

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

Joseph Wright[†]

Released 2026-09-30

Contents

I	ltx-talk – Overall set up	1
1	ltx-talk implementation	1
1.1	Set up	1
1.2	Additions for expl3	2
1.3	Extra variants	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	6
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 v0.6.8, last revised 2026-09-30.

[†]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	25
1.5	Wallpaper	25
1.6	The <code>frame</code> environment	30
V	ltx-talk-frame – The structure of frames	35
1	ltx-talk-frame-structure implementation	35
1.1	Columns	35
1.2	Floats	37
1.3	Footnotes	39
VI	ltx-talk-mode – Modes	41
1	ltx-talk-mode implementation	41
VII	ltx-talk-overlay – Overlays	42
1	ltx-talk-overlay implementation	42
1.1	Utilities	42
1.2	Opacity utilities	43
1.3	Action commands and environments	43
1.4	Non-action commands and environments	48
1.5	Fixed-size areas	49
1.6	Adding overlays to existing commands	51
1.7	Overlay patching of third-party environments	53
VIII	ltx-talk-required – “Required” definitions	55
1	ltx-talk-required implementation	55
1.1	Standard design settings	55
1.2	List support	56
IX	ltx-talk-structure – Structural commands	57

1	ltx-talk-structure implementation	57
1.1	Frame title	57
1.2	Headings	59
1.3	Table of contents	63
1.4	Block environments	65
1.5	Lists	66
1.6	Theorems, <i>etc.</i>	69
1.7	Bibliographies	70
1.8	Supplementary material	70
X	ltx-talk-title – Title pages	72
1	ltx-talk-title implementation	72
	Index	76

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} {2026-09-30} {0.6.8}
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
30   { \__box_set_to_wd:NNnn #1 \tex_vbox:D }
31 \cs_gset_protected:Npn \vbox_set_top_to_wd:Nnn #1
32   { \__box_set_to_wd:NNnn #1 \tex_vtop:D }
33 \cs_gset_protected:Npn \__box_set_to_wd:NNnn #1#2#3#4
34   {
35     \tex_setbox:D #1 #2
36     {
37       \tex_hsize:D \__box_dim_eval:n {#3}
38       \color_group_begin: #4 \par \color_group_end:
39     }
40     \box_wd:N #1 \__box_dim_eval:n {#3}
41   }
42 \cs_gset_protected:Npn \vbox_set_to_wd:Nnw #1#2
43   {
44     \cs_set_protected:Npn \__box_set_to_wd:
45       { \box_wd:N #1 \__box_dim_eval:n {#2} }
46     \tex_setbox:D #1 \tex_vbox:D
47     \c_group_begin_token
48     \tex_hsize:D \__box_dim_eval:n {#2}
49     \group_insert_after:N \__box_set_to_wd:
50     \color_group_begin:
51   }

```

Some things from `xbox` that would be useful.

```

52 \cs_gset_protected:Npn \rule:nnn #1#2#3
53   {
54     \tex_vrule:D
55     height \dim_eval:n {#2} \exp_stop_f:
56     depth \dim_eval:n {#3} \exp_stop_f:
57     width \dim_eval:n {#1} \exp_stop_f:
58     \scan_stop:
59   }

```

1.3 Extra variants

```

60 \cs_generate_variant:Nn \hook_gput_code:nnn { nne }
61 \exp_args_generate:n { nVv }
62 \cs_generate_variant:Nn \dim_max:nn { v }
63 \cs_generate_variant:Nn \str_replace_all:Nnn { NnV }
64 \cs_generate_variant:Nn \vbox_to_ht:nn { v }

```

1.4 Scratch space

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

```

65 \cs_new_protected:Npn \__talk_tmp:w { }

```

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

`\l__talk_tmp_box`

```

66 \box_new:N \l__talk_tmp_box

```

(End of definition for \l__talk_tmp_box.)

\l__talk_tmp_clist

67 \clist_new:N \l__talk_tmp_clist

(End of definition for \l__talk_tmp_clist.)

\l__talk_tmp_tl

68 \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

69 \keys_define:nn { talk }

70 {

71 aspect-ratio .str_set:N =

72 \l__talk_aspect_ratio_str ,

73 font-size .dim_set:N =

74 \l__talk_fontsize_dim ,

75 frame-title-arg .bool_set:N =

76 \l__talk_frame_title_bool ,

77 handout .code:n =

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

79 handout .value_forbidden:n = true ,

80 mode .choices:nn =

81 { handout , projector }

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

83 supplementary-frames .choices:nn =

84 { appendix , in-place , omit }

85 { \str_set_eq:NN \l__talk_suppl_mode_str \l_keys_choice_str }

86 }

87 \str_new:N \l__talk_mode_str

88 \str_new:N \l__talk_suppl_mode_str

(End of definition for \l__talk_aspect_ratio_str and others.)

Scope for options.

89 \keys_define:nn { talk }

90 {

91 aspect-ratio .usage:n = load ,

92 font-size .usage:n = load ,

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

94 mode .usage:n = load ,

95 supplementary-frames .usage:n = load

96 }

Compatibility keys for classical font size setting.

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

98 {

99 \keys_define:nn { talk }

100 {

101 #1 .meta:n = { font-size = #1 } ,

102 #1 .value_forbidden:n = true ,

```

103         #1. usage:n          = load
104     }
105 }
    Initial values.
106 \keys_set:nn { talk }
107 {
108     aspect-ratio      = 16:9      ,
109     font-size         = 11pt      ,
110     frame-title-arg   = false     ,
111     mode              = projector ,
112     supplementary-frames = appendix
113 }
114 \ProcessKeyOptions [ talk ]

```

1.6 Setting up

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

```

115 \file_if_exist_input:nF { size \dim_to_decimal:n \l__talk_fontsize_dim .clo }
116 {
117     \file_input:n { size10.clo }
118     \RequirePackage { relsize }
119     \hook_gput_code:nne { begindocument } { talk }
120     { \exp_not:N \relsize { \fp_eval:n { \l__talk_fontsize_dim / 10pt } } }
121 }

```

\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.

```

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

```

```

147         bmargin    = 8mm ,
148         lmargin    = 10mm ,
149         rmargin    = 10mm ,
150         headheight = 10mm ,
151         headsep     = 2mm ,
152         footskip    = 6mm
153     ]
154     { geometry }
155 }

```

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

Turn off justification

```

156 \raggedright

```

1.7 Math support

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

```

157 \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.

```

158 \sys_if_engine_opentype:TF
159 {
160     \@ifpackageloaded { lua-font-config }
161     { \RequirePackage { mathtools } }
162     {
163         \RequirePackage { fontspec }
164         \RequirePackage { mathtools }
165         \sys_if_engine luatex:TF
166         {
167             \RequirePackage { lua-unicode-math }
168             \tagpdfsetup { math / mathml / luamml / load = true }
169         }
170         { \RequirePackage { unicode-math } }
171         \setmainfont { NewCMSans10-Regular.otf }
172         \setsansfont { NewCMSans10-Regular.otf }
173         \setmathfont { NewCMSansMath-Regular.otf }
174     }
175 }
176 {
177     \RequirePackage { sansmathfonts }
178     \RequirePackage [ nomath ] { lmodern }
179     \cs_set_eq:NN \rmdefault \sfdefault
180 }

```

1.9 Text scripts

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

```
\textsubscript@offset  Offset values as in ConTEXt, no additional spacing added after script items.
\textsubscript@space   181 \cs_set_nopar:Npn \textsubscript@offset { 0.48ex }
\textsuperscript@offset 182 \cs_set_nopar:Npn \textsubscript@space { }
\textsuperscript@space  183 \cs_set_nopar:Npn \textsuperscript@offset { 0.86ex }
                        184 \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.

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

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

A requirement.
186 \RequirePackage { hyperref }
187 \hypersetup { hidelinks }
```

1.11 Tagging

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

```
188 \tagpdfsetup
189 {
190   role / user-NS = ltx-talk          ,
191   role / new-tag = frame / Sect      ,
192   role / new-tag = frametitle / H4
193 }
194 </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:nNnN {#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     {
64         \__talk_slide_fill:
65         \__talk_slide_auxiii:
66     }
67 }
68 \cs_new_protected:Npn \__talk_slide_auxiii:
69 {
70     \box_use_drop:N \l__talk_slide_box
71     \par
72     \bool_if:NTF \g__talk_frame_tag_bool
73     { \__talk_slide_tag_on_end: }
74     { \__talk_slide_tag_off_end: }
75     \hbox_set_end:
76     \box_use_drop:N \l__talk_slide_box
77     \vbox_set_end:
78     \bool_if:NT \g__talk_slide_continue_bool
79     { \__talk_cnt_restore: }
80     \__talk_slide_output:
81 }
82 \cs_new_protected:Npn \__talk_slide_auxiv:n #1
83 {
84     \group_begin:
85     \cs_set:Npn \obeyedline { ^^J }
86     \use:e
87     {
88         \group_end:
89         \tl_retokenize:n {#1}
90     }
91 }

```

(End of definition for __talk_slide:nm 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.

```

92 \cs_new_protected:Npn \__talk_slide_init:
93 {
94     \tl_gclear:N \g__talk_onslide_tl
95     \int_gzero:N \g__talk_pauses_int
96     \tl_gclear:N \g__talk_frame_title_tl
97     \tl_gclear:N \g__talk_frame_subtitle_tl
98     \box_gclear:N \g__talk_footnote_box
99     \__talk_cnt_save:
100     \bool_set_eq:NN \l__talk_slide_continue_bool
101     \g__talk_slide_continue_bool

```

```

102 \__talk_if_overlay:VTF \l__talk_frame_tagging_str
103 { \bool_gset_true:N \g__talk_frame_tag_bool }
104 { \bool_gset_false:N \g__talk_frame_tag_bool }
105 \bool_gset_eq:NN \g__talk_slide_continue_bool
106 \l__talk_slide_continue_bool
107 }

```

(End of definition for __talk_slide_init:.)

