

ltx-talk – A class for typesetting (accessible) presentations*

Joseph Wright[†]

Released 2026-09-16

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	2
1.4	Scratch space	2
1.5	Option handling	3
1.6	Setting up	4
1.7	Math support	5
1.8	Font selection	5
1.9	Text scripts	5
1.10	Hyperlinks	6
1.11	Tagging	6
II	ltx-talk-color – Color definitions	7
1	ltx-talk-color implementation	7
1.1	Existing definitions	7
1.2	Document (and interface) commands	7
1.3	Color definition	9
1.4	Semantic colors	9
III	ltx-talk-decode – Decoding overlay specs	10
1	ltx-talk-decode implementation	10
IV	ltx-talk-frame – The structure of frames	18

*This file describes v0.6.5, last revised 2026-09-16.

[†]E-mail: joseph@texdev.net

1	ltx-talk-frame implementation	18
1.1	Slides in frames	18
1.2	Counters	23
1.3	Frame options	24
1.4	Tagging for headers	24
1.5	Wallpaper	25
1.6	The <code>frame</code> environment	29
V	ltx-talk-frame – The structure of frames	34
1	ltx-talk-frame-structure implementation	34
1.1	Columns	34
1.2	Floats	36
1.3	Footnotes	38
VI	ltx-talk-mode – Modes	40
1	ltx-talk-mode implementation	40
VII	ltx-talk-overlay – Overlays	41
1	ltx-talk-overlay implementation	41
1.1	Utilities	41
1.2	Opacity utilities	42
1.3	Action commands and environments	42
1.4	Non-action commands and environments	47
1.5	Fixed-size areas	48
1.6	Adding overlays to existing commands	50
1.7	Overlay patching of third-party environments	52
VIII	ltx-talk-required – “Required” definitions	54
1	ltx-talk-required implementation	54
1.1	Standard design settings	54
1.2	List support	55
IX	ltx-talk-structure – Structural commands	56

1	ltx-talk-structure implementation	56
1.1	Frame title	56
1.2	Headings	57
1.3	Table of contents	61
1.4	Block environments	64
1.5	Lists	65
1.6	Theorems, <i>etc.</i>	68
1.7	Bibliographies	68
1.8	Supplementary material	69
X	ltx-talk-title – Title pages	70
1	ltx-talk-title implementation	70
	Index	74

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} {2026-09-16} {0.6.5}
4   {A class for typesetting (accessible) 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 { 2026-06-01 }
8   {
9     \msg_new:nnnn { ltx-talk } { kernel-too-old }
10    { The~ltx-talk~class~requires~LaTeX~2026-06-01~or~later. }
11    {
12      You~have~tried~to~use~the~ltx-talk~class~with~a~LaTeX~kernel~release~
13      prior~to~2026-06-01;~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 }

```

1.3 Extra variants

```

56 \cs_generate_variant:Nn \hook_gput_code:nnn { nne }
57 \exp_args_generate:n { nVv }
58 \cs_generate_variant:Nn \dim_max:nn { v }
59 \cs_generate_variant:Nn \str_replace_all:Nnn { NnV }
60 \cs_generate_variant:Nn \vbox_to_ht:nn { v }

```

1.4 Scratch space

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

```

61 \cs_new_protected:Npn \__talk_tmp:w { }

```

(End of definition for `\__talk_tmp:w`.)

`\l__talk_tmp_box`

```

62 \box_new:N \l__talk_tmp_box

```

(End of definition for `\l__talk_tmp_box`.)

`\l__talk_tmp_clist`

```

63 \clist_new:N \l__talk_tmp_clist

```

(End of definition for \l__talk_tmp_clist.)

\l__talk_tmp_tl

64 \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

\l__talk_suppl_mode_str

65 \keys_define:nn { talk }

66 {

67 aspect-ratio .str_set:N =

68 \l__talk_aspect_ratio_str ,

69 font-size .dim_set:N =

70 \l__talk_fontsize_dim ,

71 frame-title-arg .bool_set:N =

72 \l__talk_frame_title_bool ,

73 handout .code:n =

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

75 handout .value_forbidden:n = true ,

76 mode .choices:nn =

77 { handout , projector }

78 { \str_set_eq:NN \l__talk_mode_str \l_keys_choice_str } ,

79 supplementary-frames .choices:nn =

80 { appendix , in-place , omit }

81 { \str_set_eq:NN \l__talk_suppl_mode_str \l_keys_choice_str }

82 }

83 \str_new:N \l__talk_mode_str

84 \str_new:N \l__talk_suppl_mode_str

(End of definition for \l__talk_aspect_ratio_str and others.)

Scope for options.

85 \keys_define:nn { talk }

86 {

87 aspect-ratio .usage:n = load ,

88 font-size .usage:n = load ,

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

90 mode .usage:n = load ,

91 supplementary-frames .usage:n = load

92 }

Compatibility keys for classical font size setting.

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

94 {

95 \keys_define:nn { talk }

96 {

97 #1 .meta:n = { font-size = #1 } ,

98 #1 .value_forbidden:n = true ,

99 #1 .usage:n = load

100 }

101 }

Initial values.

```

102 \keys_set:nn { talk }
103 {
104   aspect-ratio      = 16:9      ,
105   font-size         = 11pt      ,
106   frame-title-arg   = false     ,
107   mode              = projector ,
108   supplementary-frames = appendix
109 }
110 \ProcessKeyOptions [ talk ]

```

1.6 Setting up

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

```

111 \file_if_exist_input:nF { size \dim_to_decimal:n \l__talk_fontsize_dim .clo }
112 {
113   \file_input:n { size10.clo }
114   \RequirePackage { relsize }
115   \hook_gput_code:nne { begindocument } { talk }
116   { \exp_not:N \relsize { \fp_eval:n { \l__talk_fontsize_dim / 10pt } } } }
117 }

```

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

```

118 \dim_const:Nn \c__talk_paper_height_dim { 100mm }
119 \use:e
120 {
121   \cs_set_protected:Npn \exp_not:N \__talk_tmp:w
122     #1 \tl_to_str:n { : } #2 \tl_to_str:n { : } #3 \exp_not:N \q_stop
123   {
124     \dim_const:Nn \exp_not:N \c__talk_paper_width_dim
125     {
126       \exp_not:N \fp_to_dim:n
127       { (#1 / #2) * \exp_not:N \c__talk_paper_height_dim }
128     }
129   }
130   \exp_not:N \__talk_tmp:w \l__talk_aspect_ratio_str
131   \tl_to_str:n { : } 100 \exp_not:N \q_stop
132 }
133 \use:e
134 {
135   \exp_not:N \RequirePackage
136   [
137     papersize =
138     {
139       \dim_use:N \c__talk_paper_width_dim ,
140       \dim_use:N \c__talk_paper_height_dim
141     } ,
142     tmargin    = 10mm ,
143     bmargin    = 8mm ,
144     lmargin    = 10mm ,
145     rmargin    = 10mm ,
146     headheight = 10mm ,

```

```

147         headsep    = 2mm ,
148         footskip   = 6mm
149     ]
150     { geometry }
151 }

```

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

Turn off justification

```

152 \raggedright

```

1.7 Math support

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

```

153 \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.

```

154 \sys_if_engine_opentype:TF
155 {
156     \RequirePackage { fontspec }
157     \RequirePackage { mathtools }
158     \sys_if_engine luatex:TF
159     {
160         \RequirePackage { lua-unicode-math }
161         \tagpdfsetup { math / mathml / luamml / load = true }
162     }
163     { \RequirePackage { unicode-math } }
164     \setmainfont { NewCMSans10-Regular.otf }
165     \setsansfont { NewCMSans10-Regular.otf }
166     \setmathfont { NewCMSansMath-Regular.otf }
167 }
168 {
169     \RequirePackage { sansmathfonts }
170     \RequirePackage [ nomath ] { lmodern }
171     \cs_set_eq:NN \rmdefault \sfdefault
172 }

```

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 ConT_EXt, no additional spacing added after script items.
`\textsubscript@space` 173 `\cs_set_nopar:Npn \textsubscript@offset { 0.48ex }`
`\textsuperscript@offset` 174 `\cs_set_nopar:Npn \textsubscript@space { }`
`\textsuperscript@space` 175 `\cs_set_nopar:Npn \textsuperscript@offset { 0.86ex }`
176 `\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.
177 `\cs_new:Npn \thepage { \@arabic \c@page }`

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

A requirement.

178 `\RequirePackage { hyperref }`
179 `\hypersetup { hidelinks }`

1.11 Tagging

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

180 `\tagpdfsetup`
181 `{`
182 `role / user-NS = ltx-talk                    ,`
183 `role / new-tag = frame / Sect                ,`
184 `role / new-tag = frametitle / H4`
185 `}`

186 `\</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 \endclass
```

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.)

`\l__talk_decode_trailing_bool` Indicates the current slide trailing after all of those in the overlay specification: needed for `\temporal`.

```
4 \bool_new:N \l__talk_decode_trailing_bool
```

(End of definition for \l__talk_decode_trailing_bool.)

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

```
\c@pauses 5 \int_new:N \g__talk_pauses_int
\thepauses 6 \cs_new_eq:NN \c@pauses \g__talk_pauses_int
7 \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_mode_bool` Tracks whether the overlay applies purely on the basis of mode: needed to handle inclusion/exclusion of entire frame environment. Will be `true` even if the slide numbers mean the overlay does not otherwise apply.

```
8 \bool_new:N \l__talk_decode_mode_bool
```

(End of definition for \l__talk_decode_mode_bool.)

`\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.

```
9 \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`.

```
10 \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.

```
11 \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

```
12 \clist_new:N \l__talk_decode_overlays_clist
13 \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.

```
14 \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
15 \bool_new:N \l__talk_decode_actions_bool
\l__talk_decode_actions_str
16 \clist_new:N \l__talk_decode_actions_clist
17 \str_new:N \l__talk_decode_actions_str
```

(End of definition for \l__talk_decode_actions_bool, \l__talk_decode_actions_clist, and \l__talk_decode_actions_str.)

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

```
18 \cs_new_protected:Npn \__talk_decode_parse:n #1
19 { \exp_args:Ne \__talk_decode_parse_auxi:n {#1} }
20 \cs_new_protected:Npn \__talk_decode_parse_auxi:n #1
21 {
22   \str_clear:N \l__talk_decode_action_str
23   \bool_lazy_or:nnTF
24     { \tl_if_blank_p:n {#1} }
25     { \str_if_eq_p:nn {#1} { all } }
26     {
27       \bool_set_true:N \l__talk_decode_mode_bool
28       \bool_set_true:N \l__talk_decode_overlays_bool
29     }
30     {
31       \str_set:Nn \l__talk_decode_arg_str {#1}
32       \bool_set_false:N \l__talk_decode_actions_bool
33       \bool_set_false:N \l__talk_decode_overlays_bool
34       \bool_set_false:N \l__talk_decode_trailing_bool
35       \bool_set_false:N \l__talk_decode_mode_bool
36       \bool_set_false:N \l__talk_decode_modes_bool
37       \clist_clear:N \l__talk_decode_actions_clist
38       \clist_clear:N \l__talk_decode_overlays_clist
39       \exp_args:No \__talk_decode_parse_auxii:n { \l__talk_decode_arg_str }
40     }
41 }
```

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.

```
42 \cs_new_protected:Npn \__talk_decode_parse_auxii:n #1
43 {
```

```

44 \bool_set_false:N \l__talk_decode_step_bool
45 \__talk_decode_parse:w #1 | \q_recursion_tail | \q_recursion_stop
46 \bool_if:NT \l__talk_decode_step_bool
47 {
48   \legacy_if:nF { measuring@ }
49   { \int_gincr:N \g__talk_pauses_int }
50 }
51 }

```

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

```

52 \cs_new_protected:Npn \__talk_decode_parse:w #1 |
53 {
54   \quark_if_recursion_tail_stop_do:nn {#1}
55   {
56     \bool_lazy_and:nnT
57     { \str_if_empty_p:N \l__talk_decode_overlays_str }
58     { ! \l__talk_decode_modes_bool }
59     {
60       \bool_set_true:N \l__talk_decode_mode_bool
61       \bool_set_true:N \l__talk_decode_overlays_bool
62     }
63   }
64   \exp_args:Ne \__talk_decode_mode:n
65   { \tl_trim_spaces:n {#1} }
66   \__talk_decode_parse:w
67 }

```

(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.

```

68 \clist_const:Ne \c__talk_modes_clist
69 {
70   \tl_to_str:n { article } ,
71   \tl_to_str:n { handout } ,
72   \tl_to_str:n { projector }
73 }

```

(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. Also notice that if frame breaking is active we only look for overall mode specifications, not any further structure.

```

74 \cs_new_protected:Npe \__talk_decode_mode:n #1
75 {
76   \clist_if_in:NnTF \exp_not:N \c__talk_modes_clist {#1}
77   {
78     \bool_set_true:N \exp_not:N \l__talk_decode_modes_bool
79     \exp_not:N \str_if_eq:VnT
80     \exp_not:N \l__talk_mode_str {#1}
81     {
82       \bool_set_true:N \exp_not:N \l__talk_decode_mode_bool

```

```

83         \bool_set_true:N \exp_not:N \l__talk_decode_overlays_bool
84     }
85 }
86 {
87     \exp_not:N \bool_if:NF \exp_not:N \l__talk_frame_break_bool
88     {
89         \exp_not:N \__talk_decode_mode:w #1 \tl_to_str:n { : : }
90         \exp_not:N \q_stop
91     }
92 }
93 }
94 \use:e
95 {
96     \cs_new_protected:Npe \exp_not:N \__talk_decode_mode:w
97     #1 \token_to_str:N :
98     #2 \token_to_str:N :
99     #3 \exp_not:N \q_stop
100 }
101 {
102     \exp_not:N \tl_if_blank:nTF {#2}
103     {
104         \exp_not:N \__talk_decode_mode:nn
105         { \tl_to_str:n { projector } } {#1}
106     }
107     { \exp_not:N \__talk_decode_mode:nn {#1} {#2} }
108 }
109 \cs_new_protected:Npn \__talk_decode_mode:nn #1#2
110 {
111     \str_if_eq:VnTF \l__talk_mode_str {#1}
112     {
113         \__talk_decode_action:n {#2}
114         \str_if_empty:NT \l__talk_decode_overlays_str
115         { \__talk_decode_overlays:nn { overlays } { * } }
116     }
117     {
118         \tl_if_blank:nT {#2}
119         { \bool_set_true:N \l__talk_decode_modes_bool }
120     }
121 }

```

(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.

```

122 \cs_new_protected:Npe \__talk_decode_action:n #1
123 {
124     \exp_not:N \__talk_decode_action:w
125     #1 \tl_to_str:n { @ @ } \exp_not:N \q_stop
126 }
127 \use:e
128 {
129     \cs_new_protected:Npn \exp_not:N \__talk_decode_action:w
130     #1 \tl_to_str:n { @ } #2 \tl_to_str:n { @ } #3 \exp_not:N \q_stop

```

```

131 }
132 {
133   \tl_if_blank:nTF {#2}
134   {
135     \str_if_empty:NTF \l__talk_decode_action_str
136     { \__talk_decode_overlays:nn { overlays } {#1} }
137     {
138       \msg_error:nnV { talk } { misplaced-action-spec }
139       \l__talk_decode_arg_str
140     }
141   }
142   {
143     \cs_if_exist:cTF { __talk_action_ #1 :N }
144     {
145       \str_if_empty:NTF \l__talk_decode_action_str
146       {
147         \str_set:Nn \l__talk_decode_action_str {#1}
148         \tl_if_blank:nF {#2}
149         {
150           \__talk_decode_overlays:nn { actions } {#2}
151           \clist_if_empty:NT \l__talk_decode_overlays_clist
152           { \bool_set_true:N \l__talk_decode_overlays_bool }
153         }
154       }
155       {
156         \msg_error:nnV { talk } { duplicate-action-spec }
157         \l__talk_decode_arg_str
158       }
159     }
160     {
161       \msg_error:nnV { talk } { bad-action-spec }
162       \l__talk_decode_arg_str
163     }
164   }
165 }

```

(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). If the overlay spec is not simply 0, the current frame is active and is marked as such.

```

\__talk_decode_overlay_+:nw
\__talk_decode_overlay_.:nw
  \__talk_decode_overlay_aux:nN
  \__talk_decode_overlay_offset:nNn
  \__talk_decode_overlay_offset:nNn
166 \cs_new_protected:Npn \__talk_decode_overlays:nn #1#2
167 {
168   \str_clear:c { l__talk_decode_ #1 _str }
169   \bool_lazy_and:nnT
170   { \str_if_eq_p:nn {#1} { overlays } }
171   { ! \str_if_eq_p:nn {#2} { 0 } }
172   { \bool_set_true:N \l__talk_decode_mode_bool }
173   \__talk_decode_overlays:nN {#1} #2 \q_recursion_tail \q_recursion_stop
174   \__talk_decode_check:n {#1}
175 }
176 \cs_new_protected:Npn \__talk_decode_overlays:nN #1#2
177 {

```

```

178 \quark_if_recursion_tail_stop:N #2
179 \cs_if_exist_use:cF { __talk_decode_overlay_ #2 :nw }
180 {
181   \str_put_right:cn { l__talk_decode_ #1 _str } {#2}
182   \__talk_decode_overlays:nN
183 }
184 {#1}
185 }
186 \cs_new_protected:cpn { __talk_decode_overlay_+:nw } #1
187 {
188   \bool_set_true:N \l__talk_decode_step_bool
189   \__talk_decode_overlay_aux:nNN {#1} 1
190 }
191 \cs_new_protected:cpn { __talk_decode_overlay_.:nw } #1
192 { \__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 (.

