g\_talk_overprint_ \_talk_frame_overprint: _tl }
297 { 0pt }
298 }
299 \tl_gset:ce { g\_talk_overprint_ \_talk_frame_overprint: _tl }
300 {
301 \dim_max:vn { g\_talk_overprint_ \_talk_frame_overprint: _tl }
302 { \box_ht:N \l\_talk_tmp_box }
303 }
304 \legacy_if:nT { @files }
305 {
306 \iow_now:Ne \@auxout
307 {
308 \gdef \exp_not:c { overprint@ \_talk_frame_overprint: }
309 {
310 \exp_not:v { g\_talk_overprint_ \_talk_frame_overprint: _tl }
311 }
312 }
313 }
314 \hook_gput_code:nne { enddocument / afterlastpage } { talk }
315 { \_talk_overprint_check_ht:n { \_talk_frame_overprint: } }
316 }
317 \cs_new_protected:Npn \_talk_overprint_check_ht:n #1
318 {
319 \bool_lazy_and:nnF
320 { \exp_not:N \cs_if_exist_p:c { overprint@ #1 } }
321 {
322 \dim_compare_p:vNv { overprint@ #1 } = { g\_talk_overprint_ #1 _tl }
323 }
324 {

```

```

325     \msg_warning:nn { talk } { overprint-ht }
326     \cs_gset_protected:Npn \__talk_overprint_check_ht:n ##1 { }
327   }
328 }
329 \msg_new:nnn { talk } { overprint-ht }
330 {
331   Overprint~area~height~has~changed:\\
332   rerun~LaTeX.
333 }

```

(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 334 \tl_map_inline:nn
\textrm      335 {
\textsc      336   \textbf
\textsf      337   \textit
\textsl      338   \textmd
\texttt      339   \textnormal
\textup      340   \textrm
\emph        341   \textsc
\stdtextbf   342   \textsf
\stdtextit   343   \textsl
\stdtextmd   344   \texttt
\stdtextmd   345   \textup
\stdtextnormal 346   \emph
\stdtextrm   347 }
\stdtextsc   348 {
\stdtextsf   349   \ExpandArgs { c } \NewCommandCopy { std \cs_to_str:N #1 } #1
\stdtextsl   350   \ExpandArgs { Nne } \RenewDocumentCommand #1
\stdtexttt   351     { D <> { all } +m }
\stdtextup   352     {
353       \exp_not:N \__talk_if_overlay:nTF {##1}
354       { \exp_not:c { std \cs_to_str:N #1 } }
355       { \exp_not:N \__talk_textcmd_equiv:n }
356       {##2}
357     }
358   }
359 \cs_new_protected:Npn \__talk_textcmd_equiv:n #1
360 {
361   \mode_if_math:TF
362   { { \mbox {#1} } }
363   {
364     \mode_leave_vertical:
365     {#1}
366   }
367 }

```

(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.

```

368 \RequirePackage { graphicx }
369 \NewCommandCopy \stdincludegraphics \includegraphics
370 \RenewDocumentCommand \includegraphics { s D <> { all } o o m }
371 {
372   \__talk_if_overlay:nT {#2}
373   {
374     \use:e
375     {
376       \exp_not:N \stdincludegraphics
377       \IfBooleanT #1 { * }
378       \IfNoValueF {#3} { [ \exp_not:n { {#3} } ] }
379       \IfNoValueF {#4} { [ \exp_not:n { {#4} } ] }
380     }
381     {#5}
382   }
383 }

```

(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
384 \RenewDocumentCommand \label { D <> { 1 } m }
385 {
386   \@bsphack
387   \__talk_if_overlay:nT {#1}
388   { \__talk_label:n {#2} }
389   \@esphack
390 }
391 \cs_new_protected:Npn \__talk_label:n #1
392 {
393   \begingroup
394     \UseHookWithArguments { label } { 1 } {#1}
395     \protected@write \@auxout { }
396     {
397       \string \newlabel {#1}
398       {
399         { \@currentlabel }
400         { \thepage }
401         { \@currentlabelname }
402         { \@currentHref }
403         { \@kernel@reserved@label@data }
404       }
405     }
406   \endgroup
407 }

```

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

`\std@label@in@display` We also need to cover `\label@in@display`: a bit more awkward as this is a document command but not set up as such in amsmath. For the present, cross our fingers on this!

```

408 \cs_new_eq:NN \std@label@in@display \label@in@display
409 \RenewDocumentCommand \label@in@display { D <> { 1 } m }
410 {
411   \__talk_if_overlay:nT {#1}
412   { \std@label@in@display {#2} }
413 }

```

(End of definition for `\std@label@in@display`. This function is documented on page ??.)

1.7 Overlay patching of third-party environments

To allow opacity to apply to the entirety of constructed graphics, we need to apply a transparency group around the whole thing. We need to do that by saving the input in an Xform object, which then allows the group to be applied.

`\g__talk_opacity_group_int` Each use needs an Xform object, which at present we have to track as there are no anonymous ones.

```

414 \int_new:N \g__talk_opacity_group_int

```

(End of definition for `\g__talk_opacity_group_int`.)

`__talk_opacity_group_begin:` A general mechanism to collect up an environment in a box, which we can then use as the source for an Xform object.

`__talk_opacity_group_end:`

```

415 \cs_new_protected:Npn \__talk_opacity_group_begin:
416 { \begin { lrbox } \l__talk_tmp_box }
417 \cs_new_protected:Npn \__talk_opacity_group_end:
418 {
419   \end { lrbox }
420   \mode_leave_vertical:
421   \int_gincr:N \g__talk_opacity_group_int
422   \exp_args:Ne \pdfxform_new:nnn
423   { talk.opacity \int_use:N \g__talk_opacity_group_int }
424   { /Group << /S /Transparency /K ~ false /I ~ false >> }
425   { \usebox \l__talk_tmp_box }
426   \exp_args:Ne \pdfxform_use:n
427   { talk.opacity \int_use:N \g__talk_opacity_group_int }
428 }

```

(End of definition for `__talk_opacity_group_begin:` and `__talk_opacity_group_end:`.)

At present, we only add code onto the main top-level functions. This is only applied in LuaTeX due to restrictions on Xform structures in pdfTeX. Here, we apply the collection code to the commands not the environment forms to deal with “direct” usage.

```

429 \sys_if_engine luatex:T
430 {
431   \AddToHook { cmd / pspicture / before } { \__talk_opacity_group_begin: }
432   \AddToHook { cmd / endpspicture / after } { \__talk_opacity_group_end: }
433 }
434 </class>

```

Part VIII

ltx-talk-required – “Required” definitions

1 ltx-talk-required implementation

Start the DocStrip guards.

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

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

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

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

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

1.1 Standard design settings

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

```

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

```

1.2 List support

```

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

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

```

Part IX

ltx-talk-structure – Structural commands

1 ltx-talk-structure implementation

Start the DocStrip guards.

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

1.1 Frame title

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

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

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

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

```
\__talk_frame_title:n Inserting the frame title requires we deal with tagging as well as appearance: if there is
\__talk_frame_title_tagged:n a title, we need to tag just this part of the header.
```

```
15 \NewTemplateType { frametitle } { 1 }
16 \DeclareTemplateInterface { frametitle } { talk } { 1 }
17 {
18   after-vspace : skip = \bigskipamount ,
19   before-vspace : skip = 0em ,
20   color : tokenlist = ,
21   font : tokenlist = \Large \bfseries
22 }
23 \DeclareTemplateCode { frametitle } { talk } { 1 }
24 {
25   after-vspace = \l__talk_frametitle_after_skip ,
26   before-vspace = \l__talk_frametitle_before_skip ,
27   color = \l__talk_frametitle_color_tl ,
28   font = \l__talk_frametitle_font_tl
29 }
```

```

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

```

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

1.2 Headings

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

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

Here, we set up the same keyval interface as used by the kernel (see `latex-lab-sec-template.dtx`)

```

\l__talk_sec_bookmark_tl
\l__talk_sec_label_tl
\l__talk_sec_nameref_tl
\l__talk_sec_numbered_bool
\l__talk_sec_quote_tl
\l__talk_sec_running_tl
\l__talk_sec_subtitle_tl
\l__talk_sec_toc_tl
72 \keys_define:nn { talk / heading }
73 {
74   bookmark .tl_set:N =
75     \l__talk_sec_bookmark_tl ,
76   label .code:n =
77     { \tl_put_right:N \l__talk_sec_label_tl { \label {#1} } } ,
78   nameref .tl_set:N =
79     \l__talk_sec_nameref_tl ,
80   numbered .bool_set:N
81     = \l__talk_sec_numbered_bool ,
82   numbered .default:n
83     = true ,
84   unnumbered .bool_set_inverse:N
85     = \l__talk_sec_numbered_bool ,
86   unnumbered .default:n
87     = true ,
88   quote .tl_set:N =
89     \l__talk_sec_quote_tl ,
90   running .tl_set:N =
91     \l__talk_sec_running_tl ,
92   shorttitle .meta:n =
93     {
94       bookmark = {#1} ,
95       nameref = {#1} ,
96       running = {#1} ,
97       toc = {#1}
98     } ,
99   subtitle .tl_set:N =
100     \l__talk_sec_subtitle_tl ,
101   toc .tl_set:N =
102     \l__talk_sec_toc_tl
103 }
104 \tl_new:N \l__talk_sec_label_tl

```

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

Here, we need full L^AT_EX counters, so create them using the appropriate mechanism: that also means we can sort out counter dependency and the appearance (using the same setup as in `article`). As (subsub)section numbers never increment inside frames, we remove these counters from the general tracker.