__talk_slide_fill: Once the horizontal box carrying the tagging is closed, the content inside it is rigid, and so any stretchable glue in it (\vfill, the `stretch` alignment) would stay at its natural size. So a slide that is not being broken is re-set here, still inside that box, to the height available for it and with the alignment glue around it. Footnotes are added below the content in __talk_slide_output:, and so their height is not available: the standard \footnoterule takes up no net height, and the depth of the last footnote becomes the depth of the slide. (The depth of the last line of content needs no allowance either: the alignment glue ends the box, and so that depth is part of its height.)

```

108 \cs_new_protected:Npn \__talk_slide_fill:
109 {
110   \vbox_set_to_ht:Nnn \l__talk_slide_box
111   { \textheight - \box_ht:N \g__talk_footnote_box }
112   {
113     \use:c { __talk_slide_align_ \l__talk_frame_alignment_tl :n }
114     { \vbox_unpack_drop:N \l__talk_slide_box }
115   }
116 }

```

(End of definition for __talk_slide_fill:.)

__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.

```

117 \cs_new_protected:Npn \__talk_slide_break:
118 {
119   \box_set_eq:NN \l__talk_break_box \l__talk_slide_box
120   \vbox_set_split_to_ht:Nnn \l__talk_slide_box \l__talk_break_box
121   { \fp_to_dim:n { \l__talk_frame_break_fp * \textheight } }
122   \box_gset_eq:NN \g__talk_break_box \l__talk_break_box
123   \__talk_slide_auxiii:
124   \box_set_eq:NN \l__talk_break_box \g__talk_break_box
125   \bool_do_until:nn { \box_if_empty_p:N \l__talk_break_box }
126   {
127     \vbox_set_split_to_ht:Nnn \l__talk_slide_box \l__talk_break_box
128     { \fp_to_dim:n { \l__talk_frame_break_fp * \textheight } }
129     \vbox_set:Nn \l__talk_slide_box
130     { \hbox:n { \box_use_drop:N \l__talk_slide_box } }
131     \__talk_slide_output:
132   }
133 }

```

(End of definition for __talk_slide_break:.)

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

```

134 \cs_new_protected:Npn \__talk_slide_output:
135 {
136   \thispdfpagelabel { \int_use:N \g__talk_frame_int }
137   \vbox_to_ht:nn { \textheight }
138   {
139     \use:c { __talk_slide_align_ \l__talk_frame_alignment_tl :n }
140     { \vbox_unpack_drop:N \l__talk_slide_box }
141     \box_if_empty:NF \g__talk_footnote_box
142     {
143       \footnoterule
144       \vbox_unpack_drop:N \g__talk_footnote_box
145     }
146   }
147   \bool_if:NF \g__talk_slide_continue_bool
148   {
149     \clist_set_eq:NN \l__talk_tmp_clist \l_talk_frame_properties_clist
150     \clist_put_left:Nn \l__talk_tmp_clist { slides }
151     \clist_remove_duplicates:N \l__talk_tmp_clist
152     \property_record:ee { frame . \int_use:N \g__talk_frame_int }
153     { \l__talk_tmp_clist }
154   }
155   \clearpage
156 }

```

(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
157 \cs_new_protected:Npn \__talk_slide_align_bottom:n #1
158 {
159   \skip_vertical:n { Opt~plus~1fil }
160   #1
161   \skip_vertical:n { Opt }
162 }
163 \cs_new_protected:Npn \__talk_slide_align_center:n #1
164 {
165   \skip_vertical:n { Opt~plus~0.5fil }
166   #1
167   \skip_vertical:n { Opt~plus~0.5fil }
168 }
169 \cs_new_protected:Npn \__talk_slide_align_stretch:n #1
170 {
171   \skip_vertical:n { Opt }
172   #1
173   \skip_vertical:n { Opt }
174 }
175 \cs_new_protected:Npn \__talk_slide_align_top:n #1
176 {
177   \skip_vertical:n { Opt }
178   #1
179   \skip_vertical:n { Opt~plus~1fil }
180 }

```

(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.

```

181 \cs_new_protected:Npn \__talk_slide_tag_on_begin:
182 {
183   \tag_struct_begin:n { tag = frame }
184   \int_gset:Nn \g__talk_frame_struct_int { \tag_get:n { struct_num } }
185 }
186 \cs_new_protected:Npn \__talk_slide_tag_on_end:
187 {
188   \tag_struct_end:
189 }

```

(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.

```

190 \cs_new_protected:Npn \__talk_slide_tag_off_begin:
191 {
192   \tag_mc_begin:n { artifact }
193   \tag_suspend:n { frame }
194 }
195 \cs_new_protected:Npn \__talk_slide_tag_off_end:
196 {
197   \tag_resume:n { frame }
198   \tag_mc_end:
199 }

```

(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).

```

200 \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`.

```

201 \seq_new:N \g__talk_cnt_reset_seq
202 \seq_gset_from_clist:Nn \g__talk_cnt_reset_seq
203 {
204   equation      ,
205   footnote      ,
206   mpfootnote    ,
207   parentequation
208 }
209 \seq_map_inline:Nn \g__talk_cnt_reset_seq
210 {
211   \int_new:c { g__talk_saved_ #1 _int }
212   \int_gset_eq:cc { g__talk_saved_ #1 _int } { c@ #1 }
213 }

```

(End of definition for `\g__talk_cnt_reset_seq`.)

```

\__talk_cnt_save: A simple save-and-restore pair.
\__talk_cnt_restore: 214 \cs_new_protected:Npn \__talk_cnt_save:
215 {
216   \seq_map_inline:Nn \g__talk_cnt_reset_seq
217   { \int_gset_eq:cc { g__talk_saved_ ##1 _int } { c@ ##1 } }
218 }
219 \cs_new_protected:Npn \__talk_cnt_restore:
220 {
221   \seq_map_inline:Nn \g__talk_cnt_reset_seq
222   { \int_gset_eq:cc { c@ ##1 } { g__talk_saved_ ##1 _int } }
223 }

```

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

```

\@definecounter Track all counters for resetting.
\std@definecounter 224 \cs_new_eq:NN \std@definecounter \@definecounter
225 \cs_gset_protected:Npn \@definecounter #1
226 {
227   \std@definecounter {#1}
228   \int_new:c { g__talk_saved_ #1 _int }
229   \seq_gput_right:Nn \g__talk_cnt_reset_seq {#1}
230 }

```

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

1.3 Frame options

```

\l__talk_frame_alignment_tl
231 \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 232 \bool_new:N \l__talk_frame_suppl_bool
\l__talk_frame_break_fp 233 \keys_define:nn { talk / frame }
\l__talk_frame_name_str 234 {
\l__talk_frame_suppl_bool 235   action-spec .str_set:N =
\l__talk_frame_tagging_str 236   \l__talk_action_spec_str ,
237   auto-break .bool_set:N =
238   \l__talk_frame_break_bool ,
239   auto-break-coverage .fp_set:N =
240   \l__talk_frame_break_fp ,
241   label .meta:n =
242   { name = {#1} } ,
243   name .str_set:N =
244   \l__talk_frame_name_str ,
245   supplementary-frame .bool_set:N =
246   \l__talk_frame_suppl_bool ,
247   tag-slides .str_set:N =
248   \l__talk_frame_tagging_str ,
249   vertical-alignment .choices:nn =

```

```

250     { bottom , center , stretch , top }
251     {
252         \tl_set_eq:NN \l__talk_frame_alignment_tl
253         \l_keys_value_tl
254     }
255 }
256 \keys_set:nn { talk / frame }
257 {
258     action-spec          =          ,
259     auto-break           = false    ,
260     auto-break-coverage = 0.85     ,
261     name                  =          ,
262     supplementary-frame = false     ,
263     tag-slides           = n        ,
264     vertical-alignment  = center
265 }

```

(End of definition for \l__talk_action_spec_str and others.)