```

193 \cs_new_protected:Npn \__talk_decode_overlay_aux:nNN #1#2#3
194 {
195   \quark_if_recursion_tail_stop_do:Nn #3
196   {
197     \__talk_decode_overlay_offset:nNn {#1} #2 { 0 }
198     \q_recursion_tail \q_recursion_stop
199   }
200   \token_if_eq_meaning:NNTF #3 ( % )
201   { \__talk_decode_overlay_offset:nNn {#1} #2 { } }
202   { \__talk_decode_overlay_offset:nNn {#1} #2 { 0 } #3 }
203 }

```

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.

```

204 \cs_new_protected:Npn \__talk_decode_overlay_offset:nNnN #1#2#3#4
205 {
206   \quark_if_recursion_tail_stop_do:Nn #4
207   {
208     \msg_error:nnV { talk } { bad-action-spec }
209     \l__talk_decode_arg_str
210   } % (
211   \token_if_eq_meaning:NNTF #4 )
212   { \__talk_decode_overlay_offset:nNn {#1} #2 {#3} }
213   { \__talk_decode_overlay_offset:nNn {#1} #2 {#3#4} }
214 }

```

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

```

215 \cs_new_protected:Npn \__talk_decode_overlay_offset:nNn #1#2#3
216 {
217   \str_put_right:ce { l__talk_decode_ #1 _str }
218   { \int_max:nn { 0 } { #3 + \g__talk_pauses_int + #2 } }
219   \__talk_decode_overlays:nN {#1}
220 }

```

(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.

```
221 \cs_new_protected:Npn __talk_decode_check:n #1
222 {
223   \clist_if_empty:cF { l__talk_decode_ #1 _clist }
224   {
225     \msg_error:nnV { talk } { bad-action-spec }
226     \l__talk_decode_arg_str
227   }
228   \clist_set:cv { l__talk_decode_ #1 _clist } { l__talk_decode_ #1 _str }
229   \clist_if_in:cnTF { l__talk_decode_ #1 _clist } { * }
230   { \bool_set_true:c { l__talk_decode_ #1 _bool } }
231   {
232     \clist_map_inline:cn { l__talk_decode_ #1 _clist }
233     { __talk_decode_check:nw {#1} 0 ##1 - - \q_stop }
234   }
235 }
```

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.

```
236 \cs_new_protected:Npn __talk_decode_check:nw #1#2 - #3 - #4 \q_stop
237 {
238   \tl_if_empty:nTF {#4}
239   { __talk_decode_check_single:nn {#1} {#2} }
240   {
241     \tl_if_blank:nTF {#3}
242     { __talk_decode_check_range:nnn {#1} {#2} { \c_max_int } }
243     { __talk_decode_check_range:nnn {#1} {#2} {#3} }
244   }
245 }
246 \cs_new_protected:Npn __talk_decode_check_single:nn #1#2
247 {
248   \int_compare:nNnTF \g__talk_slide_int = {#2}
249   { \bool_set_true:c { l__talk_decode_ #1 _bool } }
250   {
251     \int_compare:nNnTF {#2} > \g__talk_slide_int
252     { \bool_gset_true:N \g__talk_slide_continue_bool }
253     { \bool_set_true:N \l__talk_decode_trailing_bool }
254   }
255 }
```

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.

```
256 \cs_set_protected:Npn __talk_decode_check_range:nnn #1#2#3
257 {
258   \int_compare:nNnTF \g__talk_slide_int > {#3}
259   { \bool_set_true:N \l__talk_decode_trailing_bool }
```

```

260     {
261       \int_compare:nNnTF \g__talk_slide_int < {#2}
262       { \bool_gset_true:N \g__talk_slide_continue_bool }
263       {
264         \bool_set_true:c { l__talk_decode_ #1 _bool }
265         \bool_lazy_and:nnT
266           { \int_compare_p:nNn \g__talk_slide_int < {#3} }
267           { \int_compare_p:nNn {#3} < \c_max_int }
268           { \bool_gset_true:N \g__talk_slide_continue_bool }
269         \clist_map_break:
270       }
271     }
272   }

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

273 \msg_new:nnnn { talk } { bad-action-spec }
274   { Bad~overlay~specification~"#1". }
275   {
276     The~overlay~specification~given~doesn't~follow~the~pattern~described~in~
277     the~ltx-talk-manual:~it~has~been~ignored.
278   }
279 \msg_new:nnnn { talk } { duplicate-action-spec }
280   { Duplicate~action~in~overlay~specification~"#1". }
281   {
282     The~overlay~specification~contains~more~than~one~action:
283     ltx-talk-only~supports~a~single~action~in~one~overlay.
284   }
285 \msg_new:nnnn { talk } { misplaced-action-spec }
286   { Misplaced~action~in~overlay~specification~"#1". }
287   {
288     The~overlay~specification~contains~an~action~before~the~overlay~
289     part(s):~this~is~not~supported.
290   }
291 </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
\l__talk_slide_continue_bool
Tracks whether the frame continues after the current slide.
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.)

\g__talk_break_box
\l__talk_slide_box
5 \box_new:N \g__talk_break_box
6 \box_new:N \l__talk_break_box

(End of definition for \g__talk_break_box and \l__talk_slide_box.)

\l__talk_slide_box
7 \box_new:N \l__talk_slide_box

(End of definition for \l__talk_slide_box.)

\g__talk_slide_int
\c@slide
\theslide
The slide number inside the current frame: needed to know which overlays are active.
We also provide LATEX counter-style access.
8 \int_new:N \g__talk_slide_int
9 \cs_new_eq:NN \c@slide \g__talk_slide_int
10 \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.
11 \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
`\__talk_slide_auxi:nn` fragile frames. There, all `\obeyedline` in the grabbed tokens need to be turned back into
`\__talk_slide_auxii:n` `^~M` before rescanning: this ensures that any verbatim grabbing in the frame still works.
`\__talk_slide_auxiii:` The strange business with setting the continuation boolean is needed as otherwise we get
`\__talk_slide_auxiv:n` 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.

```

12 \cs_new_protected:Npn \__talk_slide:nn #1#2
13 {
14   \group_begin:
15     \tl_set:Nx \l__talk_tmp_tl
16     {
17       \property_ref:ee { frame . \int_use:N \g__talk_frame_int }
18       { slides }
19     }
20   \str_if_eq:VnTF \l__talk_frame_tagging_str { n }
21   { \str_set:NV \l__talk_frame_tagging_str \l__talk_tmp_tl }
22   {
23     \str_replace_all:NnV \l__talk_frame_tagging_str { ,n }
24     \l__talk_tmp_tl
25     \str_replace_all:NnV \l__talk_frame_tagging_str { ,~n }
26     \l__talk_tmp_tl
27   }
28   \int_gzero:N \g__talk_slide_int
29   \RenewCommandCopy \frame \__talk_latex_frame:n
30   \bool_do_while:Nn \g__talk_slide_continue_bool
31   { \__talk_slide_auxi:nn {#1} {#2} }
32   \group_end:
33 }
34 \cs_new_protected:Npn \__talk_slide_auxi:nn #1#2
35 {
36   \int_gincr:N \g__talk_slide_int
37   \bool_gset_false:N \g__talk_slide_continue_bool
38   \__talk_if_overlay:nT {#1}
39   { \__talk_slide_auxii:n {#2} }
40 }

```

The business end of the slide starts in vertical mode, and opens the surrounding vertical box. Immediately inside is a horizontal box which will contain any tagging-related what-sits. Once that is done, back to vertical mode for the user content. The use of `:Nw` rather than `:n` boxes here is to allow the the split of the functions into “before” and “after” the content is typeset.

```

41 \cs_new_protected:Npn \__talk_slide_auxii:n #1
42 {
43   \__talk_slide_init:
44   \vbox_set:Nw \l__talk_slide_box
45   \hbox_set:Nw \l__talk_slide_box
46   \bool_if:NTF \g__talk_frame_tag_bool
47   { \__talk_slide_tag_on_begin: }
48   { \__talk_slide_tag_off_begin: }
49   \vbox_set:Nw \l__talk_slide_box
50   \bool_if:NTF \l__talk_frame_verb_bool
51   { \__talk_slide_auxiv:n }
52   { \use:n }

```

```

53         {#1}
54         \tl_use:N \g__talk_onslide_tl
55     \vbox_set_end:
56     \bool_lazy_and:nnTF
57     { \l__talk_frame_break_bool }
58     {
59         \dim_compare_p:nNn { \box_ht:N \l__talk_slide_box }
60         > \textheight
61     }
62     { \__talk_slide_break: }
63     { \__talk_slide_auxiii: }
64 }
65 \cs_new_protected:Npn \__talk_slide_auxiii:
66 {
67     \box_use_drop:N \l__talk_slide_box
68     \par
69     \bool_if:NTF \g__talk_frame_tag_bool
70     { \__talk_slide_tag_on_end: }
71     { \__talk_slide_tag_off_end: }
72     \hbox_set_end:
73     \box_use_drop:N \l__talk_slide_box
74     \vbox_set_end:
75     \bool_if:NT \g__talk_slide_continue_bool
76     { \__talk_cnt_restore: }
77     \__talk_slide_output:
78 }
79 \cs_new_protected:Npn \__talk_slide_auxiv:n #1
80 {
81     \group_begin:
82     \cs_set:Npn \obeyedline { ^^J }
83     \use:e
84     {
85         \group_end:
86         \tl_retokenize:n {#1}
87     }
88 }

```

(End of definition for __talk_slide:nn and others.)

`\__talk_slide_init:` Collects up some key initialization in one place; we set up a single tagging boolean here, which needs some shuffling of the global overlay state.

```

89 \cs_new_protected:Npn \__talk_slide_init:
90 {
91     \tl_gclear:N \g__talk_onslide_tl
92     \int_gzero:N \g__talk_pauses_int
93     \tl_gclear:N \g__talk_frame_title_tl
94     \tl_gclear:N \g__talk_frame_subtitle_tl
95     \box_gclear:N \g__talk_footnote_box
96     \__talk_cnt_save:
97     \bool_set_eq:NN \l__talk_slide_continue_bool
98     \g__talk_slide_continue_bool
99     \__talk_if_overlay:VTF \l__talk_frame_tagging_str
100     { \bool_gset_true:N \g__talk_frame_tag_bool }
101     { \bool_gset_false:N \g__talk_frame_tag_bool }

```

```

102     \bool_gset_eq:NN \g__talk_slide_continue_bool
103     \l__talk_slide_continue_bool
104 }

```

(End of definition for __talk_slide_init:.)

__talk_slide_break: Breaking a slide requires we get the content out of the multiple layers required for tagging. That means first saving the inner box then splitting before closing up and finalising the first slide as normal. We then need a simple loop to chop up the rest of the content: we keep the box nesting the same for each slide in the output.

```

105 \cs_new_protected:Npn \__talk_slide_break:
106 {
107     \box_set_eq:NN \l__talk_break_box \l__talk_slide_box
108     \vbox_set_split_to_ht:NNn \l__talk_slide_box \l__talk_break_box
109     { \fp_to_dim:n { \l__talk_frame_break_fp * \textheight } }
110     \box_gset_eq:NN \g__talk_break_box \l__talk_break_box
111     \__talk_slide_auxiii:
112     \box_set_eq:NN \l__talk_break_box \g__talk_break_box
113     \bool_do_until:nn { \box_if_empty_p:N \l__talk_break_box }
114     {
115         \vbox_set_split_to_ht:NNn \l__talk_slide_box \l__talk_break_box
116         { \fp_to_dim:n { \l__talk_frame_break_fp * \textheight } }
117         \vbox_set:Nn \l__talk_slide_box
118         { \hbox:n { \box_use_drop:N \l__talk_slide_box } }
119         \__talk_slide_output:
120     }
121 }

```

(End of definition for __talk_slide_break:.)

__talk_slide_output: Send the slide content to the main vertical list, after adding any footnotes.

```

122 \cs_new_protected:Npn \__talk_slide_output:
123 {
124     \thispdfpagelabel { \int_use:N \g__talk_frame_int }
125     \vbox_to_ht:nn { \textheight }
126     {
127         \use:c { __talk_slide_align_ \l__talk_frame_alignment_tl :n }
128         { \vbox_unpack_drop:N \l__talk_slide_box }
129         \box_if_empty:NF \g__talk_footnote_box
130         {
131             \footnoterule
132             \vbox_unpack_drop:N \g__talk_footnote_box
133         }
134     }
135     \bool_if:NF \g__talk_slide_continue_bool
136     {
137         \clist_set_eq:NN \l__talk_tmp_clist \l__talk_frame_properties_clist
138         \clist_put_left:Nn \l__talk_tmp_clist { slides }
139         \clist_remove_duplicates:N \l__talk_tmp_clist
140         \property_record:ee { frame . \int_use:N \g__talk_frame_int }
141         { \l__talk_tmp_clist }
142     }
143     \clearpage
144 }

```

(End of definition for `\__talk_slide_output:`.)

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

```

\__talk_slide_align_center:n
  \__talk_slide_align_stretch:n
    \__talk_slide_align_top:n
145 \cs_new_protected:Npn \__talk_slide_align_bottom:n #1
146 {
147   \skip_vertical:n { Opt~plus~1fil }
148   #1
149   \skip_vertical:n { Opt }
150 }
151 \cs_new_protected:Npn \__talk_slide_align_center:n #1
152 {
153   \skip_vertical:n { Opt~plus~0.5fil }
154   #1
155   \skip_vertical:n { Opt~plus~0.5fil }
156 }
157 \cs_new_protected:Npn \__talk_slide_align_stretch:n #1
158 {
159   \skip_vertical:n { Opt }
160   #1
161   \skip_vertical:n { Opt }
162 }
163 \cs_new_protected:Npn \__talk_slide_align_top:n #1
164 {
165   \skip_vertical:n { Opt }
166   #1
167   \skip_vertical:n { Opt~plus~1fil }
168 }

```

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

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

```

\__talk_slide_tag_on_end:
169 \cs_new_protected:Npn \__talk_slide_tag_on_begin:
170 {
171   \tag_struct_begin:n { tag = frame }
172   \int_gset:Nn \g__talk_frame_struct_int { \tag_get:n { struct_num } }
173 }
174 \cs_new_protected:Npn \__talk_slide_tag_on_end:
175 {
176   \tag_struct_end:
177 }

```

(End of definition for `\__talk_slide_tag_on_begin:` and `\__talk_slide_tag_on_end:`.)