```

\section
\subsection
\subsubsection
\thesection
\thesubsection
\thesubsubsection
105 \newcounter { section }
106 \newcounter { subsection } [ section ]
107 \newcounter { subsubsection } [ subsection ]
108 \seq_gremove_all:Nn \g__talk_cnt_reset_seq { section }
109 \seq_gremove_all:Nn \g__talk_cnt_reset_seq { subsection }
110 \seq_gremove_all:Nn \g__talk_cnt_reset_seq { subsubsection }
111 \cs_gset:Npn \thesection { \@arabic \c@section }
112 \cs_gset:Npn \thesubsection { \thesection . \@arabic \c@subsection }
113 \cs_gset:Npn \thesubsubsection { \thesubsection . \@arabic \c@subsubsection }

```

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

<code>\section</code> <code>\subsection</code> <code>\subsubsection</code> <code>\insertsection</code> <code>\insertsubsection</code> <code>\insertsubsubsection</code> <code>__talk_head_section:Nnn</code> <code>__talk_head_subsection:Nnn</code> <code>__talk_head_subsubsection:Nnn</code>	<p>The sectioning commands all have essentially the same form: we therefore create using a generator with the necessary conditionals in place. As we do not typeset sections at this stage, the code is quite different from <code>article</code>. This also means that the bookmark links need to point forward to the next slide: if that doesn't appear, the bookmarks will be out. Using the general scratch sequence here should be OK: it really is a one-off setting. We need a sequence to allow indexed mapping to avoid any extra setup for the depth value.</p> <pre> 114 \seq_set_from_clist:Nn \l_tmpa_seq 115 { section , subsection , subsubsection } 116 \seq_map_indexed_inline:Nn \l_tmpa_seq 117 { 118 \use:e 119 { 120 \NewDocumentCommand \exp_not:c { insert #2 } { } 121 { 122 \exp_not:N \tl_use:N 123 \exp_not:c { l__talk_ #2 _tl } 124 } 125 \NewDocumentCommand \exp_not:c {#2} 126 { s D <> { all } = { shorttitle } 0 {##4} m } 127 { 128 \exp_not:N \bool_if:NF \exp_not:N \l__talk_frame_bool 129 { 130 __talk_if_overlay:nT {##2} 131 { \exp_not:c { __talk_head_ #1 :Nnn } ##1 {##3} {##4} } 132 } 133 } 134 \cs_new_protected:Npn \exp_not:c { __talk_head_ #1 :Nnn } ##1##2##3 135 { 136 \exp_not:N \refstepcounter {#2} 137 \UseTaggingSocket { sec / end } { \use:c { toplevel@ #2 } } 138 \UseTaggingSocket { sec / begin } 139 { 140 { \use:c { toplevel@ #2 } } 141 { 142 tag = 143 \exp_not:N \UseStructureName 144 { sec / \use:c { toplevel@ #2 } } 145 } 146 } 147 \tl_set:Nn \exp_not:c { l__talk_ #2 _tl } {##3} 148 \keys_set:nn { talk / heading } { shorttitle = {##3} , ##2 } 149 \exp_not:N \bool_if:NT ##1 150 { 151 \tl_clear:N \exp_not:N \l__talk_sec_bookmark_tl 152 \tl_clear:N \exp_not:N \l__talk_sec_running_tl 153 \tl_clear:N \exp_not:N \l__talk_sec_toc_tl 154 \bool_set_false:N \exp_not:N \l__talk_sec_numbered_bool 155 } 156 \UseTaggingSocket { talk / sec / title } {#2} 157 \str_if_eq:nnT {#2} { section } 158 { \tl_clear:N \exp_not:N \l__talk_subsection_tl } 159 \str_if_eq:nnF {#2} { subsubsection } </pre>
--	---

```

160         { \tl_clear:N \exp_not:N \l__talk_subsubsection_tl }
161         \exp_not:N \__talk_head_marks:nnVVV {#2} {#1}
162         \exp_not:N \l__talk_sec_toc_tl
163         \exp_not:N \l__talk_sec_bookmark_tl
164         \exp_not:N \l__talk_sec_nameref_tl
165         \hook_use:n { #2 / begin }
166     }
167     \hook_new:n { #2 / begin }
168 }
169 }

```

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

```

\addcontentslinebookmark Patches as in latex-lab-sec-template.dtx.
\addcontentslinebookmarkOff 170 \providecommand \addcontentslinebookmark [ 3 ] { }
\addcontentslinebookmarkOn 171 \providecommand \addcontentslinebookmarkOff { }
172 \providecommand \addcontentslinebookmarkReset { }
173 \providecommand \texorpdfstring [ 2 ] {#1}

```

(End of definition for `\addcontentslinebookmark`, `\addcontentslinebookmarkOff`, and `\addcontentslinebookmarkOn`. These functions are documented on page ??.)

```

\__talk_head_marks:nnnnn For handling bookmarks, TOC, etc.: the naming here follows the LATEX kernel, as does
\__talk_head_marks:nnVVV the code in general, but with some minor adjustments.
\__talk_head_marks_aux:Nnn
174 \cs_new_protected:Npn \__talk_head_marks:nnnnn #1#2#3#4#5
175 {
176     \tl_set:Nn \@currentlabelname {#5}
177     \tl_if_blank:nTF {#3}
178     {
179         \tl_if_blank:nF {#4}
180         {
181             \__talk_head_marks_aux:Nnnn
182             \addcontentslinebookmark {#1} {#2} {#4}
183         }
184     }
185     {
186         \tl_if_blank:nT {#4}
187         { \addcontentslinebookmarkOff }
188         \__talk_head_marks_aux:Nnnn
189         \addcontentsline {#1} {#2} { \texorpdfstring {#3} {#4} }
190     }
191 }
192 \cs_generate_variant:Nn \__talk_head_marks:nnnnn { nnVVV }
193 \cs_new_protected:Npn \__talk_head_marks_aux:Nnnn #1#2#3#4
194 {
195     #1 { toc } {#2}
196     {
197         \int_compare:nNnF {#3} > { \value { secnumdepth } }
198         { \protect \numberline { \use:c { the #2 } } }
199         #4
200     }
201 }

```

(End of definition for `__talk_head_marks:nnnnn` and `__talk_head_marks_aux:Nnn`.)