1.4 Tagging for headers

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

```

\__talk_header_tag_begin:e
\__talk_header_tag_end:
266 \cs_new_protected:Npn \__talk_header_tag_begin:n #1
267 {
268     \tag_resume:n { header }
269     \tag_mc_end:
270     \tag_struct_begin:n {#1}
271     \tag_mc_begin:n { }
272 }
273 \cs_generate_variant:Nn \__talk_header_tag_begin:n { e }
274 \cs_new_protected:Npn \__talk_header_tag_end:
275 {
276     \tag_mc_end:
277     \tag_struct_end:
278     \tag_mc_begin:n { artifact }
279     \tag_suspend:n { header }
280 }

```

(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
281 \NewTemplateType { footer-element } { 1 }
282 \DeclareTemplateInterface { footer-element } { talk } { 1 }
283 {
284     color          : tokenlist ,
285     font           : tokenlist = ,
286     left-hspace    : length = 0em ,
287     right-hspace   : length = 0em
288 }
289 \DeclareTemplateCode { footer-element } { talk } { 1 }
290 {

```

```

291     color          = \l__talk_footelem_color_tl ,
292     font           = \l__talk_footelem_font_tl ,
293     left-hspace    = \l__talk_footelem_left_skip ,
294     right-hspace   = \l__talk_footelem_right_skip
295   }
296   {
297     \tl_if_empty:NF {#1}
298     {
299       \hspace { \l__talk_footelem_left_skip }
300       \hbox:n
301       {
302         \tl_if_empty:NF \l__talk_footelem_color_tl
303         { \color_select:V \l__talk_footelem_color_tl }
304         \l__talk_footelem_font_tl
305         #1
306       }
307       \hspace { \l__talk_footelem_right_skip }
308     }
309   }
310   \DeclareInstance { footer-element } { date } { talk } { }
311   \DeclareInstance { footer-element } { author } { talk } { }
312   \DeclareInstance { footer-element } { title } { talk } { }
313   \DeclareInstance { footer-element } { subtitle } { talk } { }
314   \DeclareInstance { footer-element } { institute } { talk } { }
315   \DeclareInstance { footer-element } { framenumbers } { talk } { }
316   \DeclareInstance { footer-element } { totalframes } { talk } { }

```

(End of definition for \l__talk_footelem_left_skip and others.)

<pre> \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 </pre>	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. <pre> 317 \NewTemplateType { header } { 0 } 318 \DeclareTemplateInterface { header } { talk } { 0 } 319 { 320 background-color : tokenlist, 321 color : tokenlist = structure , 322 font : tokenlist = \normalfont , 323 height : length = \Gm@tmargin + \headsep , 324 left-hspace : skip = \Gm@lmargin , 325 print-frame-title : boolean = true , 326 right-hspace : skip = \Gm@rmargin 327 } 328 \DeclareTemplateCode { header } { talk } { 0 } 329 { 330 background-color = \l__talk_header_bg_tl , 331 color = \l__talk_header_fg_tl , 332 font = \l__talk_header_font_tl , 333 height = \l__talk_header_ht_dim , 334 left-hspace = \l__talk_header_left_skip , 335 print-frame-title = \l__talk_header_frametitle_bool , 336 right-hspace = \l__talk_header_right_skip 337 } </pre>
--	--

```

338 {
339   \noindent
340   \_talk_wallpaper_hrulerule:Nnn
341   \l\_talk_header_bg_tl
342   { \l\_talk_header_ht_dim - \headsep }
343   { \headsep }
344   \skip_horizontal:n { \l\_talk_header_left_skip }
345   \vbox_set_to_wd:Nnn \l\_talk_tmp_box
346   {
347     \paperwidth
348     - \l\_talk_header_left_skip
349     - \l\_talk_header_right_skip
350   }
351   {
352     \tl_if_empty:NF \l\_talk_header_fg_tl
353     { \color_select:V \l\_talk_header_fg_tl }
354     \l\_talk_header_font_tl
355     \bool_if:NT \l\_talk_header_frametitle_bool
356     {
357       \ExpandArgs { nnV }
358       \UseInstance { frametitle } { header }
359       \g\_talk_frame_title_tl
360     }
361   }
362   \dim_compare:nNnT { \box_ht:N \l\_talk_tmp_box + \box_dp:N \l\_talk_tmp_box }
363   > \l\_talk_header_ht_dim
364   { \msg_warning:nn { talk } { overheight-header } }
365   \box_use_drop:N \l\_talk_tmp_box
366   \skip_horizontal:n { \l\_talk_header_right_skip - \Gm@rmargin }
367 }
368 \DeclareInstance { header } { std } { talk } { }
369 \AddToHook { begindocument }
370 {
371   \DeclareInstanceCopy { header } { wallpaper } { std }
372   \EditInstance { header } { wallpaper }
373   { print-frame-title = false }
374 }

```

(End of definition for \l_talk_header_bg_tl and others.)

```

\l\_talk_footer_bg_tl
\l\_talk_footer_fg_tl
\l\_talk_footer_font_tl
\l\_talk_footer_order_clist
\l\_talk_footer_sep_tl
\l\_talk_footer_left_skip
\l\_talk_footer_right_skip

```

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

```

375 \NewTemplateType { footer } { 0 }
376 \DeclareTemplateInterface { footer } { talk } { 0 }
377 {
378   background-color : tokenlist ,
379   color             : tokenlist ,
380   element-order     : commalist ,
381   font              : tokenlist = \tiny ,
382   left-hspace       : length = \Gm@lmargin ,
383   right-hspace      : length = \Gm@rmargin ,
384   separator         : tokenlist = \hfil
385 }

```

```

386 \DeclareTemplateCode { footer } { talk } { 0 }
387 {
388     background-color = \l__talk_footer_bg_tl ,
389     color             = \l__talk_footer_fg_tl ,
390     element-order     = \l__talk_footer_order_clist ,
391     font              = \l__talk_footer_font_tl ,
392     left-hspace       = \l__talk_footer_left_skip ,
393     right-hspace      = \l__talk_footer_right_skip ,
394     separator         = \l__talk_footer_sep_tl
395 }
396 {
397     \noindent
398     \__talk_wallpaper_hruler:Nnn
399     \l__talk_footer_bg_tl
400     { \footskip }
401     { \Gm@bmargin - \footskip }
402     \skip_horizontal:n { \l__talk_footer_left_skip }
403     \vbox_set_to_wd:Nnn \l__talk_tmp_box
404     {
405         \paperwidth
406         - \l__talk_footer_left_skip
407         - \l__talk_footer_right_skip
408     }
409     {
410         \tl_if_empty:NF \l__talk_footer_fg_tl
411         { \color_select:V \l__talk_footer_fg_tl }
412         \l__talk_footer_font_tl
413         \clist_pop:NNT \l__talk_footer_order_clist \l__talk_tmp_tl
414         {
415             \ExpandArgs { nVv } \UseInstance { footer-element } \l__talk_tmp_tl
416             { @ \__talk_metadata_name:n { \l__talk_tmp_tl } }
417             \clist_map_inline:Nn \l__talk_footer_order_clist
418             {
419                 \tl_if_empty:cF { @ \__talk_metadata_name:n { ##1 } }
420                 {
421                     \l__talk_footer_sep_tl
422                     \ExpandArgs { nnv }
423                     \UseInstance { footer-element } {##1}
424                     { @ \__talk_metadata_name:n { ##1 } }
425                 }
426             }
427         }
428         \hfil
429     }
430     \box_use_drop:N \l__talk_tmp_box
431     \skip_horizontal:n { \l__talk_footer_right_skip - \Gm@rmargin }
432 }
433 \DeclareInstance { footer } { std } { talk } { }
434 \AddToHook { begindocument }
435 {
436     \DeclareInstanceCopy { footer } { wallpaper } { std }
437     \EditInstance { footer } { wallpaper }
438     { element-order = }
439 }

```

(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.

```

440 \cs_new:Npn \__talk_metadata_name:n #1
441 {
442   \tl_if_exist:cTF { @ short #1 }
443   { short #1 }
444   {#1}
445 }

```

(End of definition for __talk_metadata_name:n.)

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

```

446 \cs_new_protected:Npn \__talk_wallpaper_hrule:Nnn #1#2#3
447 {
448   \skip_horizontal:n { -\Gm@lmargin }
449   \tl_if_empty:NF #1
450   {
451     \group_begin:
452     \color_select:V #1
453     \rule:nnn { \paperwidth } {#2} {#3}
454     \skip_horizontal:n { -\paperwidth }
455     \group_end:
456   }
457 }

```

(End of definition for __talk_wallpaper_hrule:Nnn.)

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

```

458 \cs_set_nopar:Npn \ps@plain
459 {
460   \cs_set_nopar:Npn \@oddhead
461   {
462     \hfil
463   }
464   \cs_set_nopar:Npn \@oddfoot { }
465   \cs_set_eq:NN \@evenhead \@oddhead
466   \cs_set_eq:NN \@evenfoot \@oddfoot
467 }
468 \cs_set_nopar:Npn \ps@wallpaper
469 {
470   \cs_set_nopar:Npn \@oddhead
471   {
472     \UseInstance { header } { wallpaper }
473     \hfil
474   }
475   \cs_set_nopar:Npn \@oddfoot
476   {
477     \UseInstance { footer } { wallpaper }
478     \hfil

```

```

479     }
480     \cs_set_eq:NN \@evenhead \@oddhead
481     \cs_set_eq:NN \@evenfoot \@oddfoot
482   }
483   \cs_new_nopar:Npn \ps@talk
484   {
485     \cs_set_nopar:Npn \@oddhead
486     {
487       \UseInstance { header } { std }
488       \hfil
489     }
490     \cs_set_nopar:Npn \@oddfoot { \UseInstance { footer } { std } }
491     \cs_set_eq:NN \@evenhead \@oddhead
492     \cs_set_eq:NN \@evenfoot \@oddfoot
493   }
494   \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.

```
495 \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.

```
496 \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).

```
497 \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).

```
498 \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 499 \int_new:N \g__talk_frame_int
\theframe 500 \cs_new_eq:NN \c@frame \g__talk_frame_int
\@framenum 501 \cs_new:Npn \theframe { \@arabic \c@frame }
502 \cs_new:Npn \@framenum { \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 503 \int_new:N \g__talk_appendixframe_int
\theappendixframe 504 \cs_new_eq:NN \c@appendixframe \g__talk_appendixframe_int
\@appendixframenumber 505 \cs_new:Npn \theappendixframe { \@arabic \c@appendixframe }
506 \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.

```
507 \property_new:nnnn { frames } { shipout } { -1 }
508   { \int_eval:n { \g__talk_frame_int - \g__talk_appendixframe_int } }
509 \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.