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

```

\__talk_slide_tag_off_end:
178 \cs_new_protected:Npn \__talk_slide_tag_off_begin:
179 {
180   \tag_mc_begin:n { artifact }
181   \tag_suspend:n { frame }
182 }
183 \cs_new_protected:Npn \__talk_slide_tag_off_end:
184 {
185   \tag_resume:n { frame }

```

```

186     \tag_mc_end:
187 }

```

(End of definition for _talk_slide_tag_off_begin: and _talk_slide_tag_off_end:.)

\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).

```

188 \int_new:N \g\_talk_frame_struct_int

```

(End of definition for \g_talk_frame_struct_int.)

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.

```

189 \seq_new:N \g\_talk_cnt_reset_seq
190 \seq_gset_from_clist:Nn \g\_talk_cnt_reset_seq
191 {
192     equation      ,
193     footnote      ,
194     mpfootnote    ,
195     parentequation
196 }
197 \seq_map_inline:Nn \g\_talk_cnt_reset_seq
198 {
199     \int_new:c { g\_talk_saved_ #1 _int }
200     \int_gset_eq:cc { g\_talk_saved_ #1 _int } { c@ #1 }
201 }

```

(End of definition for \g_talk_cnt_reset_seq.)

_talk_cnt_save: A simple save-and-restore pair.

```

\_talk_cnt_restore:
202 \cs_new_protected:Npn \_talk_cnt_save:
203 {
204     \seq_map_inline:Nn \g\_talk_cnt_reset_seq
205     { \int_gset_eq:cc { g\_talk_saved_ ##1 _int } { c@ ##1 } }
206 }
207 \cs_new_protected:Npn \_talk_cnt_restore:
208 {
209     \seq_map_inline:Nn \g\_talk_cnt_reset_seq
210     { \int_gset_eq:cc { c@ ##1 } { g\_talk_saved_ ##1 _int } }
211 }

```

(End of definition for _talk_cnt_save: and _talk_cnt_restore:.)

\@definecounter Track all counters for resetting.

```

\std@definecounter
212 \cs_new_eq:NN \std@definecounter \@definecounter
213 \cs_gset_protected:Npn \@definecounter #1
214 {
215     \std@definecounter {#1}
216     \int_new:c { g\_talk_saved_ #1 _int }
217     \seq_gput_right:Nn \g\_talk_cnt_reset_seq {#1}
218 }

```

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

1.3 Frame options

\l__talk_frame_alignment_tl

219 \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_break_bool

\l__talk_frame_break_fp

\l__talk_frame_name_str

\l__talk_frame_suppl_bool

\l__talk_frame_tagging_str

220 \bool_new:N \l__talk_frame_suppl_bool

221 \keys_define:nn { talk / frame }

222 {

223 action-spec .str_set:N =

224 \l__talk_action_spec_str ,

225 auto-break .bool_set:N =

226 \l__talk_frame_break_bool ,

227 auto-break-coverage .fp_set:N =

228 \l__talk_frame_break_fp ,

229 label .meta:n =

230 { name = {#1} } ,

231 name .str_set:N =

232 \l__talk_frame_name_str ,

233 supplementary-frame .bool_set:N =

234 \l__talk_frame_suppl_bool ,

235 tag-slides .str_set:N =

236 \l__talk_frame_tagging_str ,

237 vertical-alignment .choices:nn =

238 { bottom , center , stretch , top }

239 {

240 \tl_set_eq:NN \l__talk_frame_alignment_tl

241 \l_keys_value_tl

242 }

243 }

244 \keys_set:nn { talk / frame }

245 {

246 action-spec = ,

247 auto-break = false ,

248 auto-break-coverage = 0.85 ,

249 name = ,

250 supplementary-frame = false ,

251 tag-slides = n ,

252 vertical-alignment = center

253 }

(End of definition for \l__talk_action_spec_str and others.)

1.4 Tagging for headers

__talk_header_tag_begin:n

__talk_header_tag_begin:e

__talk_header_tag_end:

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

254 \cs_new_protected:Npn __talk_header_tag_begin:n #1

255 {

```

256     \tag_resume:n { header }
257     \tag_mc_end:
258     \tag_struct_begin:n {#1}
259     \tag_mc_begin:n { }
260   }
261 \cs_generate_variant:Nn \__talk_header_tag_begin:n { e }
262 \cs_new_protected:Npn \__talk_header_tag_end:
263 {
264     \tag_mc_end:
265     \tag_struct_end:
266     \tag_mc_begin:n { artifact }
267     \tag_suspend:n { header }
268 }

```

(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
269 \NewTemplateType { footer-element } { 1 }
270 \DeclareTemplateInterface { footer-element } { talk } { 1 }
271 {
272     color      : tokenlist ,
273     font       : tokenlist = ,
274     left-hspace : length = 0em ,
275     right-hspace : length = 0em
276 }
277 \DeclareTemplateCode { footer-element } { talk } { 1 }
278 {
279     color      = \l__talk_footelem_color_tl ,
280     font       = \l__talk_footelem_font_tl ,
281     left-hspace = \l__talk_footelem_left_skip ,
282     right-hspace = \l__talk_footelem_right_skip
283 }
284 {
285     \tl_if_empty:nF {#1}
286     {
287         \hspace { \l__talk_footelem_left_skip }
288         \hbox:n
289         {
290             \tl_if_empty:NF \l__talk_footelem_color_tl
291             { \color_select:V \l__talk_footelem_color_tl }
292             \l__talk_footelem_font_tl
293             #1
294         }
295         \hspace { \l__talk_footelem_right_skip }
296     }
297 }
298 \DeclareInstance { footer-element } { date } { talk } { }
299 \DeclareInstance { footer-element } { author } { talk } { }
300 \DeclareInstance { footer-element } { title } { talk } { }
301 \DeclareInstance { footer-element } { subtitle } { talk } { }
302 \DeclareInstance { footer-element } { institute } { talk } { }
303 \DeclareInstance { footer-element } { framenumbers } { talk } { }

```

```
304 \DeclareInstance { footer-element } { totalframes } { talk } { }
```

(End of definition for \l__talk_footelem_left_skip and others.)

```

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

```

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

```

305 \NewTemplateType { header } { 0 }
306 \DeclareTemplateInterface { header } { talk } { 0 }
307 {
308   background-color : tokenlist,
309   color             : tokenlist = structure ,
310   font              : tokenlist = \normalfont ,
311   height            : length = \Gm@tmargin + \headsep ,
312   left-hspace       : skip = \Gm@lmargin ,
313   print-frame-title : boolean = true ,
314   right-hspace      : skip = \Gm@rmargin
315 }
316 \DeclareTemplateCode { header } { talk } { 0 }
317 {
318   background-color = \l__talk_header_bg_tl ,
319   color            = \l__talk_header_fg_tl ,
320   font             = \l__talk_header_font_tl ,
321   height           = \l__talk_header_ht_dim ,
322   left-hspace      = \l__talk_header_left_skip ,
323   print-frame-title = \l__talk_header_frametitle_bool ,
324   right-hspace     = \l__talk_header_right_skip
325 }
326 {
327   \noindent
328   \__talk_wallpaper_hrulerule:Nnn
329     \l__talk_header_bg_tl
330     { \l__talk_header_ht_dim - \headsep }
331     { \headsep }
332   \skip_horizontal:n { \l__talk_header_left_skip }
333   \group_begin:
334     \tl_if_empty:NF \l__talk_header_fg_tl
335     { \color_select:V \l__talk_header_fg_tl }
336     \l__talk_header_font_tl
337     \bool_if:NT \l__talk_header_frametitle_bool
338     {
339       \ExpandArgs { nnV }
340       \UseInstance { frametitle } { header }
341       \g__talk_frame_title_tl
342     }
343   \group_end:
344 }
345 \DeclareInstance { header } { std } { talk } { }
346 \AddToHook { begindocument }
347 {
348   \DeclareInstanceCopy { header } { wallpaper } { std }
349   \EditInstance { header } { wallpaper }
350     { print-frame-title = false }

```

351 }

(End of definition for \l__talk_header_bg_tl and others.)

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

```

\l__talk_footer_bg_tl
\l__talk_footer_fg_tl
\l__talk_footer_font_tl
\l__talk_footer_order_clist
\l__talk_footer_sep_tl
\l__talk_footer_left_skip
\l__talk_footer_right_skip
352 \NewTemplateType { footer } { 0 }
353 \DeclareTemplateInterface { footer } { talk } { 0 }
354 {
355     background-color : tokenlist ,
356     color             : tokenlist ,
357     element-order    : commalist ,
358     font              : tokenlist = \tiny ,
359     left-hspace       : length = \Gm@lmargin ,
360     right-hspace      : length = \Gm@rmargin ,
361     separator         : tokenlist = \hfil
362 }
363 \DeclareTemplateCode { footer } { talk } { 0 }
364 {
365     background-color = \l__talk_footer_bg_tl ,
366     color            = \l__talk_footer_fg_tl ,
367     element-order    = \l__talk_footer_order_clist ,
368     font             = \l__talk_footer_font_tl ,
369     left-hspace      = \l__talk_footer_left_skip ,
370     right-hspace     = \l__talk_footer_right_skip ,
371     separator        = \l__talk_footer_sep_tl
372 }
373 {
374     \noindent
375     \__talk_wallpaper_hrulerule:Nnn
376     \l__talk_footer_bg_tl
377     { \footskip }
378     { \Gm@bmargin - \footskip }
379     \skip_horizontal:n { \l__talk_footer_left_skip }
380     \vbox_set_to_wd:Nnn \l__talk_tmp_box
381     {
382         \paperwidth
383         - \l__talk_footer_left_skip
384         - \l__talk_footer_right_skip
385     }
386     {
387         \tl_if_empty:NF \l__talk_footer_fg_tl
388         { \color_select:V \l__talk_footer_fg_tl }
389         \l__talk_footer_font_tl
390         \clist_pop:NNT \l__talk_footer_order_clist \l__talk_tmp_tl
391         {
392             \ExpandArgs { nVv } \UseInstance { footer-element } \l__talk_tmp_tl
393             { @ \__talk_metadata_name:n { \l__talk_tmp_tl } }
394             \clist_map_inline:Nn \l__talk_footer_order_clist
395             {
396                 \tl_if_empty:cF { @ \__talk_metadata_name:n { ##1 } }
397                 {
398                     \l__talk_footer_sep_tl

```

```

399         \ExpandArgs { nnn }
400         \UseInstance { footer-element } {##1}
401         { @ \_talk\_metadata\_name:n { ##1 } }
402     }
403 }
404 }
405 \hfil
406 }
407 \box\_use\_drop:N \l\_talk\_tmp\_box
408 \skip\_horizontal:n { \l\_talk\_footer\_right\_skip - \Gm@rmargin }
409 }
410 \DeclareInstance { footer } { std } { talk } { }
411 \AddToHook { begindocument }
412 {
413     \DeclareInstanceCopy { footer } { wallpaper } { std }
414     \EditInstance { footer } { wallpaper }
415     { element-order = }
416 }

```

(End of definition for \l_talk_footer_bg_tl and others.)

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

```

417 \cs\_new:Npn \_talk\_metadata\_name:n #1
418 {
419     \tl\_if\_exist:cTF { @ short #1 }
420     { short #1 }
421     {#1}
422 }

```

(End of definition for _talk_metadata_name:n.)

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

```

423 \cs\_new\_protected:Npn \_talk\_wallpaper\_hrule:Nnn #1#2#3
424 {
425     \skip\_horizontal:n { -\Gm@lmargin }
426     \tl\_if\_empty:NF #1
427     {
428         \group\_begin:
429         \color\_select:V #1
430         \rule:nnn { \paperwidth } {#2} {#3}
431         \skip\_horizontal:n { -\paperwidth }
432         \group\_end:
433     }
434 }

```

(End of definition for _talk_wallpaper_hrule:Nnn.)

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

```

435 \cs\_set\_noper:Npn \ps@plain
436 {

```

```

437 \cs_set_nopar:Npn \@oddhead
438 {
439   \hfil
440 }
441 \cs_set_nopar:Npn \@oddfoot { }
442 \cs_set_eq:NN \@evenhead \@oddhead
443 \cs_set_eq:NN \@evenfoot \@oddfoot
444 }
445 \cs_set_nopar:Npn \ps@wallpaper
446 {
447   \cs_set_nopar:Npn \@oddhead
448   {
449     \UseInstance { header } { wallpaper }
450     \hfil
451   }
452   \cs_set_nopar:Npn \@oddfoot
453   {
454     \UseInstance { footer } { wallpaper }
455     \hfil
456   }
457   \cs_set_eq:NN \@evenhead \@oddhead
458   \cs_set_eq:NN \@evenfoot \@oddfoot
459 }
460 \cs_new_nopar:Npn \ps@talk
461 {
462   \cs_set_nopar:Npn \@oddhead
463   {
464     \UseInstance { header } { std }
465     \hfil
466   }
467   \cs_set_nopar:Npn \@oddfoot { \UseInstance { footer } { std } }
468   \cs_set_eq:NN \@evenhead \@oddhead
469   \cs_set_eq:NN \@evenfoot \@oddfoot
470 }
471 \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.

```
472 \bool_new:N \l__talk_frame_bool
```

(End of definition for \l__talk_frame_bool.)

`\g__talk_frame_suppl_int` Used to track saved supplementary frames.

```
473 \int_new:N \g__talk_frame_suppl_int
```

(End of definition for \g__talk_frame_suppl_int.)

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

```
474 \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).

475 `\bool_new:N \l__talk_frame_verb_bool`

(End of definition for `\l__talk_frame_verb_bool`.)

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

`\c@frame` 476 `\int_new:N \g__talk_frame_int`

`\theframe` 477 `\cs_new_eq:NN \c@frame \g__talk_frame_int`

`\@framenumber` 478 `\cs_new:Npn \theframe { \@arabic \c@frame }`

479 `\cs_new:Npn \@framenumber { \arabic { frame } }`

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

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

`\c@appendixframe` 480 `\int_new:N \g__talk_appendixframe_int`

`\theappendixframe` 481 `\cs_new_eq:NN \c@appendixframe \g__talk_appendixframe_int`

`\@appendixframenumber` 482 `\cs_new:Npn \theappendixframe { \@arabic \c@appendixframe }`

483 `\cs_new:Npn \@appendixframenumber { \arabic { appendixframe } }`

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

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

484 `\property_new:nnnn { frames } { shipout } { -1 }`

485 `{ \int_eval:n { \g__talk_frame_int - \g__talk_appendixframe_int } }`

486 `\cs_new:Npn \@frames { \property_ref:nn { lastpage } { frames } }`

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

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

487 `\property_new:nnnn { totalframes } { shipout } { -1 }`

488 `{ \int_use:N \g__talk_frame_int }`

489 `\cs_new:Npn \@totalframes { \property_ref:nn { lastpage } { totalframes } }`

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

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

490 `\property_new:nnnn { appendixframes } { shipout } { -1 }`

491 `{ \int_use:N \g__talk_appendixframe_int }`

492 `\cs_new:Npn \@appendixframes { \property_ref:nn { lastpage } { appendixframes } }`

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

`\l_talk_frame_properties_clist` Currently somewhat experimental: not used by us, but helpful for users.

493 `\clist_new:N \l_talk_frame_properties_clist`

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

494 `\AddToHook { enddocument / afterlastpage }`

495 `{`

496 `\property_record:nn { lastpage }`

497 `{ appendixframes , frames , totalframes }`

498 `}`

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

```
499 \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. As frames can be saved either by the user or automatically that is handled in an auxiliary.

```
\__talk_frame_process_aux:nn
  \__talk_frame_save:nn
  \__talk_frame_save:en
500 \cs_new_protected:Npn \__talk_frame_process:nn #1#2
501 {
502   \__talk_decode_parse:n {#1}
503   \bool_if:NT \l__talk_decode_mode_bool
504     { \__talk_frame_process_aux:nn {#1} {#2} }
505 }
506 \cs_new_protected:Npn \__talk_frame_process_aux:nn #1#2
507 {
508   \int_gincr:N \g__talk_frame_int
509   \bool_if:NT \l__talk_appendix_bool
510     { \int_gincr:N \g__talk_appendixframe_int }
511   \bool_set_true:N \l__talk_frame_bool
512   \str_if_empty:NF \l__talk_frame_name_str
513     { \__talk_frame_save:en { \l__talk_frame_name_str } {#2} }
514   \bool_if:NTF \l__talk_frame_suppl_bool
515     {
516       \str_case:Vn \l__talk_suppl_mode_str
517       {
518         { appendix }
519         {
520           \int_gincr:N \g__talk_frame_suppl_int
521           \__talk_frame_save:en
522             { appendix . \int_use:N \g__talk_frame_suppl_int }
523             {#2}
524         }
525         { in-place } { \__talk_slide:nn {#1} {#2} }
526       }
527     }
528     { \__talk_slide:nn {#1} {#2} }
529 }
530 \cs_new_protected:Npn \__talk_frame_save:nn #1#2
531 {
532   \tl_if_exist:cT { g__talk_frame_ #1 _tl }
533     { \msg_error:nnn { talk } { duplicate-frame-name } {#1} }
534   \tl_clear_new:c { g__talk_frame_ #1 _tl }
535   \tl_gset:ce { g__talk_frame_ #1 _tl }
536   {
537     {
538       \bool_if:NTF \l__talk_frame_verb_bool
539         { true }
540         { false }
541     }
542     { \exp_not:n {#2} }
543   }
544 }
545 \cs_generate_variant:Nn \__talk_frame_save:nn { e }
```

(End of definition for `\_talk\_frame\_process:nn`, `\_talk\_frame\_process\_aux:nn`, and `\_talk\_frame\_save:nn`.)