```

talk/sec/title The argument is one of section, subsection or subsubsection.
\__talk_head_tag:nn
202 \NewTaggingSocket { talk / sec / title } { 1 }
203 \NewTaggingSocketPlug { talk / sec / title } { default }
204 {
205     \exp_args:Ne \__talk_head_tag:nn
206     { \text_purify:v { l__talk_ #1 _ t1 } } {#1}
207 }
208 \cs_new_protected:Npn \__talk_head_tag:nn #1#2
209 {
210     \tag_struct_begin:e
211     {
212         tag =
213         \UseStructureName { sec / \use:c { toplevel@ #2 } / title } ,
214         title = {#1} ,
215         actualtext = {#1} ,
216     }
217     \tag_struct_end:
218 }
219 \AssignTaggingSocketPlug { talk / sec / title } { default }

(End of definition for talk/sec/title and \__talk_head_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 $\text{\LaTeX} 2_{\epsilon}$ code as much as possible.

```

220 \cs_gset_protected:Npn \@starttoc #1
221 {
222     \begingroup
223     \makeatletter
224     \UseTaggingSocket { toc / starttoc / before } {#1}
225     \@input { \jobname .#1 }
226     \UseTaggingSocket { toc / starttoc / after } {#1}
227     \legacy_if:nT { @filesw }
228     {
229         \AddToHook { enddocument / afterlastpage }
230         {
231             \expandafter \newwrite \csname tf@ #1 \endcsname
232             \immediate \openout \csname tf@ #1 \endcsname \jobname .#1 \relax
233         }
234     }
235     \nobreakfalse
236     \endgroup
237 }

```

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

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

```

238 \NewDocumentCommand \tableofcontents { 0 { } }
239 {
240     \group_begin:

```

```

241     \@starttoc { toc }
242     \group_end:
243 }

```

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

```

\l@section Initial hard-coded versions to be templated once we have some other effects also working.
\l@subsection We may need to look at this “higher up” as we will need to know the section numbers.
\l@subsubsection
__talk_toc_aux:nnnn 244 \cs_new_protected:Npn \l@section #1#2
__talk_toc_dest:n 245 { __talk_toc_aux:nnnn { 1 } { \bfseries \color { structure } } {#1} {#2} }
__talk_toc_dest:w 246 \cs_new_protected:Npn \l@subsection #1#2
__talk_toc_level:nnnn 247 {
248     __talk_toc_aux:nnnn
249     { 2 }
250     {
251         \skip_set:Nn \leftskip { 2em }
252         \color_ensure_current:
253     }
254     {#1} {#2}
255 }
256 \cs_new_protected:Npn \l@subsubsection #1#2
257 {
258     __talk_toc_aux:nnnn
259     { 3 }
260     {
261         \skip_set:Nn \leftskip { 4em }
262         \color_ensure_current:
263         \footnotesize
264     }
265     {#1} {#2}
266 }
267 \cs_new_protected:Npn __talk_toc_aux:nnnn #1#2#3#4
268 {
269     \int_compare:nNnTF { \value { section } } < 1
270     { \use:n }
271     { __talk_toc_dest:n }
272     { __talk_toc_level:nnnn {#1} {#2} {#3} {#4} }
273 }

```

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.

```

274 \cs_new_protected:Npn __talk_toc_dest:n
275 {
276     \exp_after:wN __talk_toc_dest:w \@contentsline@destination
277     . 0 . 0 . 0 . \q_stop
278 }
279 \cs_new_protected:Npn __talk_toc_dest:w #1 . #2 . #3 . #4 . #5 \q_stop #6
280 {
281     \bool_lazy_or:nnTF
282     {
283         \bool_lazy_and_p:nn
284         { \int_compare_p:nNn { \value { section } } = {#2} }
285         { \int_compare_p:nNn { \value { subsection } } = 0 }

```

```

286     }
287     {
288         \bool_lazy_and_p:nn
289         { \int_compare_p:nNn { \value { section } } = {#2} }
290         { \int_compare_p:nNn { \value { subsection } } = {#3} }
291     }
292     {#6}
293     {
294         \opacity_begin:n { 0.2 }
295         #6
296         \opacity_end:
297     }
298 }
299 \cs_new_protected:Npn \__talk_toc_level:nnnn #1#2#3#4
300 {
301     \int_compare:nNnF {#1} > { \value { tocdepth } }
302     {
303         \group_begin:
304         \noindent
305         #2
306         \UseHookWithArguments { contentsline / text / before } { 4 }
307         {#1} {#3} {#4} { \@contentsline@destination }
308         #3
309         \UseHookWithArguments { contentsline / text / after } { 4 }
310         {#1} {#3} {#4} { \@contentsline@destination }
311         \UseHookWithArguments { contentsline / page / before } { 4 }
312         {#1} {#3} {#4}
313         { \@contentsline@destination }
314         \UseHookWithArguments { contentsline / page / after } { 4 }
315         {#1} {#3} {#4}
316         { \@contentsline@destination }
317         \par
318         \group_end:
319         \vfil
320     }
321 }

```

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

```

322 \setcounter { tocdepth } { 2 }

```

1.4 Block environments

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

```

    quote (env.) 323 \NewDocumentEnvironment { description } { } { } { } { }
    quotation (env.) 324 \NewDocumentEnvironment { quote } { } { } { } { }
    verse (env.) 325 \NewDocumentEnvironment { quotation } { } { } { } { }
    stdquote (env.) 326 \NewDocumentEnvironment { verse } { } { } { } { }
stdquotation (env.) 327 \AddToHook { begindocument / before }
    stdverse (env.) 328 {
329     \clist_map_inline:nn { quote , quotation , verse }
330     {
331         \NewEnvironmentCopy { std #1 } {#1}
332         \RenewDocumentEnvironment {#1} { d <> !0 { } }
333         {

```

```

334         \_talk_action_begin:n {##1}
335         \begin { std #1 } [ {##2} ]
336         \ignorespaces
337     }
338     {
339         \end { std #1 }
340         \_talk_action_end:
341     }
342 }
343 }

```

`block (env.)`

```

344 \NewDocumentEnvironment { block } { d <> m }
345 {
346     \_talk_action_begin:n {#1}
347     \par
348     \vbox_set:Nw \l__talk_tmp_box
349     \group_begin:
350     \medskip
351     \leavevmode
352     \normalfont \large \bfseries
353     \color { structure }
354     #2
355     \par
356     \medskip
357     \group_end:
358 }
359 {
360     \vbox_set_end:
361     \box_use:N \l__talk_tmp_box
362     \par
363     \_talk_action_end:
364 }

```

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! Once it is stable, we should revisit here to see if it would be sensible to use the hook mechanism.

```

365 \AddToHook { begindocument / before }
366 {
367     \NewCommandCopy \stditem \item
368     \RenewDocumentCommand \item { d <> o }
369     {
370         \IfNoValueTF {#2}
371         { \stditem }
372         { \stditem [ {#2} ] }
373         \IfNoValueTF {#1}
374         {
375             \exp_after:wN \_talk_item_parse_spec:w
376             \l__talk_action_spec_str < all > \q_stop

```

```

377     }
378     { \_talk_item_parse_spec:n {#1} }
379   }
380 }

```

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.

```

381 \cs_new_protected:Npn \_talk_item_parse_spec:w #1 < #2 > #3 \q_stop
382 { \_talk_item_parse_spec:n {#2} }
383 \cs_new_protected:Npn \_talk_item_parse_spec:n #1
384 {
385   \bool_lazy_or:nnF
386     { \tl_if_blank_p:n {#1} }
387     { \str_if_eq_p:nn {#1} { all } }
388   {
389     \tl_set:Nx \l__talk_list_end_tl
390       {
391         \exp_not:N \int_compare:nNnT \tex_currentgrouplevel:D =
392           { \int_eval:n { \tex_currentgrouplevel:D + 1 } }
393         {
394           \_talk_action_end:
395           \tl_clear:N \exp_not:N \l__talk_list_end_tl
396         }
397       }
398     \_talk_action_begin:n {#1}
399   }
400 }

```

(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`

```

401 \tl_new:N \l__talk_list_end_tl

```

(End of definition for `\l__talk_list_end_tl`.)

`_block_inter_item:` Before L^AT_EX 2026-6-01, there was no hook pair for surrounding items, so so patching `\BlockEnvEnd` had to be done. The extra complexity here is that there is no test for hook existence, and the hook was not in the first dev release of 2026-06-01. So there is some low-level testing needed: `\l__block_topsepadd_skip` is used as a marker.

```

402 \cs_if_exist:NTF \l__block_topsepadd_skip
403 {
404   \cs_gset_protected:Npn \_block_inter_item:
405     {
406       \legacy_if:nT { @inlabel }
407       { \indent \par }
408       \mode_if_horizontal:T
409       {
410         \_block_skip_remove_last:
411         \_block_skip_remove_last:
412         \par
413       }

```

```

414 \l__talk_list_end_tl
415 \__kernel_list_item_end:
416 \__kernel_list_item_begin:
417 \addpenalty \@itempenalty
418 \addvspace \itemsep
419 }