```
510 \property_new:nnnn { totalframes } { shipout } { -1 }
511   { \int_use:N \g__talk_frame_int }
512 \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.

```
513 \property_new:nnnn { appendixframes } { shipout } { -1 }
514   { \int_use:N \g__talk_appendixframe_int }
515 \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.

```
516 \clist_new:N \l_talk_frame_properties_clist
```

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

```
517 \AddToHook { enddocument / afterlastpage }
518   {
519     \property_record:nn { lastpage }
520     { appendixframes , frames , totalframes }
521   }
```

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

```
522 \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
523 \cs_new_protected:Npn \__talk_frame_process:nn #1#2
524   {
525     \__talk_decode_parse:n {#1}
526     \bool_if:NT \l__talk_decode_mode_bool
527       { \__talk_frame_process_aux:nn {#1} {#2} }
528   }
529 \cs_new_protected:Npn \__talk_frame_process_aux:nn #1#2
530   {
531     \int_gincr:N \g__talk_frame_int
532     \bool_if:NT \l__talk_appendix_bool
533       { \int_gincr:N \g__talk_appendixframe_int }
534     \bool_set_true:N \l__talk_frame_bool
535     \str_if_empty:NF \l__talk_frame_name_str
```

```

536     { \__talk_frame_save:en { \l__talk_frame_name_str } {#2} }
537 \bool_if:NTF \l__talk_frame_suppl_bool
538 {
539     \str_case:Vn \l__talk_suppl_mode_str
540     {
541         { appendix }
542         {
543             \int_gincr:N \g__talk_frame_suppl_int
544             \__talk_frame_save:en
545             { appendix . \int_use:N \g__talk_frame_suppl_int }
546             {#2}
547         }
548         { in-place } { \__talk_slide:nn {#1} {#2} }
549     }
550 }
551 { \__talk_slide:nn {#1} {#2} }
552 }
553 \cs_new_protected:Npn \__talk_frame_save:nn #1#2
554 {
555     \tl_if_exist:cT { g__talk_frame_ #1 _tl }
556     { \msg_error:nnn { talk } { duplicate-frame-name } {#1} }
557     \tl_clear_new:c { g__talk_frame_ #1 _tl }
558     \tl_gset:ce { g__talk_frame_ #1 _tl }
559     {
560         {
561             \bool_if:NTF \l__talk_frame_verb_bool
562             { true }
563             { false }
564         }
565         { \exp_not:n {#2} }
566     }
567 }
568 \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
frame* document and code levels is still open.

```

569 \bool_if:NTF \l__talk_frame_title_bool
570 {
571     \RenewDocumentEnvironment { frame }
572     { D <> { all } = { action-spec } 0 { } +m +b }
573     {
574         \str_clear:N \l__talk_frame_name_str
575         \keys_set:nn { talk / frame } {#2}
576         \bool_set_false:N \l__talk_frame_verb_bool
577         \__talk_frame_process:nn {#1} { \frametitle {#3} #4 }
578     }
579     { }
580 \NewDocumentEnvironment { frame* }
581 { D <> { all } = { action-spec } 0 { } +m c }
582 {
583     \str_clear:N \l__talk_frame_name_str

```

```

584     \keys_set:nn { talk / frame } {#2}
585     \bool_set_true:N \l__talk_frame_verb_bool
586     \tl_gset:Nn \g__talk_frame_title_tl {#3}
587     \exp_args:Nne \__talk_frame_process:nn {#1}
588     { \tl_to_str:n { \frametitle } \exp_not:n { {#3} #4 } }
589   }
590   { }
591 }
592 {
593   \RenewDocumentEnvironment { frame }
594   { !D <> { all } = { action-spec } !O { } +b }
595   {
596     \str_clear:N \l__talk_frame_name_str
597     \keys_set:nn { talk / frame } {#2}
598     \bool_set_false:N \l__talk_frame_verb_bool
599     \__talk_frame_process:nn {#1} {#3}
600   }
601   { }
602   \NewDocumentEnvironment { frame* }
603   { !D <> { all } = { action-spec } !O { } c }
604   {
605     \str_clear:N \l__talk_frame_name_str
606     \keys_set:nn { talk / frame } {#2}
607     \bool_set_true:N \l__talk_frame_verb_bool
608     \__talk_frame_process:nn {#1} {#3}
609   }
610   { }
611 }

```

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

```

\__talk_frame_reuse:nn Reusing a frame is largely a question of passing the correct stored information along after
\__talk_frame_reuse_aux:nn setting the flag for re-scanning.
\__talk_frame_reuse_aux:nnn
  \reuseframe
  \againframe
612 \cs_new_protected:Npn \__talk_frame_reuse:nn #1#2
613   {
614     \tl_if_exist:cTF { g__talk_frame_ #2 _tl }
615     {
616       \exp_args:Nv \__talk_frame_reuse_aux:nn
617       { g__talk_frame_ #2 _tl } {#1}
618     }
619     { \msg_error:nnn { talk } { unknown-frame-name } {#2} }
620   }
621 \cs_new_protected:Npn \__talk_frame_reuse_aux:nn #1#2
622   { \__talk_frame_reuse_aux:nnn #1 {#2} }
623 \cs_new_protected:Npn \__talk_frame_reuse_aux:nnn #1#2#3
624   {
625     \use:c { bool_set_ #1 :N } \l__talk_frame_verb_bool
626     \__talk_frame_process:nn {#3} {#2}
627   }
628 \NewDocumentCommand \reuseframe { D <> { all } = { action-spec } O { } m }
629   {
630     \group_begin:
631     \keys_set:nn { talk / frame } {#2}
632     \str_clear:N \l__talk_frame_name_str

```

```

633     \_talk_frame_reuse:nn {#1} {#3}
634   \group_end:
635 }
636 \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.

```

637 \AddToHook { enddocument }
638 {
639   \int_compare:nNnT \g__talk_frame_suppl_int > 0
640   {
641     \bool_if:NF \l__talk_appendix_bool
642     { \appendix }
643     \int_step_inline:nn \g__talk_frame_suppl_int
644     { \_talk_frame_reuse:nn { } { appendix . #1 } }
645   }
646 }
647 \msg_new:nnnn { talk } { overheight-header }
648 { Overheight~header~on~frame~ \int_use:N \g__talk_frame_int ! }
649 {
650   Typesetting~the~frame~header,~there~was~too~much~vertical~space~used.
651   You~likely~need~to~adjust~the~overall~space~available~for~frame~titles.
652 }
653 \msg_new:nnnn { talk } { unknown-frame-name }
654 { Unknown~frame~name~"#1"! }
655 {
656   You~asked~to~reuse~the~contents~of~a~frame~with~the~name~"#1",~
657   but~that~name~was~never~defined.
658 }
659 \msg_new:nnnn { talk } { duplicate-frame-name }
660 { Duplicate~frame~name~"#1"! }
661 {
662   You~asked~to~save~the~contents~of~this~frame~with~the~name~"#1",~
663   but~that~name~has~already~been~used.
664 }
665 </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.

```





```

```

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

```

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

```

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

```

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

The spacing values needed for the standard function.

```

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

```

`\@caption` This is a copy of the kernel version of the function, but with writing to the list of whatever file removed. It is very likely this needs to be reworked as a template, but that will likely come from the kernel.

```

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

```

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

1.3 Footnotes

`\g__talk_footnote_box` Holds footnotes as they are constructed.

```

171 \box_new:N \g__talk_footnote_box

```

(End of definition for \g__talk_footnote_box.)

`\g__talk_footnote_overlay_seq` For tracking the overlays to apply.

```

172 \seq_new:N \g__talk_footnote_overlay_seq

```

(End of definition for \g__talk_footnote_overlay_seq.)

\stdfootnote

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

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

\footnote Sort-of overlay aware!

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

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

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

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

\@makefntext Use a copy of the standard setup.

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

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

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

Part VI

ltx-talk-mode – Modes

1 ltx-talk-mode implementation

Start the DocStrip guards.

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

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

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

(End of definition for __talk_mode:nT.)

`\mode`

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

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

```
13 </class>
```

Part VII

ltx-talk-overlay — Overlays

1 ltx-talk-overlay implementation

Start the DocStrip guards.

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

1.1 Utilities

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

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

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

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

\l__talk_shuffle_skip For tracking.

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

(End of definition for \l__talk_shuffle_skip.)

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

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

```

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

```

(End of definition for __talk_shuffle_skip:n.)

1.2 Opacity utilities

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

```

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

```

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

1.3 Action commands and environments

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

```

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

```

(End of definition for __talk_action_alert:N.)

```

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

```

```

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

```

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

```

\_talk_action_only:N
\_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
\visible
108 \clist_map_inline:nn { invisible , uncover , 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 \ExpandArgs { nnee } \NewDocumentEnvironment { #1 env }
118   { > { \__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
__talk_onslide:n tabular cells.

```

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 { @files }
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 LuaTeX due to restrictions on Xform structures in pdfTeX. Here, we apply the collection code to the commands not the environment forms to deal with “direct” usage.

```

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 \endclass

```

Part IX

ltx-talk-structure – Structural commands

1 ltx-talk-structure implementation

Start the DocStrip guards.