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

```

546 \bool_if:NTF \l__talk_frame_title_bool
547 {
548   \RenewDocumentEnvironment { frame }
549     { D <> { all } = { action-spec } 0 { } +m +b }
550     {
551       \str_clear:N \l__talk_frame_name_str
552       \keys_set:nn { talk / frame } {#2}
553       \bool_set_false:N \l__talk_frame_verb_bool
554       \__talk_frame_process:nn {#1} { \frametitle {#3} #4 }
555     }
556   { }
557   \NewDocumentEnvironment { frame* }
558     { D <> { all } = { action-spec } 0 { } +m c }
559     {
560       \str_clear:N \l__talk_frame_name_str
561       \keys_set:nn { talk / frame } {#2}
562       \bool_set_true:N \l__talk_frame_verb_bool
563       \tl_gset:Nn \g__talk_frame_title_tl {#3}
564       \exp_args:Nne \__talk_frame_process:nn {#1}
565         { \tl_to_str:n { \frametitle } \exp_not:n { {#3} #4 } }
566     }
567   { }
568 }
569 {
570   \RenewDocumentEnvironment { frame }
571     { !D <> { all } = { action-spec } !0 { } +b }
572     {
573       \str_clear:N \l__talk_frame_name_str
574       \keys_set:nn { talk / frame } {#2}
575       \bool_set_false:N \l__talk_frame_verb_bool
576       \__talk_frame_process:nn {#1} {#3}
577     }
578   { }
579   \NewDocumentEnvironment { frame* }
580     { !D <> { all } = { action-spec } !0 { } c }
581     {
582       \str_clear:N \l__talk_frame_name_str
583       \keys_set:nn { talk / frame } {#2}
584       \bool_set_true:N \l__talk_frame_verb_bool
585       \__talk_frame_process:nn {#1} {#3}
586     }
587   { }
588 }

```

(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` 589 `\cs_new_protected:Npn \_talk\_frame\_reuse:nn #1#2`
`\reuseframe`
`\againframe`

```

590 {
591   \tl_if_exist:cTF { g__talk_frame_ #2 _tl }
592   {
593     \exp_args:Nv \__talk_frame_reuse_aux:nn
594     { g__talk_frame_ #2 _tl } {#1}
595   }
596   { \msg_error:nnn { talk } { unknown-frame-name } {#2} }
597 }
598 \cs_new_protected:Npn \__talk_frame_reuse_aux:nn #1#2
599 { \__talk_frame_reuse_aux:nnn #1 {#2} }
600 \cs_new_protected:Npn \__talk_frame_reuse_aux:nnn #1#2#3
601 {
602   \use:c { bool_set_ #1 :N } \l__talk_frame_verb_bool
603   \__talk_frame_process:nn {#3} {#2}
604 }
605 \NewDocumentCommand \reuseframe { D <> { all } = { action-spec } O { } m }
606 {
607   \group_begin:
608   \keys_set:nn { talk / frame } {#2}
609   \str_clear:N \l__talk_frame_name_str
610   \__talk_frame_reuse:nn {#1} {#3}
611   \group_end:
612 }
613 \NewCommandCopy \againframe \reuseframe

```

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

Supplementary frames deferred to the appendix need to be swept up. If necessary, the appendix is triggered.

```

614 \AddToHook { enddocument }
615 {
616   \int_compare:nNnT \g__talk_frame_suppl_int > 0
617   {
618     \bool_if:NF \l__talk_appendix_bool
619     { \appendix }
620     \int_step_inline:nn \g__talk_frame_suppl_int
621     { \__talk_frame_reuse:nn { } { appendix . #1 } }
622   }
623 }
624 \msg_new:nnnn { talk } { unknown-frame-name }
625 { Unknown~frame~name~"#1"! }
626 {
627   You~asked~to~reuse~the~contents~of~a~frame~with~the~name~"#1",~
628   but~that~name~was~never~defined.
629 }
630 \msg_new:nnnn { talk } { duplicate-frame-name }
631 { Duplicate~frame~name~"#1"! }
632 {
633   You~asked~to~save~the~contents~of~this~frame~with~the~name~"#1",~
634   but~that~name~has~already~been~used.
635 }
636 </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 whatsits 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
\_talk_action_only_end:N
\_talk_saved_ifendpe:

```

Here, we simply throw away the content we do not want: this is done by typesetting in a disposable box. As box setting rather than insertion creates tag structures, they may need to be suspended. To avoid issues with whatsits, we have a surrounding horizontal box if required. Suspending tagging does not turn off or save/restore the @endpe mechanism, so we need to do this ourselves. This avoids issues with paragraph tagging if the mechanism would otherwise be active.

```

64 \cs_new_protected:Npn \_talk_action_only:N #1
65   {
66     \bool_if:NF #1
67     {
68       \bool_if:NT \g__talk_frame_tag_bool
69       {
70         \cs_set_eq:NN \_talk_saved_ifendpe: \if@endpe
71         \vbox_set:Nw \l__talk_tmp_box
72         \hbox_set:Nw \l__talk_tmp_box
73         \tag_suspend:n { onlyenv }
74       }
75       \vbox_set:Nw \l__talk_tmp_box
76     }
77   }
78 \cs_new_protected:Npn \_talk_action_only_end:N #1
79   {
80     \bool_if:NF #1
81     {
82       \vbox_set_end:
83       \bool_if:NT \g__talk_frame_tag_bool
84       {
85         \tag_resume:n { onlyenv }
86         \hbox_set_end:
87         \vbox_set_end:
88         \cs_gset_eq:NN \if@endpe \_talk_saved_ifendpe:
89       }
90     }
91   }
92 \cs_new_eq:NN \_talk_saved_ifendpe: \if@endpe

```

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

```
\l__talk_uncover_hidden_fp
```

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

```

93 \NewTemplateType { hidden } { 0 }
94 \DeclareTemplateInterface { hidden } { talk } { 0 }
95   { opacity : real = 0 }
96 \DeclareTemplateCode { hidden } { talk } { 0 }
97   { opacity = \l__talk_uncover_hidden_fp }
98   { \_talk_opacity_begin:n { \l__talk_uncover_hidden_fp } }
99 \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.

```

100 \cs_new_protected:Npn \__talk_action_uncover:N #1
101   {
102     \bool_if:NTF #1
103       { \__talk_opacity_begin:n { 1 } }
104       { \UseInstance { hidden } { std } }
105   }
106 \cs_new_protected:Npn \__talk_action_uncover_end:N #1
107   { \__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 108 \clist_map_inline:nn { invisible , uncover , visible }
\visible 109   {
110     \ExpandArgs { cne } \NewDocumentCommand { #1 }
111       { > { \__talk_overlay_arg:n } D <> { all } +m }
112       {
113         \exp_not:c { __talk_action_ #1 :N } ##1
114         ##2
115         \exp_not:c { __talk_action_ #1 _end:N } ##1
116       }

```

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

invisibleenv (env.) And the environment versions.

```

117 \uncoverenv (env.) \ExpandArgs { nnee } \NewDocumentEnvironment { #1 env }
118 \visibleenv (env.) { > { \__talk_overlay_arg:n } D <> { all } }
119 { \exp_not:c { __talk_action_ #1 :N } ##1 }
120 { \exp_not:c { __talk_action_ #1 _end:N } ##1 }
121 }

```

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

```

122 \NewDocumentCommand \alert { > { \__talk_overlay_arg:n } D <> { all } +m }
123   {
124     \group_begin:
125       \__talk_action_alert:N #1
126       #2
127     \group_end:
128   }

```

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

alertenv (env.) As does the environment.

```

129 \NewDocumentEnvironment { alertenv } { > { \__talk_overlay_arg:n } D <> { all } }
130   { \__talk_action_alert:N #1 }
131   { }

```

`\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).

```

132 \NewDocumentCommand \only { D <> { all } +m }
133 {
134   \__talk_if_overlay:nT {#1}
135   {#2}
136 }

```

(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.

```

137 \NewDocumentEnvironment { onlyenv } { > { \__talk_overlay_arg:n } D <> { all } }
138 { \__talk_action_only:N #1 }
139 { \__talk_action_only_end:N #1 }

```

```

\l__talk_saved_overlays_bool
\l__talk_saved_action_str
\l__talk_saved_actions_bool
140 \bool_new:N \l__talk_saved_overlays_bool
141 \str_new:N \l__talk_saved_action_str
142 \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
143 \bool_new:N \l__talk_overlay_all_bool

```

(End of definition for `\l__talk_overlay_all_bool`.)

`actionenv (action)` 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:
144 \NewDocumentCommand \action { d <> +m }
145 {
146   \group_begin:
147   \__talk_action_begin:n {#1}
148   #2
149   \__talk_action_end:
150   \group_end:
151 }
152 \NewDocumentEnvironment { actionenv } { d <> }
153 { \__talk_action_begin:n {#1} }
154 { \__talk_action_end: }
155 \cs_new_protected:Npn \__talk_action_begin:n #1
156 {
157   \group_begin:
158   \tl_if_novalue:nTF {#1}
159   {
160     \exp_after:wN \__talk_action_begin:w

```

```

161         \l__talk_action_spec_str < all > \q_stop
162     }
163     { \__talk_action_begin_auxi:n {#1} }
164 }
165 \cs_new_protected:Npn \__talk_action_begin:w #1 < #2 > #3 \q_stop
166 { \__talk_action_begin_auxi:n {#2} }
167 \cs_new_protected:Npn \__talk_action_begin_auxi:n #1
168 {
169     \str_if_eq:nnTF {#1} { all }
170     { \bool_set_true:N \l__talk_overlay_all_bool }
171     {
172         \bool_set_false:N \l__talk_overlay_all_bool
173         \__talk_action_begin_auxii:n {#1}
174     }
175 }
176 \cs_new_protected:Npn \__talk_action_begin_auxii:n #1
177 {
178     \__talk_decode_parse:n {#1}
179     \bool_set_eq:NN \l__talk_saved_overlays_bool
180     \l__talk_decode_overlays_bool
181     \str_set_eq:NN \l__talk_saved_action_str
182     \l__talk_decode_action_str
183     \bool_set_eq:NN \l__talk_saved_actions_bool
184     \l__talk_decode_actions_bool
185     \tl_if_empty:NTF \g__talk_onslide_tl
186     {
187         \bool_if:NTF \l__talk_decode_overlays_bool
188         {
189             \cs_if_exist_use:cF
190             { __talk_action_ \l__talk_decode_action_str :N }
191             { \use_none:n }
192             \l__talk_decode_actions_bool
193         }
194         { \UseInstance { hidden } { std } }
195     }
196     { \__talk_action_invisible:N \c_true_bool }
197 }
198 \cs_new_protected:Npn \__talk_action_end:
199 {
200     \bool_if:NF \l__talk_overlay_all_bool
201     {
202         \tl_if_empty:NTF \g__talk_onslide_tl
203         {
204             \bool_if:NTF \l__talk_saved_overlays_bool
205             {
206                 \cs_if_exist_use:cF
207                 { __talk_action_ \l__talk_saved_action_str _end:N }
208                 { \use_none:n }
209                 \l__talk_saved_actions_bool
210             }
211             { \__talk_opacity_end: }
212         }
213         { \__talk_action_invisible_end:N \c_true_bool }
214     }

```

```

215 \group_end:
216 }

```

(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.

```

217 \NewDocumentCommand \alt { D <> { all } +m +m }
218 {
219   \__talk_if_overlay:nTF {#1}
220     {#2}
221     {#3}
222 }

```

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

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

```

\__talk_onslide:n
223 \NewDocumentCommand \onslide { D <> { all } }
224 {
225   \__talk_onslide:n {#1}
226   \ignorespaces
227 }
228 \cs_new_protected:Npn \__talk_onslide:n #1
229 {
230   \tl_use:N \g__talk_onslide_tl
231   \tl_gclear:N \g__talk_onslide_tl
232   \__talk_if_overlay:nF {#1}
233   {
234     \__talk_opacity_begin:n { 0 }
235     \tl_gput_right:Nn \g__talk_onslide_tl
236       { \__talk_opacity_end: }
237   }
238 }

```

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

\g__talk_onslide_tl

```

239 \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.

```

240 \NewDocumentCommand \temporal { D <> { all } +m +m +m }
241 {
242   \__talk_if_overlay:nTF {#1}
243     {#3}
244     {
245       \bool_if:NTF \l__talk_decode_trailing_bool
246         {#4}
247         {#2}

```

```

248     }
249 }

```

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

\pause A thin wrapper.

```

250 \NewDocumentCommand \pause { o }
251 {
252   \legacy_if:nF { measuring@ }
253   {
254     \IfNoValueTF {#1}
255     { \int_gincr:N \g__talk_pauses_int }
256     { \int_gset:Nn \g__talk_pauses_int {#1} }
257     \exp_args:Ne \__talk_onslide:n
258     { \int_eval:n { \g__talk_pauses_int + 1 } - }
259   }
260 }

```

(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**.

```

261 \cs_new_protected:Npn \__talk_overprint_begin:n #1
262 {
263   \par
264   \vbox_set_to_wd:Nnw \l__talk_tmp_box {#1}
265   \raggedright
266   \ignorespaces
267 }

```

(End of definition for __talk_overprint_begin:n.)

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

```

268 \NewDocumentEnvironment { overlayarea } { m m }
269 { \__talk_overprint_begin:n {#1} }
270 {
271   \vbox_set_end:
272   \vbox_to_ht:nn {#2}
273   {
274     \box_use_drop:N \l__talk_tmp_box
275     \vfil
276   }
277   \par
278 }

```

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

```

279 \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.

```
280 \cs_new:Npn \__talk_frame_overprint:
281 {
282   \int_to_Roman:n \g__talk_frame_int
283   \int_to_roman:n \l__talk_overprint_int
284 }
```

(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.