```

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

```

420 \IfFormatAtLeastTF { 2026-06-01 }
421 { \cs_gset_protected:Npe \BlockEnvEnd }
422 { \cs_gset:Npe \endblockenv }
423 {
424   \exp_not:n
425   { \__block_debug_typeout:n { blockenv~common~ending \on@line } }
426   \cs_if_exist:NTF \l__block_transparent_level_bool
427   {
428     \exp_not:N \bool_if:NF
429     \exp_not:N \l__block_transparent_level_bool
430   }
431   {
432     \exp_not:N \bool_if:NT
433     \exp_not:N \l__block_level_incr_bool
434   }
435   { \int_gdecr:N \exp_not:N \g_block_nesting_depth_int }
436   \exp_not:n
437   {
438     \legacy_if:nT { @inlabel }
439     {
440       \mode_leave_vertical:
441       \legacy_if_gset_false:n { @inlabel }
442     }
443     \__block_if_list:T
444     { \legacy_if:nT { @newlist } { \@noitemerr } }
445     \mode_if_horizontal:TF
446     {
447       \__block_skip_remove_last:
448       \__block_skip_remove_last:
449       \par
450     }
451     { \@inmatherr { \end { \@currenvir } } }
452     \l__talk_list_end_tl
453     \__kernel_displayblock_end:
454     \__block_if_list:T { \legacy_if_gset_false:n { @newlist } }
455     \legacy_if_gset_false:n { @nobreak }
456     \legacy_if:nF { @nolist }
457     {
458       \__block_skip_set_to_last:N \l_tmpa_skip
459       \dim_compare:nNnT \l_tmpa_skip > \c_zero_dim
460       {
461         \skip_vertical:n { - \l_tmpa_skip }
462         \skip_vertical:n
463         { \l_tmpa_skip + \parskip - \@outerparskip }
464       }
465       \addpenalty \@endparpenalty
466       \addvspace \l__block_topsepadd_skip

```

```

467         }
468         \socket_use:n { block / endpe }
469     }
470 }
471 }

```

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

The branch where `\l__block_topsepadd_skip` is not defined, which means we have a sufficiently new kernel to use the hooks. When we revise for the updated kernel, we could consider simply adding the code directly to the hook rather than using a token list, although that would mean hook operations each time there is an overlay.

```

472 {
473     \AddToHook { item / after }
474     { \l__talk_list_end_tl }
475 }

```

`itemize (env.)` Allow for the classical beamer syntax: currently two versions but that will only last until

`enumerate (env.)` the 2026-06-01 release of L^AT_EX is out.

`description (env.)`

```

476 \AddToHook { begindocument / before }
477 {
478     \clist_map_inline:nn { itemize , enumerate , description }
479     {
480         \IfFormatAtLeastTF { 2026-06-01 }
481         {
482             \RenewDocumentEnvironment {#1} { = { action-spec } !o }
483             { \SimpleBlockEnv {#1} {##1} }
484             { \BlockEnvEnd }
485         }
486         {
487             \RenewDocumentEnvironment {#1} { = { action-spec } !o }
488             {
489                 \IfNoValueTF {##1}
490                 { \UseInstance { blockenv } {#1} { } }
491                 { \UseInstance { blockenv } {#1} {##1} }
492             }
493             { \endblockenv }
494         }
495     }
496 }

```

And add the structural color to item labels.

```

497 \AddToHook { begindocument / before }
498 {
499     \EditInstance { item } { basic }
500     { label-format = \color { structure } #1 }
501     \EditInstance { item } { description }
502     { label-format = \normalfont \bfseries \color { structure } #1 }
503 }

```

`\l__talk_action_spec_str`

Add an overlay key to the block template. Placed here, it applies *before* the `\item` starts, so we do not have to redefine the latter to do actions up-front. This also means it can apply to whatever we want it to within a block. Currently two versions but that will only last until the 2026-06-01 release of L^AT_EX is out.

```

504 \IfFormatAtLeastTF { 2026-06-01 }
505 {
506   \keys_define:nn { template / block / std }
507   { action-spec .str_set:N = \l__talk_action_spec_str }
508 }
509 {
510   \keys_define:nn { template / block / display }
511   { action-spec .str_set:N = \l__talk_action_spec_str }
512 }

```

(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.

```

513 \NewCommandCopy \stdnewtheorem \newtheorem
514 \RenewDocumentCommand \newtheorem { m o m o }
515 {
516   \IfNoValueTF {#4}
517   {
518     \IfNoValueTF {#2}
519     { \stdnewtheorem {#1} {#3} }
520     { \stdnewtheorem {#1} [ {#2} ] {#3} }
521   }
522   {
523     \IfNoValueTF {#2}
524     { \stdnewtheorem {#1} {#3} [ {#4} ] }
525     { \stdnewtheorem {#1} [ {#2} ] {#3} [ {#4} ] }
526   }
527   \NewEnvironmentCopy { std #1 } {#1}
528   \RenewDocumentEnvironment {#1} { D <> { all } o }
529   {
530     \__talk_action_begin:n {##1}
531     \IfNoValueTF {##2}
532     { \begin { std #1 } }
533     { \begin { std #1 } [ {##2} ] }
534     \ignorespaces
535   }
536   {
537     \end { std #1 }
538     \__talk_action_end:
539   }
540 }

```

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

541 `\</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	
\(77
\)	77
@ commands:	
\@_decode_overlay_+ :nw	144
\\	331
A	
\abovecaptionskip	156, 158
\action	125
actionenv (env.)	125
\addcontentsline	189
\addcontentslinebookmark	170, 182
\addcontentslinebookmarkOff	170, 187
\addcontentslinebookmarkOn	170
\addcontentslinebookmarkReset	172
\addpenalty	417, 465
\AddToHook	54, 60, 71, 229, 276, 327, 341, 365, 411, 431, 432, 473, 476, 497
\addtolength	48
\advspace	418, 466
\againframe	500
\alert	42, 103
alertenv (env.)	110
\alt	198
\and	131
\arabic	408
\arraycolsep	25
\arrayrulewidth	27
\AssignSocketPlug	194
\AssignStructureRole	125, 127
\AssignTaggingSocketPlug	219
\author	15
B	
\begin	122, 126, 130, 144, 335, 416, 532, 533
\begingroup	163, 222, 393
\belowcaptionskip	157, 159
\bfseries	21, 39, 245, 352, 502
\bigskipamount	18
block (env.)	344
block commands:	
\g_block_nesting_depth_int	435
block internal commands:	
_block_debug_typeout:n	425
_block_if_list:TF	443, 454
_block_inter_item:	402, 404
\l_block_level_incr_bool	433
_block_skip_remove_last:	410, 411, 447, 448
_block_skip_set_to_last:N	458
\l_block_topsepadd_skip	62, 64, 402, 466
\l_block_transparent_level_bool	426, 429
\BlockEnvEnd	402, 484
bool commands:	
\bool_do_while:Nn	28
\bool_gset_eq:NN	39, 44
\bool_gset_false:N	31, 451
\bool_gset_true:N	221, 229, 235, 443
\bool_if:NTF	6, 37, 44, 46, 49, 50, 58, 66, 71, 83, 91, 128, 141, 149, 168, 181, 185, 226, 267, 428, 430, 432, 457
\bool_lazy_and:nnTF	21, 47, 232, 319
\bool_lazy_and_p:nn	283, 288
\bool_lazy_any:nnTF	64, 91
\bool_lazy_or:nnTF	5, 18, 21, 281, 385
\bool_lazy_or_p:nn	24
\bool_new:N	3, 3, 4, 7, 8, 13, 121, 123, 124, 402, 403, 404
\bool_set_eq:NN	35, 160, 164
\bool_set_false:N	27, 28, 29, 35, 153, 154, 464, 486
\bool_set_true:N	24, 50, 66, 69, 101, 151, 162, 199, 218, 231, 418, 473, 495
\c_false_bool	15
\c_true_bool	14, 177, 194
box commands:	
\box_dp:N	36, 63
\box_gclear:N	82
\box_ht:N	62, 68, 277, 302
\box_if_empty:N	97
\box_new:N	5, 115, 171
\box_set_dp:Nn	66
\box_set_ht:Nn	60
\box_use:N	361
\box_use_drop:N	26, 71, 255, 282, 286, 288, 337
\box_wd:N	41
box internal commands:	
_box_dim_eval:n	33, 36, 41, 44
_box_set_to_wd:	40, 45
C	
\clearpage	103