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

1.1 Frame title

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

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

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

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

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

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

Here, we want to be able to apply a color to the frame title but also to know where the baseline of the first line is. That means worrying about whatsits: first box up just the text, then add around it the spacing and color, before adjusting the box reference point.

```

30 {
31   \noindent
32   \vbox_set_top:Nn \l__talk_tmp_box
33   {
34     \l__talk_frametitle_font_tl
35     \tl_if_blank:nF {#1}
36     { \__talk_frame_title:n {#1} }
37   }
38   \use:e
39   {
40     \vbox_set:Nn \exp_not:N \l__talk_tmp_box
41     {
42       \exp_not:N \vspace
43       { \exp_not:N \l__talk_frametitle_before_skip }
44       \group_begin:
45       \tl_if_empty:NF \l__talk_frametitle_color_tl
46       {
47         \color_select:n
48         { \exp_not:N \l__talk_frametitle_color_tl }
49       }
50       \vbox_unpack:N \exp_not:N \l__talk_tmp_box
51       \group_end:
52       \exp_not:N \vspace
53       { \exp_not:N \l__talk_frametitle_after_skip }
54     }
55     \box_set_ht:Nn \exp_not:N \l__talk_tmp_box
56     {
57       \dim_eval:n
58       { \box_ht:N \l__talk_tmp_box + \l__talk_frametitle_before_skip }
59     }
60     \box_set_dp:Nn \exp_not:N \l__talk_tmp_box
61     {
62       \dim_eval:n
63       { \box_dp:N \l__talk_tmp_box + \l__talk_frametitle_after_skip }
64     }
65   }
66   \box_use_drop:N \l__talk_tmp_box
67 }
68 \DeclareInstance { frametitle } { header } { talk } { }
69 \cs_new_protected:Npn \__talk_frame_title:n #1
70 {
71   \bool_if:NTF \g__talk_frame_tag_bool
72   { \__talk_frame_title_tagged:n }
73   { \use:n }
74   {#1}
75 }
76 \cs_new_protected:Npn \__talk_frame_title_tagged:n #1
77 {
78   \__talk_header_tag_begin:e
79   {
80     firstkid = true ,

```

```

81     parent    = \int_use:N \g__talk_frame_struct_int ,
82     tag       = frametitle ,
83     title     = { \text_purify:n { \g__talk_frame_title_tl } } ,
84   }
85   \group_begin:
86     \tagpdfparaOff
87     #1
88   \group_end:
89   \__talk_header_tag_end:
90 }

```

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

1.2 Headings

```

\l__talk_section_tl  Two versions of the data store: one set locally (but at the top level) for general use, one
\g__talk_section_tl  set (and more importantly cleared) globally to allow insertion in the header area just
\l__talk_subsection_tl once per name.
\g__talk_subsection_tl
\l__talk_subsubsection_tl
\g__talk_subsubsection_tl
91 \tl_new:N \l__talk_section_tl
92 \tl_new:N \g__talk_section_tl
93 \tl_new:N \l__talk_subsection_tl
94 \tl_new:N \g__talk_subsection_tl
95 \tl_new:N \l__talk_subsubsection_tl
96 \tl_new:N \g__talk_subsubsection_tl

```

(End of definition for \l__talk_section_tl and others.)

```

\l__talk_sec_bookmark_tl Here, we set up the same keyval interface as used by the kernel (see latex-lab-sec-template.
\l__talk_sec_label_tl    dtx)
\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
97 \keys_define:nn { talk / heading }
98   {
99     bookmark .tl_set:N =
100       \l__talk_sec_bookmark_tl ,
101     label .code:n =
102       { \tl_put_right:N \l__talk_sec_label_tl { \label {#1} } } ,
103     nameref .tl_set:N =
104       \l__talk_sec_nameref_tl ,
105     numbered .bool_set:N
106       = \l__talk_sec_numbered_bool ,
107     numbered .default:n
108       = true ,
109     unnumbered .bool_set_inverse:N
110       = \l__talk_sec_numbered_bool ,
111     unnumbered .default:n
112       = true ,
113     quote .tl_set:N =
114       \l__talk_sec_quote_tl ,
115     running .tl_set:N =
116       \l__talk_sec_running_tl ,
117     shorttitle .meta:n =
118       {
119         bookmark = {#1} ,
120         nameref = {#1} ,
121         running = {#1} ,

```

```

122         toc      = {#1}
123     } ,
124     subtitle .tl_set:N =
125         \l__talk_sec_subtitle_tl ,
126     toc .tl_set:N =
127         \l__talk_sec_toc_tl
128 }
129 \tl_new:N \l__talk_sec_label_tl

```

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

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

```

130 \newcounter { section }
131 \newcounter { subsection } [ section ]
132 \newcounter { subsubsection } [ subsection ]
133 \seq_gremove_all:Nn \g__talk_cnt_reset_seq { section }
134 \seq_gremove_all:Nn \g__talk_cnt_reset_seq { subsection }
135 \seq_gremove_all:Nn \g__talk_cnt_reset_seq { subsubsection }
136 \cs_gset:Npn \thesection { \@arabic {c@section} }
137 \cs_gset:Npn \thesubsection { \thesection . \@arabic {c@subsection} }
138 \cs_gset:Npn \thesubsubsection { \thesubsection . \@arabic {c@subsubsection} }

```

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

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

```

139 \seq_set_from_clist:Nn \l_tmpa_seq
140 { section , subsection , subsubsection }
141 \seq_map_indexed_inline:Nn \l_tmpa_seq
142 {
143     \use:e
144     {
145         \NewDocumentCommand \exp_not:c { insert #2 } { }
146         {
147             \exp_not:N \tl_use:N
148             \exp_not:c { l__talk_ #2 _tl }
149         }
150         \NewDocumentCommand \exp_not:c {#2}
151         { s D <> { all } = { shorttitle } 0 {##4} m }
152         {
153             \exp_not:N \bool_if:NF \exp_not:N \l__talk_frame_bool
154             {
155                 \__talk_if_overlay:nT {##2}
156                 { \exp_not:c { __talk_head_ #1 :Nnn } ##1 {##3} {##4} }
157             }
158         }
159     }

```

```

159 \cs_new_protected:Npn \exp_not:c { __talk_head_ #1 :Nnn } ##1##2##3
160 {
161   \exp_not:N \refstepcounter {#2}
162   \UseTaggingSocket { sec / end } { \use:c { toplevel@ #2 } }
163   \UseTaggingSocket { sec / begin }
164   {
165     { \use:c { toplevel@ #2 } }
166     {
167       tag =
168       \exp_not:N \UseStructureName
169       { sec / \use:c { toplevel@ #2 } }
170     }
171   }
172   \tl_set:Nn \exp_not:c { l__talk_ #2 _tl } {##3}
173   \keys_set:nn { talk / heading } { shorttitle = {##3} , ##2 }
174   \exp_not:N \bool_if:NT ##1
175   {
176     \tl_clear:N \exp_not:N \l__talk_sec_bookmark_tl
177     \tl_clear:N \exp_not:N \l__talk_sec_running_tl
178     \tl_clear:N \exp_not:N \l__talk_sec_toc_tl
179     \bool_set_false:N \exp_not:N \l__talk_sec_numbered_bool
180   }
181   \UseTaggingSocket { talk / sec / title } {#2}
182   \str_if_eq:nnT {#2} { section }
183   { \tl_clear:N \exp_not:N \l__talk_subsection_tl }
184   \str_if_eq:nnF {#2} { subsection }
185   { \tl_clear:N \exp_not:N \l__talk_subsubsection_tl }
186   \exp_not:N \__talk_head_marks:nnVVV {#2} {#1}
187   \exp_not:N \l__talk_sec_toc_tl
188   \exp_not:N \l__talk_sec_bookmark_tl
189   \exp_not:N \l__talk_sec_nameref_tl
190   \hook_use:n { #2 / begin }
191 }
192 \hook_new:n { #2 / begin }
193 }
194 }

```

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

```

\addcontentslinebookmark Patches as in latex-lab-sec-template.dtx.
\addcontentslinebookmarkOff 195 \providecommand \addcontentslinebookmark [ 3 ] { }
\addcontentslinebookmarkOn 196 \providecommand \addcontentslinebookmarkOff { }
197 \providecommand \addcontentslinebookmarkReset { }
198 \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
199 \cs_new_protected:Npn \__talk_head_marks:nnnnn #1#2#3#4#5
200 {
201   \tl_set:Nn \@currentlabelname {#5}
202   \tl_if_blank:nTF {#3}
203   {

```

```

204     \tl_if_blank:nF {#4}
205     {
206         \__talk_head_marks_aux:Nnnn
207         \addcontentslinebookmark {#1} {#2} {#4}
208     }
209 }
210 {
211     \tl_if_blank:nT {#4}
212     { \addcontentslinebookmarkOff }
213     \__talk_head_marks_aux:Nnnn
214     \addcontentsline {#1} {#2} { \texorpdfstring {#3} {#4} }
215 }
216 }
217 \cs_generate_variant:Nn \__talk_head_marks:nnnnn { nnVVV }
218 \cs_new_protected:Npn \__talk_head_marks_aux:Nnnn #1#2#3#4
219 {
220     #1 { toc } {#2}
221     {
222         \int_compare:nNnF {#3} > { \value { secnumdepth } }
223         { \protect \numberline { \use:c { the #2 } } }
224         #4
225     }
226 }

```

(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

```

227 \NewDocumentCommand \appendixmarker { }
228 {
229     \bool_if:NTF \l__talk_appendix_bool
230     { \cs_gset_eq:NN \contentsline \__talk_saved_contentline:nnnn }
231     { \cs_gset_eq:NN \contentsline \use_none:nnnn }
232 }
233 \cs_new_eq:NN \__talk_saved_contentline:nnnn \use_none:nnnn
234 \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

```

235 \NewTaggingSocket { talk / sec / title } { 1 }
236 \NewTaggingSocketPlug { talk / sec / title } { default }
237 {
238     \exp_args:Ne \__talk_head_tag:nn
239     { \text_purify:v { l__talk_ #1 _ tl } } {#1}
240 }
241 \cs_new_protected:Npn \__talk_head_tag:nn #1#2
242 {
243     \tag_struct_begin:e
244     {
245         tag =
246         \UseStructureName { sec / \use:c { toplevel@ #2 } / title } ,
247         title = {#1} ,
248         actualtext = {#1} ,

```

```

249     }
250     \tag_struct_end:
251   }
252   \AssignTaggingSocketPlug { talk / sec / title } { default }

```

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

1.3 Table of contents

`\@starttoc` The standard kernel implementation here deliberately overwrites the file as soon as it's read. That's no good for us as the table of contents can be read multiple times. So we modify the code: we start from the tagging-aware version (this may need to be revisited). We retain the L^AT_EX 2_ε code as much as possible.