`\__talk_overprint_check_ht:n`

```
285 \NewDocumentEnvironment { overprint } { 0 { \textwidth } }
286 { \__talk_overprint_begin:n {#1} }
287 {
288   \vbox_set_end:
289   \int_incr:N \l__talk_overprint_int
290   \__talk_overprint_save_ht:
291   \cs_if_exist:cTF
292     { overprint@ \__talk_frame_overprint: }
293     {
294       \dim_compare:vNnTF
295         { overprint@ \__talk_frame_overprint: }
296         > { \box_ht:N \l__talk_tmp_box }
297         {
298           \vbox_to_ht:vn
299             { overprint@ \__talk_frame_overprint: }
300             {
301               \box_use_drop:N \l__talk_tmp_box
302               \vfil
303             }
304           }
305         { \box_use_drop:N \l__talk_tmp_box }
306       }
307     { \box_use_drop:N \l__talk_tmp_box }
308   \par
309 }
```

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.

```
310 \cs_new_protected:Npn \__talk_overprint_save_ht:
311 {
312   \tl_if_exist:cF { g__talk_overprint_ \__talk_frame_overprint: _tl }
313   {
314     \tl_new:c { g__talk_overprint_ \__talk_frame_overprint: _tl }
315     \tl_gset:cn { g__talk_overprint_ \__talk_frame_overprint: _tl }
316     { Opt }
317   }
318   \tl_gset:ce { g__talk_overprint_ \__talk_frame_overprint: _tl }
319   {
320     \dim_max:vn { g__talk_overprint_ \__talk_frame_overprint: _tl }
321     { \box_ht:N \l__talk_tmp_box }
322   }
323 }
```

```

323 \legacy_if:nT { @filesw }
324 {
325     \iow_now:Ne \@auxout
326     {
327         \gdef \exp_not:c { overprint@ \__talk_frame_overprint: }
328         {
329             \exp_not:v { g__talk_overprint_ \__talk_frame_overprint: _t1 }
330         }
331     }
332 }
333 \hook_gput_code:nne { enddocument / afterlastpage } { talk }
334 { \__talk_overprint_check_ht:n { \__talk_frame_overprint: } }
335 }
336 \cs_new_protected:Npn \__talk_overprint_check_ht:n #1
337 {
338     \bool_lazy_and:nnF
339     { \exp_not:N \cs_if_exist_p:c { overprint@ #1 } }
340     {
341         \dim_compare_p:vNv { overprint@ #1 } = { g__talk_overprint_ #1 _t1 }
342     }
343     {
344         \msg_warning:nn { talk } { overprint-ht }
345         \cs_gset_protected:Npn \__talk_overprint_check_ht:n ##1 { }
346     }
347 }
348 \msg_new:nnn { talk } { overprint-ht }
349 {
350     Overprint~area~height~has~changed:\\
351     rerun~LaTeX.
352 }

```

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

1.6 Adding overlays to existing commands

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

\textit

\textmd

\textnormal

353 \tl_map_inline:nn

\textrm

354 {

\textsc

355 \textbf

\textsf

356 \textit

\textsl

357 \textmd

\texttt

358 \textnormal

\textup

359 \textrm

\emph

360 \textsc

\stdtextbf

361 \textsf

\stdtextit

362 \textsl

\stdtextmd

363 \texttt

\stdtextnormal

364 \textup

\stdtextrm

365 \emph

\stdtextsc

366 }

\stdtextsf

367 {

\stdtextsl

368 \ExpandArgs { c } \NewCommandCopy { std \cs_to_str:N #1 } #1

\stdtexttt

\stdtextup

\stdemph

__talk_textcmd_eqiv:n

```

369 \ExpandArgs { Nne } \RenewDocumentCommand #1
370 { D <> { all } +m }
371 {
372   \exp_not:N \__talk_if_overlay:nTF {##1}
373   { \exp_not:c { std \cs_to_str:N #1 } }
374   { \exp_not:N \__talk_textcmd_equiv:n }
375   {##2}
376 }
377 }
378 \cs_new_protected:Npn \__talk_textcmd_equiv:n #1
379 {
380   \mode_if_math:TF
381   { { \mbox {#1} } }
382   {
383     \mode_leave_vertical:
384     {#1}
385   }
386 }

```

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

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

`\stdincludegraphics`

```

387 \RequirePackage { graphicx }
388 \NewCommandCopy \stdincludegraphics \includegraphics
389 \RenewDocumentCommand \includegraphics { s D <> { all } o o m }
390 {
391   \__talk_if_overlay:nT {#2}
392   {
393     \use:e
394     {
395       \exp_not:N \stdincludegraphics
396       \IfBooleanT #1 { * }
397       \IfNoValueF {#3} { [ \exp_not:n { {#3} } ] }
398       \IfNoValueF {#4} { [ \exp_not:n { {#4} } ] }
399     }
400     {#5}
401   }
402 }

```

(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`

```

403 \RenewDocumentCommand \label { D <> { 1 } m }
404 {
405   \@bsphack
406   \__talk_if_overlay:nT {#1}
407   { \__talk_label:n {#2} }
408   \@esphack
409 }
410 \cs_new_protected:Npn \__talk_label:n #1

```

```

411 {
412   \begingroup
413     \UseHookWithArguments { label } { 1 } {#1}
414     \protected@write \@auxout { }
415     {
416       \string \newlabel {#1}
417       {
418         { \@currentlabel }
419         { \thepage }
420         { \@currentlabelname }
421         { \@currentHref }
422         { \@kernel@reserved@label@data }
423       }
424     }
425   \endgroup
426 }

```

(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!

```

427 \cs_new_eq:NN \std@label@in@display \label@in@display
428 \RenewDocumentCommand \label@in@display { D <> { 1 } m }
429 {
430   \_talk_if_overlay:nT {#1}
431   { \std@label@in@display {#2} }
432 }

```

(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.

```

433 \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:

```

434 \cs_new_protected:Npn \_talk_opacity_group_begin:
435 { \begin { lrbox } \l__talk_tmp_box }
436 \cs_new_protected:Npn \_talk_opacity_group_end:
437 {
438   \end { lrbox }
439   \mode_leave_vertical:
440   \int_gincr:N \g__talk_opacity_group_int
441   \exp_args:Ne \pdfxform_new:nnn
442     { talk.opacity \int_use:N \g__talk_opacity_group_int }
443     { /Group << /S /Transparency /K ~ false /I ~ false >> }

```

```

444     { \usebox \l__talk_tmp_box }
445     \exp_args:Ne \pdfxform_use:n
446     { talk.opacity \int_use:N \g__talk_opacity_group_int }
447 }

```

(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 LuaT_EX due to restrictions on Xform structures in pdfT_EX. Here, we apply the collection code to the commands not the environment forms to deal with “direct” usage.

```

448 \sys_if_engine luatex:T
449 {
450     \AddToHook { cmd / pspicture / before } { \__talk_opacity_group_begin: }
451     \AddToHook { cmd / endpspicture / after } { \__talk_opacity_group_end: }
452 }
453 \</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 \end{class}

```

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>	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. <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>	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. <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`.)

`\appendixmarker` At present, the best way to deal with the TOC for the appendix is to set a marker: this
`\contentsline` will likely be generalized once we add support for parts.
`\_talk_saved_contentline:nnnn`

```

202 \NewDocumentCommand \appendixmarker { }
203 {
204   \bool_if:NTF \l__talk_appendix_bool
205     { \cs_gset_eq:NN \contentsline \_talk_saved_contentline:nnnn }
206     { \cs_gset_eq:NN \contentsline \use_none:nnnn }
207 }
208 \cs_new_eq:NN \_talk_saved_contentline:nnnn \use_none:nnnn
209 \cs_gset_eq:NN \_talk_saved_contentline:nnnn \contentsline

```

(End of definition for `\appendixmarker`, `\contentsline`, and `\_talk_saved_contentline:nnnn`. These functions are documented on page ??.)

`talk/sec/title` The argument is one of section, subsection or subsubsection.
`\_talk_head_tag:nn`

```

210 \NewTaggingSocket { talk / sec / title } { 1 }
211 \NewTaggingSocketPlug { talk / sec / title } { default }
212 {
213   \exp_args:Ne \_talk_head_tag:nn
214     { \text_purify:v { l__talk_ #1 _ t1 } } { #1 }
215 }
216 \cs_new_protected:Npn \_talk_head_tag:nn #1#2
217 {
218   \tag_struct_begin:e
219   {
220     tag      =
221       \UseStructureName { sec / \use:c { toplevel@ #2 } / title } ,
222     title    = { #1 } ,
223     actualtext = { #1 } ,
224   }
225   \tag_struct_end:
226 }
227 \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
`\@starttoc@aux__talknn` read. That's no good for us as the table of contents can be read multiple times. So we
 modify the code: we start from the tagging-aware version (this may need to be revisited).
 We retain the L^AT_EX 2_ε code as much as possible.

```

228 \cs_gset_protected:Npn \@starttoc #1
229 {
230   \begingroup
231   \makeatletter
232   \bool_if:NTF \g__talk_frame_tag_bool
233     { \@starttoc@aux@nn }
234     { \use_ii:nn }
235     { #1 } { \input { \jobname . #1 } }
236   \legacy_if:nT { @filesw }
237   {
238     \AddToHook { enddocument / afterlastpage }

```

```

239         {
240             \expandafter \newwrite \csname tf@ #1 \endcsname
241             \immediate \openout \csname tf@ #1 \endcsname \jobname .#1 \relax
242         }
243     }
244     \@nobreakfalse
245     \endgroup
246 }
247 \cs_new_protected:Npn \@starttoc@aux@nn #1#2
248 {
249     \UseTaggingSocket { toc / starttoc / before } {#1}
250     #2
251     \UseTaggingSocket { toc / starttoc / after } {#1}
252 }

```

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

\tableofcontents For the present simply print the output.

```

253 \NewDocumentCommand \tableofcontents { 0 { } }
254 {
255     \group_begin:
256     \@starttoc { toc }
257     \group_end:
258 }

```

(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 259 \cs_new_protected:Npn \l@section #1#2

__talk_toc_aux:nnnn 260 { __talk_toc_aux:nnnn { 1 } { \bfseries \color { structure } } {#1} {#2} }

__talk_toc_dest:n 261 \cs_new_protected:Npn \l@subsection #1#2

__talk_toc_dest:w 262 {

__talk_toc_level:nnnn 263 __talk_toc_aux:nnnn

264 { 2 }

265 {

266 \skip_set:Nn \leftskip { 2em }

267 \color_ensure_current:

268 }

269 {#1} {#2}

270 }

271 \cs_new_protected:Npn \l@subsubsection #1#2

272 {

273 __talk_toc_aux:nnnn

274 { 3 }

275 {

276 \skip_set:Nn \leftskip { 4em }

277 \color_ensure_current:

278 \footnotesize

279 }

280 {#1} {#2}

281 }

282 \cs_new_protected:Npn __talk_toc_aux:nnnn #1#2#3#4

283 {

```

284 \int_compare:nNnTF { \value { section } } < 1
285 { \use:n }
286 { \__talk_toc_dest:n }
287 { \__talk_toc_level:nnnn {#1} {#2} {#3} {#4} }
288 }

```

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.

```

289 \cs_new_protected:Npn \__talk_toc_dest:n
290 {
291   \exp_after:wN \__talk_toc_dest:w \@contentsline@destination
292   . 0 . 0 . 0 . \q_stop
293 }
294 \cs_new_protected:Npn \__talk_toc_dest:w #1 . #2 . #3 . #4 . #5 \q_stop #6
295 {
296   \bool_lazy_or:nnTF
297   {
298     \bool_lazy_and_p:nn
299     { \int_compare_p:nNn { \value { section } } = {#2} }
300     { \int_compare_p:nNn { \value { subsection } } = 0 }
301   }
302   {
303     \bool_lazy_and_p:nn
304     { \int_compare_p:nNn { \value { section } } = {#2} }
305     { \int_compare_p:nNn { \value { subsection } } = {#3} }
306   }
307   {#6}
308   {
309     \opacity_begin:n { 0.2 }
310     #6
311     \opacity_end:
312   }
313 }
314 \cs_new_protected:Npn \__talk_toc_level:nnnn #1#2#3#4
315 {
316   \int_compare:nNnF {#1} > { \value { tocdepth } }
317   {
318     \group_begin:
319     \noindent
320     #2
321     \UseHookWithArguments { contentsline / text / before } { 4 }
322     {#1} {#3} {#4} { \@contentsline@destination }
323     #3
324     \UseHookWithArguments { contentsline / text / after } { 4 }
325     {#1} {#3} {#4} { \@contentsline@destination }
326     \UseHookWithArguments { contentsline / page / before } { 4 }
327     {#1} {#3} {#4}
328     { \@contentsline@destination }
329     \UseHookWithArguments { contentsline / page / after } { 4 }
330     {#1} {#3} {#4}
331     { \@contentsline@destination }
332     \par
333   \group_end:

```

```

334         \vfil
335     }
336 }

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

337 \setcounter { tocdepth } { 2 }

```

1.4 Block environments

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

```

    quote (env.) 338 \NewDocumentEnvironment { description } { } { } { }
    quotation (env.) 339 \NewDocumentEnvironment { quote } { } { } { }
    verse (env.) 340 \NewDocumentEnvironment { quotation } { } { } { }
    stdquote (env.) 341 \NewDocumentEnvironment { verse } { } { } { }
    stdquotation (env.) 342 \AddToHook { begindocument / before }
    stdverse (env.) 343 {
344     \clist_map_inline:nn { quote , quotation , verse }
345     {
346         \NewEnvironmentCopy { std #1 } {#1}
347         \RenewDocumentEnvironment {#1} { d <> !0 { } }
348         {
349             \__talk_action_begin:n {##1}
350             \begin { std #1 } [ {##2} ]
351             \ignorespaces
352         }
353         {
354             \end { std #1 }
355             \__talk_action_end:
356         }
357     }
358 }

```

block (*env.*)

```

359 \NewDocumentEnvironment { block } { d <> m }
360 {
361     \__talk_action_begin:n {#1}
362     \par
363     \vbox_set:Nw \l__talk_tmp_box
364     \group_begin:
365         \medskip
366         \leavevmode
367         \normalfont \large \bfseries
368         \color { structure }
369         #2
370         \par
371         \medskip
372     \group_end:
373 }
374 {
375     \vbox_set_end:
376     \box_use:N \l__talk_tmp_box
377     \par
378     \__talk_action_end:
379 }

```

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.

```

380 \AddToHook { begindocument / before }
381 {
382   \NewCommandCopy \stditem \item
383   \RenewDocumentCommand \item { d <> o }
384   {
385     \IfNoValueTF {#2}
386     { \stditem }
387     { \stditem [ {#2} ] }
388     \IfNoValueTF {#1}
389     {
390       \exp_after:wN \__talk_item_parse_spec:w
391       \l__talk_action_spec_str < all > \q_stop
392     }
393     { \__talk_item_parse_spec:n {#1} }
394   }
395 }

```

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.

```

396 \cs_new_protected:Npn \__talk_item_parse_spec:w #1 < #2 > #3 \q_stop
397 { \__talk_item_parse_spec:n {#2} }
398 \cs_new_protected:Npn \__talk_item_parse_spec:n #1
399 {
400   \bool_lazy_or:nnF
401   { \tl_if_blank_p:n {#1} }
402   { \str_if_eq_p:nn {#1} { all } }
403   {
404     \tl_set:Nx \l__talk_list_end_tl
405     {
406       \exp_not:N \int_compare:nNnT \tex_currentgrouplevel:D =
407       { \int_eval:n { \tex_currentgrouplevel:D + 1 } }
408       {
409         \__talk_action_end:
410         \tl_clear:N \exp_not:N \l__talk_list_end_tl
411       }
412     }
413     \__talk_action_begin:n {#1}
414   }
415 }

```

(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
```

```

416 \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 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.

```

417 \cs_if_exist:NTF \l__block_topsepadd_skip
418 {
419   \cs_gset_protected:Npn \__block_inter_item:
420   {
421     \legacy_if:nT { @inlabel }
422     { \indent \par }
423     \mode_if_horizontal:T
424     {
425       \__block_skip_remove_last:
426       \__block_skip_remove_last:
427       \par
428     }
429     \l__talk_list_end_tl
430     \__kernel_list_item_end:
431     \__kernel_list_item_begin:
432     \addpenalty \@itempenalty
433     \addvspace \itemsep
434   }