clist commands:	
\clist_const:Nn	56
\clist_if_in:NnTF	64, 198
\clist_map_break:	236
\clist_map_inline:Nn	132, 201, 324
\clist_map_inline:nn	3, 89, 139, 140, 329, 478
\clist_new:N	10, 14
\clist_pop:NNTF	320
\clist_set:Nn	104, 197
\color	4, 11, 56, 245, 353, 500, 502
color commands:	
\color_ensure_current:	63, 252, 262
\color_group_begin:	34, 46
\color_group_end:	34
\color_math:nn	9, 26
\color_math:nnn	10, 27
\color_select:n	7, 16, 35, 37, 45, 46, 69, 107, 221, 265, 318, 359
\color_select:nn	8, 17, 38
color internal commands:	
_color_backend_reset:	64
\colorlet	68
column (env.)	78
columns (env.)	11
\columnwidth	20, 87, 122
cs commands:	
\cs_generate_variant:Nn	7, 8, 9, 10, 104, 105, 107, 108, 109, 110, 111, 112, 113, 191, 192
\cs_gset:Npe	422
\cs_gset:Npn	111, 112, 113
\cs_gset_eq:NN	56
\cs_gset_nopar:Npn	49, 57, 64
\cs_gset_protected:Npe	62, 89, 421
\cs_gset_protected:Npn	29, 38, 48, 56, 58, 153, 220, 326, 404
\cs_if_exist:NTF	125, 272, 402, 426
\cs_if_exist_p:N	320
\cs_if_exist_use:NTF	153, 170, 187
\cs_new:Npn	6, 8, 34, 35, 36, 37, 38, 39, 40, 41, 42, 223, 261, 347, 407, 408, 413
\cs_new_eq:NN	5, 7, 152, 195, 406, 408
\cs_new_nopar:Npn	3, 390
\cs_new_protected:Npe	62, 78, 104
\cs_new_protected:Npn	10, 11, 16, 18, 18, 33, 38, 40, 42, 43, 44, 48, 51, 52, 54, 54, 56, 56, 61, 62, 64, 69, 73, 77, 81, 87, 87, 91, 105, 111, 111, 114, 117, 123, 134, 134, 136, 142, 144, 146, 147, 148, 150, 157, 160, 165, 167, 174, 178, 179, 184, 189, 192, 193, 195, 205, 208, 209, 215, 242, 244, 246, 256, 267, 274, 279, 291, 299, 317, 353, 359, 381, 383, 391, 415, 415, 417, 439, 447, 500, 509, 511
\cs_set:Npn	14, 15, 64
\cs_set_eq:NN	61, 62, 63, 64, 217, 372, 373, 387, 388, 398, 399
\cs_set_nopar:Npn	130, 219, 220, 221, 222, 365, 367, 371, 375, 377, 382, 392, 397
\cs_set_protected:Npn	40, 131, 160, 167, 224
\cs_to_str:N	349, 354
\csname	167, 231, 232
D	
\date	15
\day	20
\DeclareColor	65, 71, 72, 73
\DeclareInstance	43, 79, 80, 81, 83, 85, 87, 133, 228, 229, 230, 231, 232, 233, 234, 275, 340
\DeclareInstanceCopy	278, 343
\DeclareTemplateCode	23, 54, 77, 109, 110, 207, 246, 293
\DeclareTemplateInterface	16, 45, 75, 95, 105, 200, 236, 283
\definecolor	69
description (env.)	323, 476
dim commands:	
\dim_compare:nNnTF	108, 275, 459
\dim_compare_p:nNn	109, 322
\dim_const:Nn	164, 170
\dim_eval:n	51, 52, 53
\dim_max:nn	110, 301
\dim_set:Nn	19, 86
\dim_set_eq:NN	20, 87
\dim_to_decimal:n	157
\dim_use:N	185, 186
\c_zero_dim	459
\DocumentMetadata	8
\doublerulesep	28
E	
\EditInstance	279, 344, 499, 501
\emph	334
\end	136, 137, 137, 146, 339, 419, 451, 537
\endblockenv	402, 493
\endcsname	167, 231, 232
\endfloatenv	134, 147
\endgroup	169, 236, 406
enumerate (env.)	476
environments:	
actionenv	125
alertenv	110
block	344

column	78	fnote commands:	
columns	11	\fnote_makefnstext:n	195
description	323, 476	\font	77
enumerate	476	\fontchardp	77
figure	139	\fontcharht	77
invisibleenv	98	\footins	30
itemize	476	\footnote	173, 174
onlyenv	118	\footnoterule	99
overlayarea	249	\footnotesize	263
overprint	266	\footskip	307, 308
quotation	323	fp commands:	
quote	323	\fp_eval:n	162
stdquotation	323	\fp_to_dim:n	172
stdquote	323	\frame	28, 27, 414
stdverse	323	frame	457
table	139	frame*	457
uncoverenv	98	\framesubtitle	10
verse	323	\frametitle	5, 465, 476
visibleenv	98		
exp commands:		G	
\exp_after:wN	141, 276, 375	\gdef	308
\exp_args:Ne		\geometry	5
	17, 52, 136, 205, 238, 422, 426	group commands:	
\exp_args:Nne	475	\group_begin:	
\exp_args:No	30		12, 33, 33, 60, 63, 67, 105,
\exp_args:NV	129, 188	\c_group_begin_token	43
\exp_args:Nv	504	\group_end:	40, 41, 59, 63, 67, 75, 108,
\exp_args_generate:n	106		131, 196, 242, 273, 318, 357, 362, 522
\exp_not:N	64, 66, 67, 68, 69,	\group_insert_after:N	45
	71, 72, 72, 73, 78, 80, 81, 84, 86, 89,		
	94, 96, 100, 101, 106, 107, 111, 112,	H	
	120, 122, 123, 125, 128, 131, 134,	hbox commands:	
	136, 143, 147, 149, 151, 152, 153,	\hbox:n	218
	154, 158, 160, 161, 162, 162, 163,	\hbox_set_end:	25
	164, 167, 168, 170, 172, 173, 176,	\hbox_set_to_wd:Nnw	18
	177, 181, 308, 320, 353, 354, 355,	\headsep	241, 260, 261
	376, 391, 395, 428, 429, 432, 433, 435	\hfil	83, 291, 335, 369, 380, 385, 395
\exp_not:n		hook commands:	
	310, 378, 379, 424, 434, 436, 476	\hook_gput_code:nnn	105, 161, 314
\exp_stop_f:	51, 52, 53	\hook_new:n	167
\expandafter	231	\hook_use:n	165
\ExpandArgs		\hspace	217, 225
	91, 98, 134, 269, 322, 329, 349, 350	\hypersetup	225
		I	
F		\IfBooleanT	377
\fboxrule	32	\ifcase	5
\fboxsep	31	\IfFormatAtLeastF	7
\fi	18, 165	\IfFormatAtLeastTF	420, 480, 504
figure (env.)	139	\IfNoValueF	378, 379
\figurename	150	\IfNoValueTF	15, 25, 36, 47, 67, 177,
file commands:			235, 370, 373, 489, 516, 518, 523, 531
\file_if_exist_input:nTF	157	\ignorespaces	
\file_input:n	159		19, 21, 92, 167, 207, 247, 336, 534