```

253 \cs_gset_protected:Npn \@starttoc #1
254 {
255   \begingroup
256   \makeatletter
257   \bool_if:NTF \g__talk_frame_tag_bool
258     { \@starttoc@aux@nn }
259     { \use_i:nn }
260     { #1 } { \input { \jobname . #1 } }
261   \legacy_if:nT { @filesw }
262   {
263     \AddToHook { enddocument / afterlastpage }
264     {
265       \expandafter \newwrite \csname tf@ #1 \endcsname
266       \immediate \openout \csname tf@ #1 \endcsname \jobname .#1 \relax
267     }
268   }
269   \@nobreakfalse
270   \endgroup
271 }
272 \cs_new_protected:Npn \@starttoc@aux@nn #1#2
273 {
274   \UseTaggingSocket { toc / starttoc / before } { #1 }
275   #2
276   \UseTaggingSocket { toc / starttoc / after } { #1 }
277 }

```

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

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

```

278 \NewDocumentCommand \tableofcontents { 0 { } }
279 {
280   \group_begin:
281   \@starttoc { toc }
282   \group_end:
283 }

```

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

```

\l@section Initial hard-coded versions to be templated once we have some other effects also working.
\l@subsection We may need to look at this “higher up” as we will need to know the section numbers.
\l@subsubsection
\__talk_toc_aux:nnnn 284 \cs_new_protected:Npn \l@section #1#2
\__talk_toc_dest:n 285 { \__talk_toc_aux:nnnn { 1 } { \bfseries \color { structure } } {#1} {#2} }
\__talk_toc_dest:w 286 \cs_new_protected:Npn \l@subsection #1#2
\__talk_toc_level:nnnn 287 {
288 \__talk_toc_aux:nnnn
289 { 2 }
290 {
291 \skip_set:Nn \leftskip { 2em }
292 \color_ensure_current:
293 }
294 {#1} {#2}
295 }
296 \cs_new_protected:Npn \l@subsubsection #1#2
297 {
298 \__talk_toc_aux:nnnn
299 { 3 }
300 {
301 \skip_set:Nn \leftskip { 4em }
302 \color_ensure_current:
303 \footnotesize
304 }
305 {#1} {#2}
306 }
307 \cs_new_protected:Npn \__talk_toc_aux:nnnn #1#2#3#4
308 {
309 \int_compare:nNnTF { \value { section } } < 1
310 { \use:n }
311 { \__talk_toc_dest:n }
312 { \__talk_toc_level:nnnn {#1} {#2} {#3} {#4} }
313 }

```

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.

```

314 \cs_new_protected:Npn \__talk_toc_dest:n
315 {
316 \exp_after:wN \__talk_toc_dest:w \@contentsline@destination
317 . 0 . 0 . 0 . \q_stop
318 }
319 \cs_new_protected:Npn \__talk_toc_dest:w #1 . #2 . #3 . #4 . #5 \q_stop #6
320 {
321 \bool_lazy_or:nnTF
322 {
323 \bool_lazy_and_p:nn
324 { \int_compare_p:nNn { \value { section } } = {#2} }
325 { \int_compare_p:nNn { \value { subsection } } = 0 }
326 }
327 {
328 \bool_lazy_and_p:nn
329 { \int_compare_p:nNn { \value { section } } = {#2} }
330 { \int_compare_p:nNn { \value { subsection } } = {#3} }
331 }

```

```

332     {#6}
333     {
334         \opacity_begin:n { 0.2 }
335         #6
336         \opacity_end:
337     }
338 }
339 \cs_new_protected:Npn \__talk_toc_level:nnnn #1#2#3#4
340 {
341     \int_compare:nNf {#1} > { \value { tocdepth } }
342     {
343         \group_begin:
344         \noindent
345         #2
346         \UseHookWithArguments { contentsline / text / before } { 4 }
347         {#1} {#3} {#4} { \@contentsline@destination }
348         #3
349         \UseHookWithArguments { contentsline / text / after } { 4 }
350         {#1} {#3} {#4} { \@contentsline@destination }
351         \UseHookWithArguments { contentsline / page / before } { 4 }
352         {#1} {#3} {#4}
353         { \@contentsline@destination }
354         \UseHookWithArguments { contentsline / page / after } { 4 }
355         {#1} {#3} {#4}
356         { \@contentsline@destination }
357         \par
358         \group_end:
359         \vfil
360     }
361 }

```

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

```

362 \setcounter { tocdepth } { 2 }

```

1.4 Block environments

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

```

quote (env.) 363 \NewDocumentEnvironment { description } { } { } { } { }
quotation (env.) 364 \NewDocumentEnvironment { quote } { } { } { } { }
verse (env.) 365 \NewDocumentEnvironment { quotation } { } { } { } { }
stdquote (env.) 366 \NewDocumentEnvironment { verse } { } { } { } { }
stdquotation (env.) 367 \AddToHook { begindocument / before }
stdverse (env.) 368 {
369     \clist_map_inline:nn { quote , quotation , verse }
370     {
371         \NewEnvironmentCopy { std #1 } {#1}
372         \RenewDocumentEnvironment {#1} { d <> !0 { } }
373         {
374             \__talk_action_begin:n {##1}
375             \begin { std #1 } [ {##2} ]
376             \ignorespaces
377         }
378         {
379             \end { std #1 }

```

```

380         \_talk_action_end:
381     }
382 }
383 }

block (env.)

384 \NewDocumentEnvironment { block } { d <> m }
385 {
386     \_talk_action_begin:n {#1}
387     \par
388     \vbox_set:Nw \l__talk_tmp_box
389     \group_begin:
390         \medskip
391         \leavevmode
392         \normalfont \large \bfseries
393         \color { structure }
394         #2
395     \par
396     \medskip
397 \group_end:
398 }
399 {
400     \vbox_set_end:
401     \box_use:N \l__talk_tmp_box
402     \par
403     \_talk_action_end:
404 }

```

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.

```

405 \AddToHook { begindocument / before }
406 {
407     \NewCommandCopy \stditem \item
408     \RenewDocumentCommand \item { d <> o }
409     {
410         \IfNoValueTF {#2}
411         { \stditem }
412         { \stditem [ {#2} ] }
413         \IfNoValueTF {#1}
414         {
415             \exp_after:wN \_talk_item_parse_spec:w
416             \l__talk_action_spec_str < all > \q_stop
417         }
418         { \_talk_item_parse_spec:n {#1} }
419     }
420 }

```

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.

```

421 \cs_new_protected:Npn \__talk_item_parse_spec:w #1 < #2 > #3 \q_stop
422 { \__talk_item_parse_spec:n {#2} }
423 \cs_new_protected:Npn \__talk_item_parse_spec:n #1
424 {
425   \bool_lazy_or:nnF
426     { \tl_if_blank_p:n {#1} }
427     { \str_if_eq_p:nn {#1} { all } }
428   {
429     \tl_set:Nx \l__talk_list_end_tl
430       {
431         \exp_not:N \int_compare:nNnT \tex_currentgrouplevel:D =
432           { \int_eval:n { \tex_currentgrouplevel:D + 1 } }
433         {
434           \__talk_action_end:
435           \tl_clear:N \exp_not:N \l__talk_list_end_tl
436         }
437       }
438     \__talk_action_begin:n {#1}
439   }
440 }

```

(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`

```

441 \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.

```

442 \cs_if_exist:NTF \l__block_topsepadd_skip
443 {
444   \cs_gset_protected:Npn \__block_inter_item:
445   {
446     \legacy_if:nT { @inlabel }
447     { \indent \par }
448     \mode_if_horizontal:T
449     {
450       \__block_skip_remove_last:
451       \__block_skip_remove_last:
452       \par
453     }
454     \l__talk_list_end_tl
455     \__kernel_list_item_end:
456     \__kernel_list_item_begin:
457     \addpenalty \@itempenalty
458     \addvspace \itemsep
459   }