```

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

```

435 \cs_gset_protected:Npe \BlockEnvEnd
436 {
437   \exp_not:n
438   { \__block_debug_typeout:n { blockenv~common~ending \on@line } }
439   \cs_if_exist:NTF \l__block_transparent_level_bool
440   {
441     \exp_not:N \bool_if:NF
442     \exp_not:N \l__block_transparent_level_bool
443   }
444   {
445     \exp_not:N \bool_if:NT
446     \exp_not:N \l__block_level_incr_bool
447   }
448   { \int_gdecr:N \exp_not:N \g_block_nesting_depth_int }
449   \exp_not:n
450   {
451     \legacy_if:nT { @inlabel }
452     {
453       \mode_leave_vertical:
454       \legacy_if_gset_false:n { @inlabel }
455     }
456     \__block_if_list:T
457     { \legacy_if:nT { @newlist } { \@noitemerr } }
458     \mode_if_horizontal:TF
459     {
460       \__block_skip_remove_last:
461       \__block_skip_remove_last:
462       \par

```

```

463     }
464     { \@inmatherr { \end { \@currenvir } } }
465 \l__talk_list_end_tl
466 \__kernel_displayblock_end:
467 \__block_if_list:T { \legacy_if_gset_false:n { @newlist } }
468 \legacy_if_gset_false:n { @nobreak }
469 \legacy_if:nF { @noparlist }
470 {
471     \__block_skip_set_to_last:N \l_tmpa_skip
472     \dim_compare:nNnT \l_tmpa_skip > \c_zero_dim
473     {
474         \skip_vertical:n { - \l_tmpa_skip }
475         \skip_vertical:n
476         { \l_tmpa_skip + \parskip - \@outerparskip }
477     }
478     \addpenalty \@endparpenalty
479     \addvspace \l__block_topsepadd_skip
480 }
481 \socket_use:n { block / endpe }
482 }
483 }
484 }

```

(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.

```

485 {
486     \AddToHook { item / after }
487     { \l__talk_list_end_tl }
488 }

```

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

`enumerate (env.)`

`description (env.)`

```

489 \AddToHook { begindocument / before }
490 {
491     \clist_map_inline:nn { itemize , enumerate , description }
492     {
493         \RenewDocumentEnvironment {#1} { = { action-spec } !o }
494         { \SimpleBlockEnv {#1} {##1} }
495         { \BlockEnvEnd }
496     }
497 }

```

And add the structural color to item labels.

```

498 \AddToHook { begindocument / before }
499 {
500     \EditInstance { item } { basic }
501     { label-format = \color { structure } #1 }
502     \EditInstance { item } { description }
503     { label-format = \normalfont \bfseries \color { structure } #1 }
504 }

```

`\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.

```
505 \keys_define:nn { template / block / std }
506   { action-spec .str_set:N = \l__talk_action_spec_str }
```

(End of definition for `\l__talk_action_spec_str`.)

1.6 Theorems, etc.

`\newtheorem` We need to extend the creation of theorems by adding the overlay argument. This means
`\stdnewtheorem` grabbing all of the arguments up-front, then having to pass them on to different versions of `\newtheorem` depending on what else has been loaded.

```
507 \NewCommandCopy \stdnewtheorem \newtheorem
508 \RenewDocumentCommand \newtheorem { s m o m o }
509   {
510     \use:e
511     {
512       \exp_not:N \stdnewtheorem
513       \IfBooleanT #1 { * }
514       { \exp_not:n {#2} }
515       \IfNoValueF {#3} { \exp_not:n { [ {#3} ] } }
516       { \exp_not:n {#4} }
517       \IfNoValueF {#5} { \exp_not:n { [ {#5} ] } }
518     }
519     \NewEnvironmentCopy { std #2 } {#2}
520     \RenewDocumentEnvironment {#2} { D <> { all } o }
521     {
522       \__talk_action_begin:n {##1}
523       \IfNoValueTF {##2}
524       { \begin { std #2 } }
525       { \begin { std #2 } [ {##2} ] }
526       \ignorespaces
527     }
528     {
529       \end { std #2 }
530       \__talk_action_end:
531     }
532   }
```

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

1.7 Bibliographies

`thebibliography` The article definition but without the section part.

```
533 \NewDocumentEnvironment { thebibliography } { m }
534   {
535     \list { \@biblabel { \@arabic \c@enumiv } }
536     {
537       \settowidth \labelwidth { \@biblabel {#1} }
538       \leftmargin \labelwidth
539       \advance \leftmargin \labelsep
540       \@openbib@code
```

```

541     \usecounter { enumiv }
542     \let \p@enumiv \@empty
543     \renewcommand \theenumiv { \@arabic \c@enumiv }
544   }
545   \sloppy
546   \clubpenalty 4000
547   \@clubpenalty \clubpenalty
548   \widowpenalty 4000
549   \sfcode `\. \@m
550 }
551 {
552   \def \@noitemerr
553     { \@latex@warning { Empty~`thebibliography'~environment } }
554   \endlist
555 }
556 \cs_set_eq:NN \@openbib@code \@empty

```

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

1.8 Supplementary material

`\l__talk_appendix_bool`

```

557 \bool_new:N \l__talk_appendix_bool

```

(End of definition for \l__talk_appendix_bool.)

`\appendix` Set up the global boolean and the TOC control.

```

558 \NewDocumentCommand \appendix { D <> { all } }
559 {
560   \__talk_if_overlay:nT {#1}
561   { \bool_set_true:N \l__talk_appendix_bool }
562   \addtocontents { toc } { \appendixmarker }
563   \cs_gset_eq:NN \contentsline \use_none:nnnn
564 }

```

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

```

565 </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 with just some expansion needed.

```

139 \ExpandArgs { Nne } \NewDocumentCommand \maketitle { 0 {} }
140 {
141   \exp_not:N \bool_if:NTF \exp_not:N \l__talk_frame_bool
142   { \exp_not:N \UseTemplate { titlepage } { talk } {##1} }
143   {
144     \exp_not:N \begin { frame }
145       \bool_if:NT \l__talk_frame_title_bool { { } }
146       \exp_not:N \UseTemplate { titlepage } { talk } {##1}
147     \exp_not:N \end { frame }
148   }
149 }

```

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

```

150 </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
\.	549
@ commands:	
\@_decode_overlay_+::nw	166
@starttoc@aux internal commands:	
\starttoc@aux__talknn	228
\\	350
A	
\abovecaptionskip	156, 158
\action	144
actionenv (env.)	144
\addcontentsline	189
\addcontentslinebookmark	170, 182
\addcontentslinebookmarkOff	170, 187
\addcontentslinebookmarkOn	170
\addcontentslinebookmarkReset	172
\addpenalty	432, 478
\addtocontents	562
\AddToHook	54, 238, 342, 346, 380, 411, 450, 451, 486, 489, 494, 498, 614
\addtolength	48
\addvspace	433, 479
\advance	539
\againframe	589
\alert	44, 122
alertenv (env.)	129
\alt	217
\and	131
\appendix	558, 619
\appendixmarker	202, 562
\arabic	479, 483
\arraycolsep	25
\arrayrulewidth	27
\AssignSocketPlug	194
\AssignStructureRole	125, 127
\AssignTaggingSocketPlug	227
\author	15
B	
\begin	122, 126, 130, 144, 350, 435, 524, 525
\begingroup	163, 230, 412
\belowcaptionskip	157, 159
\bfseries	21, 39, 260, 367, 503
\bigskipamount	18
block (env.)	359
block commands:	
\g_block_nesting_depth_int	448
block internal commands:	
_block_debug_typeout:n	438
_block_if_list:TF	456, 467
_block_inter_item:	417, 419
\l_block_level_incr_bool	446
_block_skip_remove_last:	425, 426, 460, 461
_block_skip_set_to_last:N	471
\l_block_topsepadd_skip	66, 67, 417, 479
\l_block_transparent_level_bool	439, 442
\BlockEnvEnd	417, 495
bool commands:	
\bool_do_until:nn	113
\bool_do_while:Nn	30
\bool_gset_eq:NN	102
\bool_gset_false:N	37, 101
\bool_gset_true:N	100, 252, 262, 268
\bool_if:NTF	6, 44, 46, 46, 46, 50, 50, 58, 66, 68, 69, 75, 80, 83, 87, 102, 128, 135, 141, 145, 149, 187, 200, 204, 204, 232, 245, 337, 441, 445, 503, 509, 514, 538, 546, 618
\bool_lazy_and:nnTF	21, 56, 56, 169, 265, 338
\bool_lazy_and_p:nn	298, 303
\bool_lazy_or:nnTF	5, 18, 23, 296, 400
\bool_lazy_or_p:nn	24
\bool_new:N	3, 3, 4, 4, 8, 9, 10, 15, 140, 142, 143, 220, 472, 474, 475, 557
\bool_set_eq:NN	97, 179, 183
\bool_set_false:N	32, 33, 34, 35, 36, 44, 154, 172, 553, 575
\bool_set_true:N	27, 28, 60, 61, 78, 82, 83, 119, 152, 170, 172, 188, 230, 249, 253, 259, 264, 511, 561, 562, 584
\c_false_bool	15
\c_true_bool	14, 196, 213
box commands:	
\box_dp:N	36, 63
\box_gclear:N	95
\box_gset_eq:NN	110
\box_ht:N	59, 62, 68, 296, 321
\box_if_empty:NTF	129
\box_if_empty_p:N	113

<code>\group_end:</code>	32, 40, 41, 63, 75, 85, 127, 150, 215, 257, 333, 343, 372, 432, 611
<code>\group_insert_after:N</code>	45
H	
hbox commands:	
<code>\hbox:n</code>	118, 288
<code>\hbox_set:Nw</code>	45, 72
<code>\hbox_set_end:</code>	25, 72, 86
<code>\hbox_set_to_wd:Nnw</code>	18
<code>\headsep</code>	311, 330, 331
<code>\hfil</code>	83, 361, 405, 439, 450, 455, 465
hook commands:	
<code>\hook_gput_code:nnn</code>	56, 115, 333
<code>\hook_new:n</code>	167
<code>\hook_use:n</code>	165
<code>\hspace</code>	287, 295
<code>\hypersetup</code>	179
I	
<code>\IfBooleanT</code>	396, 513
<code>\ifcase</code>	5
<code>\IfFormatAtLeastF</code>	7
<code>\IfNoValueF</code>	397, 398, 515, 517
<code>\IfNoValueTF</code>	15, 25, 36, 47, 67, 177, 254, 385, 388, 523
<code>\ignorespaces</code>	...
<code>\immediate</code>	19, 21, 92, 167, 226, 266, 351, 526
<code>\includegraphics</code>	241
<code>\indent</code>	387
<code>\insertsection</code>	422
<code>\insertsubsection</code>	114
<code>\insertsubsubsection</code>	114
<code>\institute</code>	114
<code>\institute</code>	34
int commands:	
<code>\int_compare:nNnTF</code>	82, 197, 248, 251, 258, 261, 284, 316, 406, 616
<code>\int_compare_p:nNn</code>	266, 267, 299, 300, 304, 305
<code>\int_eval:n</code>	258, 407, 485
<code>\int_gdecr:N</code>	448
<code>\int_gincr:N</code>	...
<code>\int_gset:Nn</code>	36, 49, 81, 255, 440, 508, 510, 520
<code>\int_gset_eq:NN</code>	172, 256
<code>\int_gset_eq:NN</code>	27, 200, 205, 210
<code>\int_gzero:N</code>	16, 28, 92
<code>\int_incr:N</code>	289
<code>\int_max:nn</code>	218
<code>\int_new:N</code>	5, 8, 9, 10, 188, 199, 216, 279, 433, 473, 476, 480
<code>\int_set_eq:NN</code>	15
<code>\int_step_inline:nn</code>	620
<code>\int_to_Roman:n</code>	282
<code>\int_to_roman:n</code>	283
<code>\int_use:N</code>	11, 17, 56, 124, 140, 442, 446, 488, 491, 522
<code>\c_max_int</code>	242, 267
<code>\invisible</code>	108
<code>\invisibleenv (env.)</code>	117
iow commands:	
<code>\iow_now:Nn</code>	325
<code>\item</code>	68, 380
<code>itemize (env.)</code>	489
<code>\itemsep</code>	54, 62, 69, 433
J	
<code>\jobname</code>	235, 241
K	
kernel internal commands:	
<code>__kernel_displayblock_end:</code>	466
<code>__kernel_list_item_begin:</code>	431
<code>__kernel_list_item_end:</code>	430
keys commands:	
<code>\l_keys_choice_str</code>	78, 81
<code>\keys_define:nn</code>	...
<code>\keys_set:nn</code>	3, 5, 6, 31, 65, 72, 85, 95, 221, 505
<code>\keys_set_known:nn</code>	7, 17, 17, 24, 29, 36, 41, 50, 84, 102, 148, 244, 552, 561, 574, 583, 608
<code>\l_keys_value_tl</code>	20, 32
<code>\l_keys_value_tl</code>	46, 241
L	
<code>\label</code>	77, 403
<code>\labelenumi</code>	34
<code>\labelenumii</code>	35
<code>\labelenumiii</code>	36
<code>\labelenumiv</code>	37
<code>\labelitemfont</code>	38, 39, 40, 41, 42
<code>\labelitemi</code>	38
<code>\labelitemii</code>	39
<code>\labelitemiii</code>	40
<code>\labelitemiv</code>	41
<code>\labelsep</code>	29, 33, 46, 48, 539
<code>\labelwidth</code>	47, 48, 537, 538
<code>\Large</code>	21, 90
<code>\large</code>	367
<code>\leavevmode</code>	89, 366
<code>\leftmargin</code>	51, 59, 66, 538, 539
<code>\leftmargini</code>	43, 47, 51
<code>\leftmarginii</code>	44, 59
<code>\leftmarginiii</code>	45, 66
<code>\leftskip</code>	266, 276
legacy commands:	
<code>\legacy_if:nTF</code>	...
<code>\legacy_if:nTF</code>	48, 236, 252, 323, 421, 451, 457, 469

<code>\legacy_if_gset_false:n</code>	454, 467, 468	O	
<code>\let</code>	542	<code>\obeyedline</code>	19, 82
<code>\list</code>	535	<code>\only</code>	49, 132
		<code>onlyenv (env.)</code>	137
M		<code>\onslide</code>	45, 223
<code>\makeatletter</code>	231	opacity commands:	
<code>\maketitle</code>	139	<code>\opacity_begin:n</code>	39, 309
<code>\mathcolor</code>	5, 11	<code>\opacity_end:</code>	41, 311
<code>\mbox</code>	381	<code>\openout</code>	241
<code>\medskip</code>	365, 371	<code>\or</code>	5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
<code>\mode</code>	40, 11	<code>overlayarea (env.)</code>	268
mode commands:		<code>overprint (env.)</code>	285
<code>\mode_if_horizontal:TF</code>	31, 423, 458		
<code>\mode_if_horizontal_p:</code>	25	P	
<code>\mode_if_math:TF</code>	380	<code>\pagecolor</code>	43
<code>\mode_if_vertical_p:</code>	26	<code>\pagestyle</code>	471
<code>\mode_leave_vertical:</code>	32, 383, 439, 453	<code>\paperheight</code>	57, 58
<code>\month</code>	5	<code>\paperwidth</code>	58, 382, 430, 431
msg commands:		<code>\par</code>	34, 36, 14, 28, 34, 39, 68, 73, 80, 95, 100, 162, 168, 263, 277, 308, 332, 362, 370, 377, 422, 427, 462
<code>\msg_error:nnn</code>	138, 156, 161, 208, 225, 533, 596	<code>\parsep</code>	53, 61, 62, 68, 69
<code>\msg_fatal:nn</code>	15	<code>\parskip</code>	476
<code>\g_msg_module_name_prop</code>	5	<code>\partopsep</code>	71
<code>\g_msg_module_type_prop</code>	6	<code>\pause</code>	45, 250
<code>\msg_new:nnn</code>	22, 348	pdfxform commands:	
<code>\msg_new:nnnn</code>	9, 273, 279, 285, 624, 630	<code>\pdfxform_new:nnn</code>	441
<code>\msg_warning:nn</code>	27, 344	<code>\pdfxform_use:n</code>	445
N		prg commands:	
<code>\NeedsDocumentMetadata</code>	17	<code>\prg_generate_conditional_-</code> <code>variant:Nnn</code>	10
<code>\NewCommandCopy</code>	4, 5, 6, 173, 368, 382, 388, 499, 507, 613	<code>\prg_new_protected_conditional:Npnn</code>	3, 3
<code>\newcounter</code>	23, 105, 106, 107, 150	<code>\prg_return_false:</code>	8, 9
<code>\NewDocumentCommand</code>	5, 10, 11, 34, 39, 65, 110, 120, 122, 125, 132, 139, 144, 202, 217, 223, 240, 250, 253, 558, 605	<code>\prg_return_true:</code>	7, 8
<code>\NewDocumentEnvironment</code>	11, 78, 117, 129, 137, 141, 152, 268, 285, 338, 339, 340, 341, 359, 533, 557, 579	<code>\ProcessedArgument</code>	14, 15
<code>\NewEnvironmentCopy</code>	346, 519	<code>\ProcessKeyOptions</code>	110
<code>\newlabel</code>	416	prop commands:	
<code>\newlength</code>	156, 157	<code>\prop_gput:Nnn</code>	5, 6
<code>\NewSocketPlug</code>	181	property commands:	
<code>\NewTaggingSocket</code>	210	<code>\property_new:nnnn</code>	11, 484, 487, 490
<code>\NewTaggingSocketPlug</code>	211	<code>\property_record:nn</code>	140, 496
<code>\NewTemplateType</code>	15, 44, 93, 94, 104, 269, 305, 352	<code>\property_ref:nn</code>	17, 486, 489, 492
<code>\newtheorem</code>	68, 507	<code>\protect</code>	198
<code>\newwrite</code>	240	<code>\providecommand</code>	170, 171, 172, 173
<code>\noindent</code>	31, 319, 327, 374	<code>\ProvidesExplClass</code>	3
<code>\normalfont</code>	42, 50, 310, 367, 503	<code>\put</code>	57
<code>\normalsize</code>	166	Q	
<code>\number</code>	20, 22	<code>\quad</code>	131
<code>\numberline</code>	198	quark commands:	
		<code>\quark_if_recursion_tail_stop:N</code>	178
		<code>\quark_if_recursion_tail_stop_-</code> <code>do:Nn</code>	195, 206