<code>\RenewDocumentCommand</code>	11, 15, 21, 22, 27, 30, 43, 174, 350, 368, 370, 384, 409, 514
<code>\RenewDocumentEnvironment</code>	332, 459, 481, 482, 487, 528
<code>\RequirePackage</code>	3, 160, 181, 199, 202, 203, 206, 209, 215, 216, 224, 368
<code>\reuseframe</code>	500
<code>\rmdefault</code>	217
<code>\rule</code>	58
rule commands:	
<code>\rule:nnn</code>	48, 360
S	
scan commands:	
<code>\scan_stop:</code>	54
<code>\scriptsize</code>	84
<code>\section</code>	105, 114
seq commands:	
<code>\seq_gpop_left:NN</code>	186
<code>\seq_gpush:Nn</code>	176
<code>\seq_gput_right:Nn</code>	157
<code>\seq_gremove_all:Nn</code>	108, 109, 110
<code>\seq_gset_from_clist:Nn</code>	130
<code>\seq_map_indexed_inline:Nn</code>	116
<code>\seq_map_inline:Nn</code>	137, 144, 149
<code>\seq_new:N</code>	129, 172
<code>\seq_set_from_clist:Nn</code>	114
<code>\l_tmpa_seq</code>	114, 116
<code>\setcounter</code>	24, 322
<code>\setlength</code>	25, 26, 27, 28, 29, 31, 32, 33, 43, 44, 45, 46, 47, 71, 158, 159
<code>\setmainfont</code>	210
<code>\setmathfont</code>	212
<code>\setsansfont</code>	211
<code>\SetTemplateKeys</code>	121
<code>\sfdefault</code>	217
<code>\SimpleBlockEnv</code>	483
<code>\skip</code>	30
skip commands:	
<code>\skip_horizontal:n</code>	32, 262, 309, 338, 355, 361
<code>\skip_if_eq_p:nn</code>	22
<code>\skip_new:N</code>	17
<code>\skip_set:Nn</code>	251, 261
<code>\skip_set_eq:NN</code>	20
<code>\skip_vertical:n</code>	33, 107, 109, 113, 115, 119, 121, 125, 127, 461, 462
<code>\l_tmpa_skip</code>	458, 459, 461, 463
socket commands:	
<code>\socket_use:n</code>	468
<code>\space</code>	19, 21
<code>\stdcolor</code>	4
<code>\stdemph</code>	334
<code>\stdfootnote</code>	173, 178, 179
<code>\stdincludegraphics</code>	368
<code>\stditem</code>	367, 371, 372
<code>\stdmathcolor</code>	4
<code>\stdnewtheorem</code>	513
<code>stdquotation (env.)</code>	323
<code>stdquote (env.)</code>	323
<code>\stdtextbf</code>	334
<code>\stdtextcolor</code>	4
<code>\stdtextit</code>	334
<code>\stdtextmd</code>	334
<code>\stdtextnormal</code>	334
<code>\stdtextrm</code>	334
<code>\stdtextsc</code>	334
<code>\stdtextsf</code>	334
<code>\stdtextsl</code>	334
<code>\stdtexttt</code>	334
<code>\stdtextup</code>	334
<code>stdverse (env.)</code>	323
<code>\stepcounter</code>	21
str commands:	
<code>\str_clear:N</code>	20, 146, 462, 471, 484, 493, 520
<code>\str_if_empty:NTF</code>	96, 117, 127, 419
<code>\str_if_empty_p:N</code>	48
<code>\str_if_eq:nnTF</code>	18, 67, 93, 150, 157, 159
<code>\str_if_eq_p:nn</code>	6, 7, 23, 387
<code>\str_new:N</code>	9, 11, 12, 15, 122, 132
<code>\str_put_right:Nn</code>	155, 191
<code>\str_replace_all:Nnn</code>	21, 23, 111
<code>\str_set:Nn</code>	19, 26, 126, 129, 130
<code>\str_set_eq:NN</code>	162
<code>\string</code>	397
<code>\subsection</code>	105, 114
<code>\subsubsection</code>	105, 114
<code>\subtitle</code>	34
sys commands:	
<code>\c_sys_engine_str</code>	24
<code>\sys_if_engine luatex:TF</code>	204, 429
<code>\sys_if_engine luatex_p:</code>	19, 67, 94
<code>\sys_if_engine opentype:TF</code>	200
<code>\sys_if_engine pdftex_p:</code>	20, 66, 93
<code>\sys_if_engine xetex:TF</code>	76
<code>\sys_if_engine xetex_p:</code>	68, 95
T	
<code>\tabbingsep</code>	29
<code>\tabcolsep</code>	26
<code>table (env.)</code>	139
<code>\tablename</code>	150
<code>\tableofcontents</code>	238
tag commands:	
<code>\tag_get:n</code>	442
<code>\tag_mc_begin:n</code>	189, 196, 449
<code>\tag_mc_end:</code>	187, 194, 455

\tag_resume:n	186, 454	__talk_decode_check:n	148, 195, 195
\tag_struct_begin:n	91, 188, 210, 441	__talk_decode_check:nw	195, 202, 205
\tag_struct_end:	92, 195, 217, 445	__talk_decode_check_range:nnn	
\tag_suspend:n	197, 450		195, 211, 212, 224
tag internal commands:		__talk_decode_check_single:nn	
\g__tag_title_author_tl	19		195, 208, 215
\g__tag_title_title_tl	31	__talk_decode_mode:n	52, 62, 62
\tagpdfparaOff	61	__talk_decode_mode:nn	86, 89, 91
\tagpdfsetup	207, 226	__talk_decode_mode:w	62, 72, 78
talk internal commands:		__talk_decode_mode_aux:n	62
__talk_action_alert:N	42, 42, 106, 111	\l__talk_decode_modes_bool	
__talk_action_begin:n	13, 91, 125,		7, 29, 49, 66, 101
	128, 134, 136, 143, 334, 346, 398, 530	__talk_decode_overlay_:nw	144
__talk_action_begin:w	125, 141, 146	__talk_decode_overlay_aux:nNN	
__talk_action_begin_auxi:n			144, 163, 166, 167
	125, 144, 147, 148	__talk_decode_overlay_offset:nNn	
__talk_action_begin_auxii:n			144, 171, 176, 186, 189
	125, 154, 157	__talk_decode_overlay_offset:nNnN	
__talk_action_end:	34, 29, 96, 125,		144, 175, 178, 187
	130, 135, 148, 179, 340, 363, 394, 538	__talk_decode_overlays:nN	
__talk_action_invisible:N	48, 48, 177		144, 147, 150, 156, 193
__talk_action_invisible_end:N		__talk_decode_overlays:nn	
	48, 54, 194		97, 118, 131, 144, 144
__talk_action_only:N	64, 64, 119	\l__talk_decode_overlays_bool	3,
__talk_action_only_end:N	64, 69, 120		6, 24, 28, 50, 69, 161, 168, 189, 191
\l__talk_action_spec_str		\l__talk_decode_overlays_clist	10
	142, 160, 376, 504	\l__talk_decode_overlays_str	
__talk_action_uncover:N	81, 81, 189		10, 48, 96
__talk_action_uncover_end:N		__talk_decode_parse:n	
	81, 87, 191		5, 16, 16, 159, 188
__talk_action_visible:N	48, 56	__talk_decode_parse:w	16, 36, 43, 54
__talk_action_visible_end:N	48, 62	__talk_decode_parse_auxi:n	
\l__talk_aspect_ratio_str	117, 176		16, 17, 18
\g__talk_cnt_reset_seq		__talk_decode_parse_auxii:n	
	108, 109, 110, 129, 144, 149, 157		16, 30, 33
__talk_cnt_restore:	92, 142, 147	\l__talk_decode_step_bool	
__talk_cnt_save:	83, 142, 142		8, 35, 37, 162
__talk_column_align_bottom:n	54, 54	\l__talk_float_alignment_tl	
__talk_column_align_center:n	54, 56		103, 115, 116, 117, 126, 136
__talk_column_align_top:n	54, 73	\l__talk_fontsize_dim	117, 157, 162
\l__talk_column_alignment_tl	31, 98	\l__talk_footelem_color_tl	199
\g__talk_column_int	9, 15, 16, 27, 81, 82	\l__talk_footelem_font_tl	199
\l__talk_column_int	9, 15, 27	\l__talk_footelem_left_skip	199
\l__talk_columns_wd_tl	5, 18, 19	\l__talk_footelem_right_skip	199
__talk_decode_action:n	95, 104, 104	\l__talk_footer_bg_tl	282
__talk_decode_action:w	104, 106, 111	\l__talk_footer_fg_tl	282
\l__talk_decode_action_str		\l__talk_footer_font_tl	282
	12, 20, 117, 127, 129, 163, 171	\l__talk_footer_left_skip	282
\l__talk_decode_actions_bool		\l__talk_footer_order_clist	282
	13, 27, 165, 173	\l__talk_footer_right_skip	282
\l__talk_decode_actions_clist	13	\l__talk_footer_sep_tl	282
\l__talk_decode_actions_str	13	\g__talk_footnote_box	
\l__talk_decode_arg_str			82, 97, 100, 171, 183, 185
	9, 26, 30, 121, 135, 140, 183		