```

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

```

460 \cs_gset_protected:Npe \BlockEnvEnd
461 {
462   \exp_not:n
463   { \__block_debug_typeout:n { blockenv~common~ending \on@line } }
464   \cs_if_exist:NTF \l__block_transparent_level_bool
465   {
466     \exp_not:N \bool_if:NF
467     \exp_not:N \l__block_transparent_level_bool
468   }
469   {
470     \exp_not:N \bool_if:NT
471     \exp_not:N \l__block_level_incr_bool
472   }
473   { \int_gdecr:N \exp_not:N \g_block_nesting_depth_int }
474   \exp_not:n
475   {
476     \legacy_if:nT { @inlabel }
477     {
478       \mode_leave_vertical:
479       \legacy_if_gset_false:n { @inlabel }
480     }
481     \__block_if_list:T
482     { \legacy_if:nT { @newlist } { \@noitemerr } }
483     \mode_if_horizontal:TF
484     {
485       \__block_skip_remove_last:
486       \__block_skip_remove_last:
487       \par
488     }
489     { \@inmatherr { \end { \@currenvir } } }
490     \l__talk_list_end_tl
491     \__kernel_displayblock_end:
492     \__block_if_list:T { \legacy_if_gset_false:n { @newlist } }
493     \legacy_if_gset_false:n { @nobreak }
494     \legacy_if:nF { @noparlist }
495     {
496       \__block_skip_set_to_last:N \l_tmpa_skip
497       \dim_compare:nNnT \l_tmpa_skip > \c_zero_dim
498       {
499         \skip_vertical:n { - \l_tmpa_skip }
500         \skip_vertical:n
501         { \l_tmpa_skip + \parskip - \@outerparskip }
502       }
503       \addpenalty \@endparpenalty
504       \addvspace \l__block_topsepadd_skip
505     }
506     \socket_use:n { block / endpe }
507   }
508 }
509 }

```

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

The branch where `\l__block_topsepad_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.

```

510 {
511   \AddToHook { item / after }
512   { \l__talk_list_end_tl }
513 }

```

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

```

description (env.) 514 \AddToHook { begindocument / before }
515 {
516   \clist_map_inline:nn { itemize , enumerate , description }
517   {
518     \RenewDocumentEnvironment {#1} { = { action-spec } !o }
519     { \SimpleBlockEnv {#1} {##1} }
520     { \BlockEnvEnd }
521   }
522 }

```

And add the structural color to item labels.

```

523 \AddToHook { begindocument / before }
524 {
525   \EditInstance { item } { basic }
526   { label-format = \color { structure } #1 }
527   \EditInstance { item } { description }
528   { label-format = \normalfont \bfseries \color { structure } #1 }
529 }

```

`\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.

```

530 \keys_define:nn { template / block / std }
531 { 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.

```

532 \NewCommandCopy \stdnewtheorem \newtheorem
533 \RenewDocumentCommand \newtheorem { s m o m o }
534 {
535   \use:e
536   {
537     \exp_not:N \stdnewtheorem
538     \IfBooleanT #1 { * }
539     { \exp_not:n {#2} }
540     \IfNoValueF {#3} { \exp_not:n { [ {#3} ] } }
541     { \exp_not:n {#4} }

```

```

542         \IfNoValueF {#5} { \exp_not:n { [ {#5} ] } }
543     }
544     \NewEnvironmentCopy { std #2 } {#2}
545     \RenewDocumentEnvironment {#2} { D <> { all } o }
546     {
547         \__talk_action_begin:n {##1}
548         \IfNoValueTF {##2}
549         { \begin { std #2 } }
550         { \begin { std #2 } [ {##2} ] }
551         \ignorespaces
552     }
553     {
554         \end { std #2 }
555         \__talk_action_end:
556     }
557 }

```

(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.

```

558 \NewDocumentEnvironment { thebibliography } { m }
559 {
560     \list { \@biblabel { \@arabic \c@enumiv } }
561     {
562         \settowidth \labelwidth { \@biblabel {#1} }
563         \leftmargin \labelwidth
564         \advance \leftmargin \labelsep
565         \@openbib@code
566         \usecounter { enumiv }
567         \let \p@enumiv \@empty
568         \renewcommand \theenumiv { \@arabic \c@enumiv }
569     }
570     \sloppy
571     \clubpenalty 4000
572     \@clubpenalty \clubpenalty
573     \widowpenalty 4000
574     \sfcode `\. \@m
575 }
576 {
577     \def \@noitemerr
578     { \@latex@warning { Empty~`thebibliography'-environment } }
579     \endlist
580 }
581 \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`

```

582 \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.

```
583 \NewDocumentCommand \appendix { D <> { all } }  
584 {  
585   \__talk_if_overlay:nT {#1}  
586   { \bool_set_true:N \l__talk_appendix_bool }  
587   \addtocontents { toc } { \appendixmarker }  
588   \cs_gset_eq:NN \contentsline \use_none:nnnn  
589 }
```

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