<code>\quark_if_recursion_tail_stop_-</code>	<code>\SimpleBlockEnv</code>	494
<code>do:nn</code>	<code>\skip</code>	30
<code>\q_recursion_stop</code>	skip commands:	
<code>\q_recursion_tail</code>	<code>\skip_horizontal:n</code>	
<code>\q_stop</code>		32, 332, 379, 408, 425, 431
161, 165, 233, 236, 292, 294, 391, 396	<code>\skip_if_eq_p:nn</code>	22
quotation (env.)	<code>\skip_new:N</code>	17
quote (env.)	<code>\skip_set:Nn</code>	266, 276
	<code>\skip_set_eq:NN</code>	20
	<code>\skip_vertical:n</code>	33, 147, 149,
	153, 155, 159, 161, 165, 167, 474, 475	
	<code>\l_tmpa_skip</code>	471, 472, 474, 476
	<code>\sloppy</code>	545
	socket commands:	
	<code>\socket_use:n</code>	481
	<code>\space</code>	19, 21
	<code>\stdcolor</code>	4
	<code>\stdemph</code>	353
	<code>\stdfootnote</code>	173, 178, 179
	<code>\stdincludegraphics</code>	387
	<code>\stditem</code>	382, 386, 387
	<code>\stdmathcolor</code>	4
	<code>\stdnewtheorem</code>	507
	<code>stdquotation</code> (env.)	338
	<code>stdquote</code> (env.)	338
	<code>\stdtextbf</code>	353
	<code>\stdtextcolor</code>	4
	<code>\stdtextit</code>	353
	<code>\stdtextmd</code>	353
	<code>\stdtextnormal</code>	353
	<code>\stdtextrm</code>	353
	<code>\stdtextsc</code>	353
	<code>\stdtextsf</code>	353
	<code>\stdtextsl</code>	353
	<code>\stdtexttt</code>	353
	<code>\stdtextup</code>	353
	<code>stdverse</code> (env.)	338
	<code>\stepcounter</code>	23
	str commands:	
	<code>\str_case:nn</code>	516
	<code>\str_clear:N</code>	
		22, 168, 551, 560, 573, 582, 609
	<code>\str_if_empty:NTF</code>	114, 135, 145, 512
	<code>\str_if_empty_p:N</code>	57
	<code>\str_if_eq:nnTF</code>	
		20, 79, 111, 157, 159, 169
	<code>\str_if_eq_p:nn</code>	6, 7, 25, 170, 171, 402
	<code>\str_new:N</code>	11, 13, 14, 17, 83, 84, 141
	<code>\str_put_right:Nn</code>	181, 217
	<code>\str_replace_all:Nnn</code>	23, 25, 59
	<code>\str_set:Nn</code>	21, 31, 74, 147
	<code>\str_set_eq:NN</code>	78, 81, 181
	<code>\string</code>	416
	<code>\subsection</code>	105, 114
R		
<code>\raggedright</code>	90, 152, 265	
<code>\refstepcounter</code>	136	
<code>\relax</code>	241	
<code>\relsize</code>	116	
<code>\renewcommand</code>	543	
<code>\RenewCommandCopy</code>	29	
<code>\RenewDocumentCommand</code> 11, 15, 21, 22, 27,		
30, 43, 174, 369, 383, 389, 403, 428, 508		
<code>\RenewDocumentEnvironment</code>		
.	347, 493, 520, 548, 570	
<code>\RequirePackage</code>	3, 114, 135, 153,	
156, 157, 160, 163, 169, 170, 178, 387		
<code>\reuseframe</code>	589	
<code>\rmdefault</code>	171	
<code>\rule</code>	58	
rule commands:		
<code>\rule:nnn</code>	48, 430	
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>	217	
<code>\seq_gremove_all:Nn</code>	108, 109, 110	
<code>\seq_gset_from_clist:Nn</code>	190	
<code>\seq_map_indexed_inline:Nn</code>	116	
<code>\seq_map_inline:Nn</code>	197, 204, 209	
<code>\seq_new:N</code>	172, 189	
<code>\seq_set_from_clist:Nn</code>	114	
<code>\l_tmpa_seq</code>	114, 116	
<code>\setcounter</code>	24, 337	
<code>\setlength</code>	25, 26, 27, 28, 29, 31,	
32, 33, 43, 44, 45, 46, 47, 71, 158, 159		
<code>\setmainfont</code>	164	
<code>\setmathfont</code>	166	
<code>\setsansfont</code>	165	
<code>\SetTemplateKeys</code>	121	
<code>\settowidth</code>	537	
<code>\sfcode</code>	549	
<code>\sfdefault</code>	171	

\subsubsection	105 , 114	\l__talk_appendix_bool	
\subtitle	34		204 , 509 , 557 , 561 , 618
sys commands:		\g__talk_appendixframe_int	
\c_sys_engine_str	24		480 , 485 , 491 , 510
\sys_if_engine luatex:TF ..	158 , 448	\l__talk_aspect_ratio_str ..	65 , 130
\sys_if_engine luatex_p:	19	\g__talk_break_box	5 , 110 , 112
\sys_if_engine opentype:TF	154	\l__talk_break_box	
\sys_if_engine pdftex_p:	20		6 , 107 , 108 , 110 , 112 , 113 , 115
T			
\tabbingsep	29	\g__talk_cnt_reset_seq	
\tabcolsep	26		108 , 109 , 110 , 189 , 204 , 209 , 217
table (env.)	139	__talk_cnt_restore:	76 , 202 , 207
\tablename	150	__talk_cnt_save:	96 , 202 , 202
\tableofcontents	253	__talk_column_align_bottom:n	54 , 54
tag commands:		__talk_column_align_center:n	54 , 56
\tag_get:n	172	__talk_column_align_top:n ..	54 , 73
\tag_mc_begin:n	180 , 259 , 266	\l__talk_column_alignment_tl ..	31 , 98
\tag_mc_end:	186 , 257 , 264	\g__talk_column_int	9 , 15 , 16 , 27 , 81 , 82
\tag_resume:n	85 , 185 , 256	\l__talk_column_int	9 , 15 , 27
\tag_struct_begin:n ..	91 , 171 , 218 , 258	\l__talk_columns_wd_tl	5 , 18 , 19
\tag_struct_end: ...	92 , 176 , 225 , 265	__talk_decode_action:n	113 , 122 , 122
\tag_suspend:n	73 , 181 , 267	__talk_decode_action:w	122 , 124 , 129
tag internal commands:		\l__talk_decode_action_str	
\g__tag_title_author_tl	19		14 , 22 , 135 , 145 , 147 , 182 , 190
\g__tag_title_title_tl	31	\l__talk_decode_actions_bool ...	
\tagpdfparaOff	61		15 , 32 , 184 , 192
\tagpdfsetup	161 , 180	\l__talk_decode_actions_clist ..	15 , 37
talk commands:		\l__talk_decode_actions_str	15
\l__talk_frame_properties_clist .		\l__talk_decode_arg_str	
.....	137 , 493	...	11 , 31 , 39 , 139 , 157 , 162 , 209 , 226
talk internal commands:		__talk_decode_check:n ..	174 , 221 , 221
__talk_action_alert:N ..	42 , 42 , 125 , 130	__talk_decode_check:nw ..	221 , 233 , 236
__talk_action_begin:n ..	13 , 91 , 143 , 144 , 147 , 153 , 155 , 349 , 361 , 413 , 522	__talk_decode_check_range:nnn .	
__talk_action_begin:w ..	144 , 160 , 165		221 , 242 , 243 , 256
__talk_action_begin_auxi:n	144 , 163 , 166 , 167	__talk_decode_check_single:nn .	
__talk_action_begin_auxii:n	144 , 173 , 176		221 , 239 , 246
__talk_action_end: ..	36 , 29 , 96 , 144 , 148 , 149 , 154 , 198 , 355 , 378 , 409 , 530	__talk_decode_mode:n	64 , 74 , 74
__talk_action_invisible:N ..	48 , 48 , 196	__talk_decode_mode:nn ..	104 , 107 , 109
__talk_action_invisible_end:N .		__talk_decode_mode:w	74 , 89 , 96
.....	48 , 54 , 213	__talk_decode_mode_aux:n	74
__talk_action_only:N	64 , 64 , 138	\l__talk_decode_mode_bool	
__talk_action_only_end:N ..	64 , 78 , 139		8 , 27 , 35 , 60 , 82 , 172 , 503
\l__talk_action_spec_str		\l__talk_decode_modes_bool	
.....	161 , 220 , 391 , 505		9 , 36 , 58 , 78 , 119
__talk_action_uncover:N ..	100 , 100 , 189	__talk_decode_overlay_:nw	166
__talk_action_uncover_end:N ...		__talk_decode_overlay_aux:nNN .	
.....	100 , 106 , 191		166 , 189 , 192 , 193
__talk_action_visible:N	48 , 56	__talk_decode_overlay_offset:nNn	
__talk_action_visible_end:N ..	48 , 62		166 , 197 , 202 , 212 , 215
		__talk_decode_overlay_offset:nNnN	
			166 , 201 , 204 , 213
		__talk_decode_overlays:nN	
			166 , 173 , 176 , 182 , 219
		__talk_decode_overlays:nn	
			115 , 136 , 150 , 166 , 166

\l__talk_decode_overlays_bool .	3 ,	__talk_frame_reuse_aux:nn	
	6 , 28 , 33 , 61 , 83 , 152 , 180 , 187 , 189 , 191		589 , 593 , 598
\l__talk_decode_overlays_clist .		__talk_frame_reuse_aux:nnn	
	12 , 38 , 151		589 , 599 , 600
\l__talk_decode_overlays_str		__talk_frame_save:nn	
	12 , 57 , 114		500 , 513 , 521 , 530 , 545
__talk_decode_parse:n		\g__talk_frame_struct_int	56 , 172 , 188
	5 , 18 , 18 , 178 , 188 , 502	\g__talk_frame_subtitle_tl	3 , 13 , 94
__talk_decode_parse:w .	18 , 45 , 52 , 66	\l__talk_frame_suppl_bool .	220 , 514
__talk_decode_parse_auxi:n		\g__talk_frame_suppl_int	473 , 520 , 522 , 616 , 620
	18 , 19 , 20	\g__talk_frame_tag_bool	46 , 46 , 68 , 69 , 83 , 100 , 101 , 232 , 474
__talk_decode_parse_auxii:n		\l__talk_frame_tagging_str	20 , 21 , 23 , 25 , 99 , 220
	18 , 39 , 42	__talk_frame_title:n	15 , 38 , 44
\l__talk_decode_step_bool	10 , 44 , 46 , 188	\l__talk_frame_title_bool	65 , 145 , 546
\l__talk_decode_trailing_bool	4 , 34 , 245 , 253 , 259	__talk_frame_title_tagged:n	15 , 47 , 51
\l__talk_float_alignment_tl	103 , 115 , 116 , 117 , 126 , 136	\g__talk_frame_title_tl	3 , 8 , 58 , 93 , 341 , 563
\l__talk_fontsize_dim	65 , 111 , 116	\l__talk_frame_verb_bool	50 , 475 , 538 , 553 , 562 , 575 , 584 , 602
\l__talk_footelem_color_tl	269	\l__talk_frametitle_after_skip	25 , 41
\l__talk_footelem_font_tl	269	\l__talk_frametitle_before_skip	26 , 32
\l__talk_footelem_left_skip	269	\l__talk_frametitle_color_tl	27 , 34 , 35
\l__talk_footelem_right_skip	269	\l__talk_frametitle_font_tl	28 , 36
\l__talk_footer_bg_tl	352	__talk_head_marks:nnnnn	161 , 174 , 174 , 192
\l__talk_footer_fg_tl	352	__talk_head_marks_aux:Nnn	174
\l__talk_footer_font_tl	352	__talk_head_marks_aux:Nnnn	181 , 188 , 193
\l__talk_footer_left_skip	352	__talk_head_section:Nnn	114
\l__talk_footer_order_clist	352	__talk_head_subsection:Nnn	114
\l__talk_footer_right_skip	352	__talk_head_subsubsection:Nnn .	114
\l__talk_footer_sep_tl	352	__talk_head_tag:nn	210 , 213 , 216
\g__talk_footnote_box	95 , 129 , 132 , 171 , 183 , 185	\l__talk_header_bg_tl	305
\g__talk_footnote_overlay_seq	172 , 176 , 186	\l__talk_header_fg_tl	305
\l__talk_frame_alignment_tl	94 , 127 , 219 , 240	\l__talk_header_font_tl	305
\l__talk_frame_bool	128 , 141 , 472 , 511	\l__talk_header_frametitle_bool	305
\l__talk_frame_break_bool	57 , 87 , 220	\l__talk_header_ht_dim	305
\l__talk_frame_break_fp	109 , 116 , 220	\l__talk_header_left_skip	305
\g__talk_frame_int	17 , 124 , 140 , 282 , 476 , 485 , 488 , 508	\l__talk_header_right_skip	305
\l__talk_frame_name_str	220 , 512 , 513 , 551 , 560 , 573 , 582 , 609	__talk_header_tag_begin:n	53 , 254 , 254 , 261
__talk_frame_overprint:	280 , 280 , 292 , 295 , 299 ,	__talk_header_tag_end: .	64 , 254 , 262
	312 , 314 , 315 , 318 , 320 , 327 , 329 , 334	__talk_if_overlay:n	3 , 10
__talk_frame_process:nn	500 , 500 , 554 , 564 , 576 , 585 , 603	__talk_if_overlay:nTF . .	3 , 7 , 12 ,
__talk_frame_process_aux:nn	500 , 504 , 506		13 , 13 , 23 , 34 , 38 , 45 , 99 , 130 , 134 ,
__talk_frame_reuse:nn	589 , 589 , 610 , 621		219 , 232 , 242 , 372 , 391 , 406 , 430 , 560