<code>\g__talk_footnote_overlay_seq</code> ..	
.....	172 , 176 , 186
<code>\l__talk_frame_alignment_tl</code>	
.....	94 , 95 , 159 , 173
<code>\l__talk_frame_bool</code> 128 , 141 , 402 , 418	
<code>\g__talk_frame_int</code>	
.....	15 , 57 , 73 , 263 , 405 , 410 , 417
<code>\l__talk_frame_name_str</code>	
.....	160 , 419 , 421 , 424 , 426 , 427 , 462 , 471 , 484 , 493 , 520
<code>__talk_frame_notag:n</code> ... 46 , 447 , 447	
<code>__talk_frame_overprint:</code>	
.....	261 , 261 , 273 , 276 , 280 , 293 , 295 , 296 , 299 , 301 , 308 , 310 , 315
<code>__talk_frame_process:nn</code>	
....	415 , 415 , 465 , 475 , 487 , 496 , 514
<code>__talk_frame_reuse:nn</code> . 500 , 500 , 521	
<code>__talk_frame_reuse_aux:nn</code>	
.....	500 , 504 , 509
<code>__talk_frame_reuse_aux:nnn</code>	
.....	500 , 510 , 511
<code>\g__talk_frame_struct_int</code> 56 , 76 , 442	
<code>\g__talk_frame_subtitle_tl</code> 3 , 13 , 81	
<code>__talk_frame_tag:n</code> 41 , 439 , 439	
<code>\g__talk_frame_tag_bool</code>	
.....	46 , 403 , 443 , 451
<code>\l__talk_frame_tagging_str</code>	
.....	18 , 19 , 21 , 23 , 37 , 160
<code>__talk_frame_title:n</code> 15 , 38 , 44	
<code>\l__talk_frame_title_bool</code> . 117 , 457	
<code>__talk_frame_title_tagged:n</code> ...	
.....	15 , 47 , 51
<code>\g__talk_frame_title_tl</code>	
.....	3 , 8 , 58 , 80 , 271 , 474
<code>\l__talk_frame_verb_bool</code>	
. 49 , 404 , 430 , 464 , 473 , 486 , 495 , 513	
<code>\l__talk_frametitle_after_skip</code> .	
.....	25 , 41
<code>\l__talk_frametitle_before_skip</code>	
.....	26 , 32
<code>\l__talk_frametitle_color_tl</code> ...	
.....	27 , 34 , 35
<code>\l__talk_frametitle_font_tl</code> .. 28 , 36	
<code>__talk_head_marks:nnnnn</code>	
.....	161 , 174 , 174 , 192
<code>__talk_head_marks_aux:Nnn</code> 174	
<code>__talk_head_marks_aux:Nnnn</code>	
.....	181 , 188 , 193
<code>__talk_head_section:Nnn</code>	114
<code>__talk_head_subsection:Nnn</code> 114	
<code>__talk_head_subsubsection:Nnn</code> . 114	
<code>__talk_head_tag:nn</code> 202 , 205 , 208	
<code>\l__talk_header_bg_tl</code>	235
<code>\l__talk_header_fg_tl</code>	235
<code>\l__talk_header_font_tl</code>	235
<code>\l__talk_header_frametitle_bool</code> 235	
<code>\l__talk_header_ht_dim</code>	235
<code>\l__talk_header_left_skip</code>	235
<code>\l__talk_header_right_skip</code> 235	
<code>__talk_header_tag_begin:n</code>	
.....	53 , 184 , 184 , 191
<code>__talk_header_tag_end:</code> . 64 , 184 , 192	
<code>__talk_if_overlay:n</code>	3 , 10
<code>__talk_if_overlay:nTF</code> 3 , 7 , 12 , 13 , 13 , 23 , 32 , 34 , 37 , 45 , 115 , 130 , 200 , 213 , 223 , 353 , 372 , 387 , 411	
<code>__talk_item_parse_spec:n</code>	
.....	365 , 378 , 382 , 383
<code>__talk_item_parse_spec:w</code>	
.....	365 , 375 , 381
<code>__talk_label:n</code>	384 , 388 , 391
<code>__talk_latex_frame:n</code> .. 27 , 414 , 414	
<code>\l__talk_list_end_tl</code>	
.....	389 , 395 , 401 , 414 , 452 , 474
<code>__talk_metadata_name:n</code>	
.....	323 , 326 , 331 , 347 , 347
<code>__talk_mode:n</code>	3
<code>__talk_mode:nTF</code>	3 , 12
<code>\l__talk_mode_str</code> 7 , 68 , 93 , 117	
<code>\c__talk_modes_clist</code>	56 , 64
<code>__talk_onslide:n</code> . 204 , 206 , 209 , 238	
<code>\g__talk_onslide_tl</code>	
..	85 , 89 , 166 , 183 , 211 , 212 , 216 , 220
<code>__talk_opacity_begin:n</code>	
...	38 , 38 , 51 , 52 , 59 , 60 , 79 , 84 , 215
<code>__talk_opacity_end:</code>	
.....	38 , 40 , 55 , 63 , 88 , 192 , 217
<code>__talk_opacity_group_begin:</code> ...	
.....	415 , 415 , 431
<code>__talk_opacity_group_end:</code>	
.....	415 , 417 , 432
<code>\g__talk_opacity_group_int</code>	
.....	414 , 421 , 423 , 427
<code>\l__talk_overlay_all_bool</code>	
.....	124 , 151 , 153 , 181
<code>__talk_overlay_arg:n</code>	
.....	3 , 11 , 92 , 99 , 103 , 110 , 118
<code>__talk_overprint_begin:n</code>	
.....	242 , 242 , 250 , 267
<code>__talk_overprint_check_ht:n</code> ...	
.....	266 , 315 , 317 , 326
<code>\l__talk_overprint_int</code> . 260 , 264 , 270	
<code>__talk_overprint_save_ht:</code>	
.....	266 , 271 , 291
<code>__talk_pagecolor:n</code> 43 , 48 , 49 , 52	
<code>\c__talk_paper_height_dim</code>	164
<code>\c__talk_paper_width_dim</code>	164

<code>\g__talk_pauses_int</code>	85, 99, 115 , 245 , 255 , 277 , 282 , 286 ,
..... 11 , 4 , 40 , 79 , 192 , 236 , 237 , 239	288 , 302 , 310 , 337 , 348 , 361 , 416 , 425
<code>\l__talk_saved_action_str</code>	<code>\l__talk_tmp_tl</code> 13 , 19 ,
..... 121 , 162 , 188	22 , 24 , 112 , 116 , 187 , 188 , 320 , 322 , 323
<code>\l__talk_saved_actions_bool</code>	<code>__talk_toc_aux:nnnn</code>
..... 121 , 164 , 190 244 , 245 , 248 , 258 , 267
<code>\l__talk_saved_overlays_bool</code> ...	<code>__talk_toc_dest:n</code> 244 , 271 , 274
..... 121 , 160 , 185	<code>__talk_toc_dest:w</code> 244 , 276 , 279
<code>\l__talk_sec_bookmark_tl</code> 72 , 151 , 163	<code>__talk_toc_level:nnnn</code> . 244 , 272 , 299
<code>\l__talk_sec_label_tl</code> 72	<code>\l__talk_uncover_hidden_fp</code> 74
<code>\l__talk_sec_nameref_tl</code> 72 , 164	<code>\l__talk_vcenter_offset_tl</code> 64 , 69 , 75
<code>\l__talk_sec_numbered_bool</code> . 72 , 154	<code>__talk_wallpaper_hruler:Nnn</code>
<code>\l__talk_sec_quote_tl</code> 72 258 , 305 , 353 , 353
<code>\l__talk_sec_running_tl</code> 72 , 152	<code>talk/sec/title</code> 202
<code>\l__talk_sec_subtitle_tl</code> 72	<code>\temporal</code> 221
<code>\l__talk_sec_toc_tl</code> 72 , 153 , 162	TeX and L ^A T _E X 2 _ε commands:
<code>\g__talk_section_tl</code> 66	<code>\@arabic</code> .. 6 , 8 , 111 , 112 , 113 , 223 , 407
<code>\l__talk_section_tl</code> 66	<code>\@author</code> 3 , 18 , 19
<code>\l__talk_shuffle_skip</code> .. 17 , 20 , 22 , 34	<code>\@auxout</code> 306 , 395
<code>__talk_shuffle_skip:n</code> . 18 , 18 , 39 , 41	<code>\@bsphack</code> 386
<code>__talk_slide:nn</code> 10 , 10 , 437	<code>\@caption</code> 160
<code>__talk_slide_align_bottom:n</code> 105 , 105	<code>\@capttype</code> 130
<code>__talk_slide_align_center:n</code> 105 , 111	<code>\@contentsline@destination</code>
<code>__talk_slide_align_stretch:n</code> 59 , 276 , 307 , 310 , 313 , 316
..... 105 , 117	<code>\@currentHref</code> 402
<code>__talk_slide_align_top:n</code> . 105 , 123	<code>\@currentlabel</code> 399
<code>__talk_slide_aux:n</code> 10 , 50 , 61	<code>\@currentlabelname</code> 176 , 401
<code>__talk_slide_begin:</code> 34 , 77 , 77	<code>\@currentenv</code> 451
<code>\l__talk_slide_box</code> 5 , 84 , 96	<code>\@date</code> 3 , 25
<code>\g__talk_slide_continue_bool</code> .. 3 ,	<code>\@definecounter</code> 152
28 , 31 , 36 , 39 , 44 , 91 , 221 , 226 , 229 , 235	<code>\@endparpenalty</code> 465
<code>\l__talk_slide_continue_bool</code> ...	<code>\@esphack</code> 389
..... 3 , 35 , 40 , 45	<code>\@evenfoot</code> 373 , 388 , 399
<code>__talk_slide_end:</code> 54 , 77 , 87	<code>\@evenhead</code> 372 , 387 , 398
<code>\g__talk_slide_int</code>	<code>\@framenummer</code> 405
.. 6 , 9 , 26 , 30 , 217 , 220 , 226 , 228 , 233	<code>\@ignore</code> 34
<code>\g__talk_subsection_tl</code> 66	<code>\@ignoretrue</code> 101
<code>\l__talk_subsection_tl</code> 66 , 158	<code>\@inmatherr</code> 451
<code>\g__talk_subsubsection_tl</code> 66	<code>\@input</code> 225
<code>\l__talk_subsubsection_tl</code> .. 66 , 160	<code>\@institute</code> 3 , 37
<code>__talk_textcmd_equiv:n</code> . 334 , 355 , 359	<code>\@itempenalty</code> 417
<code>\l__talk_titlelem_after_skip</code> 44	<code>\@kernel@reserved@label@data</code> ... 403
<code>\l__talk_titlelem_before_skip</code> ... 44	<code>\@listI</code> 56
<code>\l__talk_titlelem_color_tl</code> 44	<code>\@listi</code> 49 , 56
<code>\l__talk_titlelem_font_tl</code> 44	<code>\@listii</code> 57
<code>\l__talk_titlelem_tag_begin_tl</code> .. 44	<code>\@listiii</code> 64
<code>\l__talk_titlelem_tag_end_tl</code> 44	<code>\@makecaption</code> 167
<code>\l__talk_titlepage_alignment_tl</code> . 94	<code>\@makefntext</code> 195
<code>\l__talk_titlepage_framestyle_tl</code> 94	<code>\@mpfootins</code> 30
<code>\l__talk_titlepage_order_clist</code> .. 94	<code>\@nobreakfalse</code> 235
<code>__talk_tmp:w</code> 114 , 114 , 167 , 176	<code>\@noitemerr</code> 444
<code>\l__talk_tmp_box</code> 18 ,	<code>\@oddfoot</code> . 371 , 373 , 382 , 388 , 397 , 399
26 , 59 , 60 , 62 , 63 , 66 , 67 , 68 , 71 ,	<code>\@oddhead</code> . 367 , 372 , 377 , 387 , 392 , 398
	<code>\@outerparskip</code> 463