_talk_item_parse_spec:n		380, 393, 397, 398
_talk_item_parse_spec:w		380, 390, 396
_talk_label:n		403, 407, 410
_talk_latex_frame:n	..	29, 499, 499
\l_talk_list_end_tl		404, 410, 416, 429, 465, 487
_talk_metadata_name:n		393, 396, 401, 417, 417
_talk_mode:n		3
_talk_mode:nTF		3, 12
\l_talk_mode_str		7, 65, 80, 111
\c_talk_modes_clist		68, 76
_talk_onslide:n	..	223, 225, 228, 257
\g_talk_onslide_tl		54, 91, 185, 202, 230, 231, 235, 239
_talk_opacity_begin:n		38, 38, 51, 52, 59, 60, 98, 103, 234
_talk_opacity_end:		38, 40, 55, 63, 107, 211, 236
_talk_opacity_group_begin:	...	434, 434, 450
_talk_opacity_group_end:		434, 436, 451
\g_talk_opacity_group_int		433, 440, 442, 446
\l_talk_overlay_all_bool		143, 170, 172, 200
_talk_overlay_arg:n		3, 11, 111, 118, 122, 129, 137
_talk_overprint_begin:n		261, 261, 269, 286
_talk_overprint_check_ht:n	...	285, 334, 336, 345
\l_talk_overprint_int	..	279, 283, 289
_talk_overprint_save_ht:		285, 290, 310
_talk_pagecolor:n		43, 48, 49, 52
\c_talk_paper_height_dim		118
\c_talk_paper_width_dim		118
\g_talk_pauses_int		10, 5, 49, 92, 218, 255, 256, 258
\l_talk_saved_action_str		140, 181, 207
\l_talk_saved_actions_bool		140, 183, 209
_talk_saved_contentline:nnnn	..	202, 205, 208, 209
_talk_saved_ifendpe:	..	64, 70, 88, 92
\l_talk_saved_overlays_bool	...	140, 179, 204
\l_talk_sec_bookmark_tl	..	72, 151, 163
\l_talk_sec_label_tl		72
\l_talk_sec_nameref_tl		72, 164
\l_talk_sec_numbered_bool	..	72, 154
\l_talk_sec_quote_tl		72
\l_talk_sec_running_tl		72, 152
\l_talk_sec_subtitle_tl		72
\l_talk_sec_toc_tl		72, 153, 162
\g_talk_section_tl		66
\l_talk_section_tl		66
\l_talk_shuffle_skip	..	17, 20, 22, 34
_talk_shuffle_skip:n	..	18, 18, 39, 41
_talk_slide:nn		12, 12, 525, 528
_talk_slide_align_bottom:n	145, 145	
_talk_slide_align_center:n	145, 151	
_talk_slide_align_stretch:n	..	145, 157
_talk_slide_align_top:n	..	145, 163
_talk_slide_auxi:nn		12, 31, 34
_talk_slide_auxii:n		12, 39, 41
_talk_slide_auxiii:	..	12, 63, 65, 111
_talk_slide_auxiv:n		12, 51, 79
\l_talk_slide_box	5, 7, 44, 45, 49,	59, 67, 73, 107, 108, 115, 117, 118, 128
_talk_slide_break:		62, 105, 105
\g_talk_slide_continue_bool	..	3,
		30, 37, 75, 98, 102, 135, 252, 262, 268
\l_talk_slide_continue_bool		3, 97, 103
_talk_slide_init:		43, 89, 89
\g_talk_slide_int		8,
		11, 28, 36, 248, 251, 258, 261, 266
_talk_slide_output:	77, 119, 122, 122	
_talk_slide_tag_off_begin:	...	48, 178, 178
_talk_slide_tag_off_end:		71, 178, 183
_talk_slide_tag_on_begin:		47, 169, 169
_talk_slide_tag_on_end:	70, 169, 174	
\g_talk_subsection_tl		66
\l_talk_subsection_tl		66, 158
\g_talk_subsubsection_tl		66
\l_talk_subsubsection_tl	..	66, 160
\l_talk_suppl_mode_str		65, 516
_talk_textcmd_equiv:n	..	353, 374, 378
\l_talk_titlelem_after_skip		44
\l_talk_titlelem_before_skip	...	44
\l_talk_titlelem_color_tl		44
\l_talk_titlelem_font_tl		44
\l_talk_titlelem_tag_begin_tl	..	44
\l_talk_titlelem_tag_end_tl		44
\l_talk_titlepage_alignment_tl	..	94
\l_talk_titlepage_framestyle_tl	..	94
\l_talk_titlepage_order_clist	..	94
_talk_tmp:w		61, 61, 121, 130

<code>\l__talk_tmp_box</code>	18, 26, 59, 60, <u>62</u> , 62, 63, 66, 68, 71, 71, 72, 75, 85, 99, 264, 274, 296, 301, 305, 307, 321, 363, 376, 380, 407, 435, 444
<code>\l__talk_tmp_clist</code>	63, 137, 138, 139, 141
<code>\l__talk_tmp_tl</code>	15, 21, 24, 26, <u>64</u> , 112, 187, 188, 390, 392, 393
<code>__talk_toc_aux:nnnn</code>	<u>259</u> , 260, 263, 273, 282
<code>__talk_toc_dest:n</code>	<u>259</u> , 286, 289
<code>__talk_toc_dest:w</code>	<u>259</u> , 291, 294
<code>__talk_toc_level:nnnn</code> .	<u>259</u> , 287, 314
<code>\l__talk_uncover_hidden_fp</code>	<u>93</u>
<code>\l__talk_vcenter_offset_tl</code>	<u>64</u> , 69, <u>75</u>
<code>__talk_wallpaper_hrule:Nnn</code>	328, 375, <u>423</u> , 423
<code>talk/sec/title</code>	210
<code>\temporal</code>	10, <u>240</u>
TeX and L ^A T _E X 2 _ε commands:	
<code>\@appendixframenumber</code>	<u>480</u>
<code>\@appendixframes</code>	<u>490</u>
<code>\@arabic</code>	7, 10, 111, 112, 113, 177, 478, 482, 535, 543
<code>\@author</code>	3, 18, 19
<code>\@auxout</code>	325, 414
<code>\@biblabel</code>	535, 537
<code>\@bsphack</code>	405
<code>\@caption</code>	<u>160</u>
<code>\@capytype</code>	<u>130</u>
<code>\@clubpenalty</code>	547
<code>\@contentsline@destination</code>	63, 291, 322, 325, 328, 331
<code>\@currentHref</code>	421
<code>\@currentlabel</code>	418
<code>\@currentlabelname</code>	176, 420
<code>\@currenvir</code>	464
<code>\@date</code>	3, 25
<code>\@definecounter</code>	<u>212</u>
<code>\@empty</code>	542, 556
<code>\@endparpenalty</code>	478
<code>\@esphack</code>	408
<code>\@evenfoot</code>	443, 458, 469
<code>\@evenhead</code>	442, 457, 468
<code>\@framenumber</code>	<u>476</u>
<code>\@frames</code>	<u>484</u>
<code>\@ignore</code>	36
<code>\@ignoretrue</code>	101
<code>\@inmatherr</code>	464
<code>\@input</code>	235
<code>\@institute</code>	3, 37
<code>\@itempenalty</code>	432
<code>\@kernel@reserved@label@data</code> ...	422
<code>\@latex@warning</code>	553
<code>\@listI</code>	56
<code>\@listi</code>	49, 56
<code>\@listii</code>	57
<code>\@listiii</code>	64
<code>\@m</code>	549
<code>\@makecaption</code>	167
<code>\@makefnrtxt</code>	<u>195</u>
<code>\@mpfootins</code>	30
<code>\@nobreakfalse</code>	244
<code>\@noitemerr</code>	457, 552
<code>\@oddfnfoot</code> .	441, 443, 452, 458, 467, 469
<code>\@oddhead</code> .	437, 442, 447, 457, 462, 468
<code>\@openbib@code</code>	540, 556
<code>\@outerparskip</code>	476
<code>\@parboxrestore</code>	88, 164
<code>\@setminipage</code>	165
<code>\@shortauthor</code>	3
<code>\@shortdate</code>	3
<code>\@shortinstitute</code>	3
<code>\@shortsubtitle</code>	3
<code>\@shorttitle</code>	3
<code>\@skiphyperreftrue</code>	37, 131
<code>\@starttoc</code>	<u>228</u> , 256
<code>\@starttoc@aux@nn</code>	233, 247
<code>\@subtitle</code>	3, 42
<code>\@title</code>	3, 30, 31
<code>\@totalframes</code>	<u>487</u>
<code>\c@appendixframe</code>	<u>480</u>
<code>\c@enumiv</code>	535, 543
<code>\c@figure</code>	<u>150</u>
<code>\c@frame</code>	<u>476</u>
<code>\c@page</code>	177
<code>\c@pauses</code>	5
<code>\c@section</code>	111
<code>\c@slide</code>	8
<code>\c@subsection</code>	112
<code>\c@subsubsection</code>	113
<code>\c@table</code>	<u>150</u>
<code>\check@mathfonts</code>	58
<code>\currentgrouplevel</code>	65
<code>\fnum@figure</code>	<u>150</u>
<code>\fnum@table</code>	<u>150</u>
<code>\Gm@bmargin</code>	378
<code>\Gm@lmargin</code>	312, 359, 425
<code>\Gm@rmargin</code>	314, 360, 408
<code>\Gm@tmargin</code>	311
<code>\if@endpe</code>	70, 88, 92
<code>\if@minipage</code>	165
<code>\ifmeasuring@</code>	11
<code>\ignorespaces</code>	36
<code>\l@section</code>	<u>259</u>
<code>\l@subsection</code>	<u>259</u>
<code>\l@subsubsection</code>	<u>259</u>
<code>\label@in@display</code>	52, 427, 428

<code>\on@line</code>	438	<code>\thepauses</code>	5
<code>\p@enumiv</code>	542	<code>\thesection</code>	105
<code>\protected@write</code>	414	<code>\theslide</code>	8
<code>\ps@plain</code>	435	<code>\thesubsection</code>	105
<code>\ps@talk</code>	435	<code>\thesubsubsection</code>	105
<code>\ps@wallpaper</code>	435	<code>\thetable</code>	150
<code>\reset@color</code>	62, 63	<code>\thispagestyle</code>	129
<code>\set@color</code>	61, 63	<code>\thispdfpagelabel</code>	124
<code>\std@definecounter</code>	212	<code>\tiny</code>	358
<code>\std@label@in@display</code>	427	<code>\title</code>	15
<code>\stdreset@color</code>	61	tl commands:	
<code>\stdset@color</code>	61	<code>\tl_clear:N</code>	151, 152, 153, 158, 160, 410
<code>\textsubscript@offset</code>	173	<code>\tl_clear_new:N</code>	534
<code>\textsubscript@space</code>	173	<code>\tl_gclear:N</code>	12, 13, 91, 93, 94, 231
<code>\textsuperscript@offset</code>	173	<code>\tl_gput_right:Nn</code>	235
<code>\textsuperscript@space</code>	173	<code>\tl_gset:Nn</code>	8, 13, 18, 25, 30, 37, 42, 315, 318, 535, 563
tex commands:		<code>\tl_gset_eq:NN</code>	14, 19, 31
<code>\tex_currentgrouplevel:D</code>	406, 407	<code>\tl_if_blank:N</code>	37, 102, 118, 133, 148, 177, 179, 186, 241
<code>\tex_hsize:D</code>	33, 44	<code>\tl_if_blank_p:n</code>	24, 401
<code>\tex_lastskip:D</code>	20	<code>\tl_if_empty:N</code>	34, 68, 128, 185, 202, 290, 334, 387, 396, 426
<code>\tex_setbox:D</code>	31, 42	<code>\tl_if_empty:N</code>	64, 238, 285
<code>\tex_unskip:D</code>	29	<code>\tl_if_exist:N</code>	312, 419, 532, 591
<code>\tex_vbox:D</code>	31, 42	<code>\tl_if_novalue:N</code>	158
<code>\tex_vrule:D</code>	50	<code>\tl_map_inline:nn</code>	353
<code>\texorpdfstring</code>	173, 189	<code>\tl_new:N</code>	3, 4, 49, 64, 66, 67, 68, 69, 70, 71, 75, 103, 104, 151, 153, 219, 239, 314, 416
text commands:		<code>\tl_put_right:N</code>	77
<code>\text_purify:n</code>	58, 214	<code>\tl_retokenize:n</code>	86
<code>\text_titlecase_first:n</code>	152	<code>\tl_set:Nn</code>	15, 76, 115, 115, 116, 116, 117, 117, 121, 122, 123, 124, 147, 152, 176, 404
<code>\textasteriskcentered</code>	40	<code>\tl_set_eq:NN</code>	45, 154, 240
<code>\textbf</code>	353	<code>\tl_to_str:n</code>	70, 71, 72, 89, 105, 122, 125, 130, 131, 565
<code>\textbullet</code>	38	<code>\tl_trim_spaces:n</code>	65
<code>\textcolor</code>	6, 11	<code>\tl_use:N</code>	54, 122, 230
<code>\textendash</code>	39	<code>\today</code>	3
<code>\textheight</code>	60, 109, 116, 125	token commands:	
<code>\textit</code>	353	<code>\token_if_eq_meaning:NNTF</code>	200, 211
<code>\textmd</code>	353	<code>\token_to_str:N</code>	97, 98
<code>\textnormal</code>	353	<code>\topsep</code>	52, 60, 67
<code>\textperiodcentered</code>	41	U	
<code>\textrm</code>	353	<code>\uncover</code>	108
<code>\textsc</code>	353	<code>\uncoverenv (env.)</code>	117
<code>\textsf</code>	353	<code>\unskip</code>	24
<code>\textsl</code>	353	use commands:	
<code>\texttt</code>	353	<code>\use:N</code>	98, 127, 137, 140, 144, 198, 221, 602
<code>\textup</code>	353		
<code>\textwidth</code>	34, 8, 19, 20, 86, 87, 285		
<code>\theappendixframe</code>	480		
<code>thebibliography</code>	533		
<code>\theenumi</code>	34		
<code>\theenumii</code>	35		
<code>\theenumiii</code>	36		
<code>\theenumiv</code>	37, 543		
<code>\thefigure</code>	150		
<code>\theframe</code>	476		
<code>\thepage</code>	6, 177, 419		

<code>\use:n</code>	48, 52, 83, 94, 118, 119, 123, 127, 133, 285, 393, 510	<code>\vbox_set_end:</code>	55, 74, 82, 87, 97, 271, 288, 375
<code>\use_ii:nn</code>	234	<code>\vbox_set_split_to_ht:NNn</code> .	108, 115
<code>\use_none:n</code>	191, 208	<code>\vbox_set_to_wd:Nnn</code>	29, 380
<code>\use_none:nnnn</code>	206, 208, 563	<code>\vbox_set_to_wd:Nnw</code>	38, 85, 264
<code>\usebox</code>	444	<code>\vbox_to_ht:nn</code>	60, 125, 272, 298
<code>\usecounter</code>	541	<code>\vbox_top:n</code>	74
<code>\UseHookWithArguments</code>	321, 324, 326, 329, 413	<code>\vbox_unpack:N</code>	185
<code>\UseInstance</code>	104, 134, 144, 194, 340, 392, 400, 449, 454, 464, 467	<code>\vbox_unpack_drop:N</code>	99, 128, 132
<code>\UseStructureName</code>	128, 143, 221	vcoffin commands:	
<code>\UseTaggingSocket</code> .	137, 138, 156, 249, 251	<code>\vcoffin_set:Nnn</code>	2
<code>\UseTemplate</code>	142, 146	<code>verse (env.)</code>	338
V		<code>\vfil</code>	275, 302, 334
<code>\value</code> ...	197, 284, 299, 300, 304, 305, 316	<code>\visible</code>	108
vbox commands:		<code>visibleenv (env.)</code>	117
<code>\vbox:n</code>	55	<code>\vspace</code>	32, 41, 66, 76
<code>\vbox_gset:Nn</code>	183	W	
<code>\vbox_set:Nn</code>	59, 117	<code>\widowpenalty</code>	548
<code>\vbox_set:Nw</code>	44, 49, 71, 75, 363	Y	
		<code>\year</code>	22