\@parboxrestore	88, 164	\tex_lastskip:D	20
\@setminipage	165	\tex_setbox:D	31, 42
\@shortauthor	3	\tex_unskip:D	29
\@shortdate	3	\tex_vbox:D	31, 42
\@shortinstitute	3	\tex_vrule:D	50
\@shortsubtitle	3	\texorpdfstring	173, 189
\@shorttitle	3	text commands:	
\@skiphyperreftrue	35, 131	\text_purify:n	58, 112, 206
\@starttoc	220, 241	\text_titlecase_first:n	152
\@subtitle	3, 42	\textasteriskcentered	40
\@title	3, 30, 31	\textbf	334
\@totalframes	409	\textbullet	38
\c@figure	150	\textcolor	6, 11
\c@frame	405	\textendash	39
\c@page	223	\textheight	93
\c@pauses	4	\textit	334
\c@section	111	\textmd	334
\c@slide	6	\textnormal	334
\c@subsection	112	\textperiodcentered	41
\c@subsubsection	113	\textrm	334
\c@table	150	\textsc	334
\check@mathfonts	58	\textsf	334
\currentgrouplevel	62	\textsl	334
\fnum@figure	150	\texttt	334
\fnum@table	150	\textup	334
\Gm@bmargin	308	\textwidth	32, 8, 19, 20, 86, 87, 266
\Gm@lmargin	242, 289, 355	\theenumi	34
\Gm@rmargin	244, 290, 338	\theenumii	35
\Gm@tmargin	241	\theenumiii	36
\if@minipage	165	\theenumiv	37
\ifmeasuring@	12	\thefigure	150
\ignorespaces	34	\theframe	405
\l@section	244	\thepage	7, 223, 400
\l@subsection	244	\thepauses	4
\l@subsubsection	244	\thesection	105
\label@in@display	49, 408, 409	\theslide	6
\on@line	425	\thesubsection	105
\protected@write	395	\thesubsubsection	105
\ps@plain	365	\thetable	150
\ps@talk	365	\thispagestyle	129
\ps@wallpaper	365	\tiny	288
\reset@color	62, 63	\title	15
\set@color	61, 63	tl commands:	
\std@definecounter	152	\tl_clear:N	151, 152, 153, 158, 160, 395
\std@label@in@display	408	\tl_clear_new:N	426
\stdreset@color	61	\tl_gclear:N	12, 13, 80, 81, 85, 212
\stdset@color	61	\tl_gput_right:Nn	216
\textsubscript@offset	219	\tl_gset:Nn	8,
\textsubscript@space	219		13, 18, 25, 30, 37, 42, 296, 299, 427, 474
\textsuperscript@offset	219	\tl_gset_eq:NN	14, 19, 31
\textsuperscript@space	219	\tl_if_blank:nTF	37,
tex commands:			84, 100, 115, 130, 177, 179, 186, 210
\tex_currentgrouplevel:D	391, 392	\tl_if_blank_p:n	22, 386
\tex_hsize:D	33, 44		

<code>\tl_if_empty:N</code>	NTF	34, 68, 128, 166, 183, 220, 264, 317, 326, 356
<code>\tl_if_empty:n</code>	NTF	64, 207, 215
<code>\tl_if_exist:N</code>	NTF	293, 349, 421, 502
<code>\tl_if_novalue:n</code>	NTF	139
<code>\tl_map_inline:nn</code>		334
<code>\tl_new:N</code>		3, 4, 49, 66, 67, 68, 69, 70, 71, 75, 103, 104, 116, 151, 153, 159, 220, 295, 401
<code>\tl_put_right:N</code>		77
<code>\tl_retokenize:n</code>		68
<code>\tl_set:Nn</code>		13, 71, 72, 76, 115, 116, 117, 118, 121, 122, 123, 124, 147, 152, 176, 389
<code>\tl_set_eq:NN</code>		45, 154, 173
<code>\tl_to_str:n</code>		58, 59, 60, 72, 87, 107, 112, 168, 177, 476
<code>\tl_trim_spaces:n</code>		53
<code>\tl_use:N</code>		89, 122, 211
<code>\today</code>		3
token commands:		
<code>\token_if_eq_meaning:NNTF</code>		174, 185
<code>\token_to_str:N</code>		79, 80
<code>\topsep</code>		52, 60, 67
U		
<code>\uncover</code>		89
<code>\uncoverenv (env.)</code>		98
<code>\unskip</code>		24
use commands:		
<code>\use:N</code>		95, 98, 137, 140, 144, 198, 213, 513
<code>\use:n</code>		48, 51, 65, 76, 109, 118, 123, 165, 179, 270, 374
<code>\use_none:n</code>		172, 189
<code>\usebox</code>		425
<code>\UseHookWithArguments</code>		306, 309, 311, 314, 394
<code>\UseInstance</code>		85, 134, 144, 175, 270, 322, 330, 379, 384, 394, 397, 490, 491
<code>\UseStructureName</code>		128, 143, 213
<code>\UseTaggingSocket</code>		137, 138, 156, 224, 226
<code>\UseTemplate</code>		142, 145
V		
<code>\value</code>		197, 269, 284, 285, 289, 290, 301
vbox commands:		
<code>\vbox:n</code>		55
<code>\vbox_gset:Nn</code>		183
<code>\vbox_set:Nn</code>		59
<code>\vbox_set:Nw</code>		67, 84, 348
<code>\vbox_set_end:</code>		72, 90, 97, 252, 269, 360
<code>\vbox_set_to_wd:Nnn</code>		29, 310
<code>\vbox_set_to_wd:Nnw</code>		38, 85, 245
<code>\vbox_to_ht:nn</code>		93, 113, 253, 279
<code>\vbox_top:n</code>		74
<code>\vbox_unpack:N</code>		185
<code>\vbox_unpack_drop:N</code>		96, 99, 100
vcoffin commands:		
<code>\vcoffin_set:Nnn</code>		2
<code>verse (env.)</code>		323
<code>\vfil</code>		256, 283, 319
<code>\visible</code>		89
<code>\visibleenv (env.)</code>		98
<code>\vspace</code>		32, 41, 66, 76
Y		
<code>\year</code>		22