```
590 \endclass
```

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		block commands:	
\(	77	\g_block_nesting_depth_int	473
\)	77	block internal commands:	
\.	574	_block_debug_typeout:n	463
@ commands:		_block_if_list:TF	481, 492
_@_decode_overlay_+:nw	166	_block_inter_item:	442, 444
@starttoc@aux internal commands:		\l_block_level_incr_bool	471
_@starttoc@aux__talknn	253	_block_skip_remove_last:	450, 451, 485, 486
\\	350	_block_skip_set_to_last:N	496
A		\l_block_topsepadd_skip	67, 69, 442, 504
\abovecaptionskip	156, 158	\l_block_transparent_level_bool	464, 467
\action	144	\BlockEnvEnd	442, 520
actionenv (env.)	144	bool commands:	
\addcontentsline	214	\bool_do_until:nn	125
\addcontentslinebookmark	195, 207	\bool_do_while:Nn	30
\addcontentslinebookmarkOff	195, 212	\bool_gset_eq:NN	105
\addcontentslinebookmarkOn	195	\bool_gset_false:N	37, 104
\addcontentslinebookmarkReset	197	\bool_gset_true:N	103, 252, 262, 268
\addpenalty	457, 503	\bool_if:NTF	6, 44, 46, 46, 50, 50, 58, 66, 68, 71, 72, 78, 80, 83, 87, 102, 141, 145, 147, 153, 174, 187, 200, 204, 229, 245, 257, 355, 466, 470, 526, 532, 537, 561, 569, 641
\addtocontents	587	\bool_lazy_and:nnTF	21, 56, 56, 169, 265, 338
\AddToHook	54, 263, 367, 369, 405, 434, 450, 451, 511, 514, 517, 523, 637	\bool_lazy_and_p:nn	323, 328
\addtolength	48	\bool_lazy_or:nnTF	5, 18, 23, 321, 425
\addvspace	458, 504	\bool_lazy_or_p:nn	24
\advance	564	\bool_new:N	3, 3, 4, 4, 8, 9, 10, 15, 140, 142, 143, 232, 495, 497, 498, 582
\againframe	612	\bool_set_eq:NN	100, 179, 183
\alert	45, 122	\bool_set_false:N	32, 33, 34, 35, 36, 44, 172, 179, 576, 598
alertenv (env.)	129	\bool_set_true:N	27, 28, 60, 61, 78, 82, 83, 119, 152, 170, 172, 188, 230, 249, 253, 259, 264, 534, 585, 586, 607
\alt	217	\c_false_bool	15
\and	131	\c_true_bool	14, 196, 213
\appendix	583, 642	box commands:	
\appendixmarker	227, 587	\box_dp:N	63, 63, 362
\arabic	502, 506	\box_gc clear:N	98
\arraycolsep	25	\box_gset_eq:NN	122
\arrayrulewidth	27	\box_ht:N	58, 59, 62, 68, 111, 296, 321, 362
\AssignSocketPlug	194	\box_if_empty:N	141
\AssignStructureRole	125, 127		
\AssignTaggingSocketPlug	252		
\author	15		
B			
\begin	122, 126, 130, 144, 375, 435, 549, 550		
\begingroup	163, 255, 412		
\belowcaptionskip	157, 159		
\bfseries	21, 39, 285, 392, 528		
\bigskipamount	18		
block (env.)	384		

`\box_if_empty_p:N` 125
`\box_new:N` 5, 6, 7, 66, 171
`\box_set_dp:Nn` 60, 66
`\box_set_eq:NN` 119, 124
`\box_set_ht:Nn` 55, 60
`\box_use:N` 401
`\box_use_drop:N` 26, 66, 70,
71, 76, 130, 274, 301, 305, 307, 365, 430
`\box_wd:N` 40, 45
box internal commands:
`\__box_dim_eval:n` 37, 40, 45, 48
`\__box_set_to_wd:` 44, 49
`\__box_set_to_wd:NNnn` 30, 32, 33

C

`\clearpage` 155
clist commands:
`\clist_clear:N` 37, 38
`\clist_const:Nn` 68
`\clist_if_empty:NTF` 151, 223
`\clist_if_in:NnTF` 76, 229
`\clist_map_break:` 269
`\clist_map_inline:Nn` ... 132, 232, 417
`\clist_map_inline:nn`
..... 3, 97, 108, 139, 369, 516
`\clist_new:N` 12, 16, 67, 516
`\clist_pop:NNTF` 413
`\clist_put_left:Nn` 150
`\clist_remove_duplicates:N` 151
`\clist_set:Nn` 228
`\clist_set_eq:NN` 149
`\clubpenalty` 571, 572
`\color` 4, 11, 56, 285, 393, 526, 528
color commands:
`\color_ensure_current:` .. 63, 292, 302
`\color_group_begin:` 38, 50
`\color_group_end:` 38
`\color_math:nn` 9, 26
`\color_math:nnn` 10, 27
`\color_select:n` 7,
16, 37, 45, 46, 47, 69, 303, 353, 411, 452
`\color_select:nn` 8, 17, 38
color internal commands:
`\__color_backend_reset:` 64
`\colorlet` 68
`column (env.)` 78
`columns (env.)` 11
`\columnwidth` 20, 87, 122
`\contentsline` 227, 588
cs commands:
`\cs_generate_variant:Nn` 7,
8, 9, 10, 60, 62, 63, 64, 217, 273, 568
`\cs_gset:Npn` 136, 137, 138
`\cs_gset_eq:NN` 56, 88, 230, 231, 234, 588

`\cs_gset_nopar:Npn` 49, 57, 64
`\cs_gset_protected:Npe` 460
`\cs_gset_protected:Npn`
. 29, 31, 33, 42, 52, 225, 253, 345, 444
`\cs_if_exist:NTF` ... 143, 291, 442, 464
`\cs_if_exist_p:N` 339
`\cs_if_exist_use:NTF` ... 179, 189, 206
`\cs_new:Npn` 7, 10, 34, 35,
36, 37, 38, 39, 40, 41, 42, 185, 280,
440, 501, 502, 505, 506, 509, 512, 515
`\cs_new_eq:NN` 6,
9, 92, 195, 224, 233, 427, 500, 504
`\cs_new_nopar:Npn` 3, 483
`\cs_new_protected:Npe` 74, 96, 122
`\cs_new_protected:Npn` 11,
12, 18, 18, 20, 34, 38, 40, 41, 42,
42, 48, 52, 52, 54, 54, 56, 56, 62,
64, 65, 68, 69, 73, 76, 78, 82, 92,
100, 106, 108, 109, 117, 129, 134,
134, 155, 157, 159, 163, 165, 166,
167, 169, 175, 176, 176, 181, 186,
186, 190, 191, 193, 195, 198, 199,
204, 214, 215, 218, 219, 221, 228,
236, 241, 246, 261, 266, 272, 274,
284, 286, 296, 307, 310, 314, 319,
336, 339, 378, 410, 421, 423, 434,
436, 446, 523, 529, 553, 612, 621, 623
`\cs_set:Npn` 14, 15, 85
`\cs_set_eq:NN` ... 61, 62, 63, 64, 70,
179, 465, 466, 480, 481, 491, 492, 581
`\cs_set_nopar:Npn`
..... 130, 181, 182, 183, 184,
458, 460, 464, 468, 470, 475, 485, 490
`\cs_set_protected:Npn`
..... 44, 125, 131, 160, 256
`\cs_to_str:N` 368, 373
`\csname` 167, 265, 266

D

`\date` 15
`\day` 20
`\DeclareColor` 65, 71, 72, 73
`\DeclareInstance` 68,
79, 81, 83, 85, 87, 99, 133, 310,
311, 312, 313, 314, 315, 316, 368, 433
`\DeclareInstanceCopy` 371, 436
`\DeclareTemplateCode`
... 23, 54, 96, 109, 110, 289, 328, 386
`\DeclareTemplateInterface`
.... 16, 45, 94, 95, 105, 282, 318, 376
`\def` 577
`\definecolor` 69
`description (env.)` 363, 514

<code>\c_group_begin_token</code>	47	<code>\int_to_Roman:n</code>	282
<code>\group_end:</code> .. 32, 41, 51, 75, 88, 88,		<code>\int_to_roman:n</code>	283
127, 150, 215, 282, 358, 397, 455, 634		<code>\int_use:N</code>	11, 17, 81,
<code>\group_insert_after:N</code>	49	136, 152, 442, 446, 511, 514, 545, 648	
H		<code>\c_max_int</code>	242, 267
hbox commands:		<code>\invisible</code>	108
<code>\hbox:n</code>	130, 300	<code>invisibleenv (env.)</code>	117
<code>\hbox_set:Nw</code>	45, 72	iow commands:	
<code>\hbox_set_end:</code>	25, 75, 86	<code>\iow_now:Nn</code>	325
<code>\hbox_set_to_wd:Nnw</code>	18	<code>\item</code>	69, 405
<code>\headsep</code>	323, 342, 343	<code>itemize (env.)</code>	514
<code>\hfil</code>	83, 384, 428, 462, 473, 478, 488	<code>\itemsep</code>	54, 62, 69, 458
hook commands:		J	
<code>\hook_gput_code:nnn</code>	60, 119, 333	<code>\jobname</code>	260, 266
<code>\hook_new:n</code>	192	K	
<code>\hook_use:n</code>	190	kernel internal commands:	
<code>\hspace</code>	299, 307	<code>__kernel_displayblock_end:</code>	491
<code>\hypersetup</code>	187	<code>__kernel_list_item_begin:</code>	456
I		<code>__kernel_list_item_end:</code>	455
<code>\IfBooleanT</code>	396, 538	keys commands:	
<code>\ifcase</code>	5	<code>\l_keys_choice_str</code>	82, 85
<code>\IfFormatAtLeastF</code>	7	<code>\keys_define:nn</code>	
<code>\IfNoValueF</code>	397, 398, 540, 542	.. 3, 5, 6, 31, 69, 89, 97, 99, 233, 530	
<code>\IfNoValueTF</code>	15,	<code>\keys_set:nn</code>	
25, 36, 47, 67, 177, 254, 410, 413, 548		.. 7, 17, 17, 24, 29, 36, 41, 50, 84,	
<code>\ignorespaces</code>		106, 173, 256, 575, 584, 597, 606, 631	
.. 19, 21, 92, 167, 226, 266, 376, 551		<code>\keys_set_known:nn</code>	20, 32
<code>\immediate</code>	266	<code>\l_keys_value_tl</code>	46, 253
<code>\includegraphics</code>	387	L	
<code>\indent</code>	447	<code>\label</code>	102, 403
<code>\insertsection</code>	139	<code>\labelenumi</code>	34
<code>\insertsubsection</code>	139	<code>\labelenumii</code>	35
<code>\insertsubsubsection</code>	139	<code>\labelenumiii</code>	36
<code>\institute</code>	34	<code>\labelenumiv</code>	37
int commands:		<code>\labelitemfont</code>	38, 39, 40, 41, 42
<code>\int_compare:nNnTF</code>	82, 222,	<code>\labelitemi</code>	38
248, 251, 258, 261, 309, 341, 431, 639		<code>\labelitemii</code>	39
<code>\int_compare_p:nNn</code>		<code>\labelitemiii</code>	40
.. 266, 267, 324, 325, 329, 330		<code>\labelitemiv</code>	41
<code>\int_eval:n</code>	258, 432, 508	<code>\labelsep</code>	29, 33, 46, 48, 564
<code>\int_gdecr:N</code>	473	<code>\labelwidth</code>	47, 48, 562, 563
<code>\int_gincr:N</code>		<code>\Large</code>	21, 90
.. 36, 49, 81, 255, 440, 531, 533, 543		<code>\large</code>	392
<code>\int_gset:Nn</code>	184, 256	<code>\leavevmode</code>	89, 391
<code>\int_gset_eq:NN</code>	27, 212, 217, 222	<code>\leftmargin</code>	51, 59, 66, 563, 564
<code>\int_gzero:N</code>	16, 28, 95	<code>\leftmargini</code>	43, 47, 51
<code>\int_incr:N</code>	289	<code>\leftmarginii</code>	44, 59
<code>\int_max:nn</code>	218	<code>\leftmarginiii</code>	45, 66
<code>\int_new:N</code>	5, 8, 9, 10,	<code>\leftskip</code>	291, 301
200, 211, 228, 279, 433, 496, 499, 503		legacy commands:	
<code>\int_set_eq:NN</code>	15	<code>\legacy_if:nTF</code>	
<code>\int_step_inline:nn</code>	643	.. 48, 252, 261, 323, 446, 476, 482, 494	

<code>\legacy_if_gset_false:n</code>	479, 492, 493	<code>\numberline</code>	223
<code>\let</code>	567	O	
<code>\list</code>	560	<code>\obeyedline</code>	19, 85
M		<code>\only</code>	50, 132
<code>\makeatletter</code>	256	<code>onlyenv (env.)</code>	137
<code>\maketitle</code>	139	<code>\onslide</code>	46, 223
<code>\mathcolor</code>	5, 11	opacity commands:	
<code>\mbox</code>	381	<code>\opacity_begin:n</code>	39, 334
<code>\medskip</code>	390, 396	<code>\opacity_end:</code>	41, 336
<code>\mode</code>	41, 11	<code>\openout</code>	266
mode commands:		<code>\or</code>	5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
<code>\mode_if_horizontal:TF</code>	31, 448, 483	<code>overlayarea (env.)</code>	268
<code>\mode_if_horizontal_p:</code>	25	<code>overprint (env.)</code>	285
<code>\mode_if_math:TF</code>	380	P	
<code>\mode_if_vertical_p:</code>	26	<code>\pagecolor</code>	43
<code>\mode_leave_vertical:</code>	32, 383, 439, 478	<code>\pagestyle</code>	494
<code>\month</code>	5	<code>\paperheight</code>	57, 58
msg commands:		<code>\paperwidth</code>	58, 347, 405, 453, 454
<code>\msg_error:nnn</code>	138, 156, 161, 208, 225, 556, 619	<code>\par</code>	35, 37, 14, 28, 38, 71, 73, 80, 95, 100, 162, 168, 263, 277, 308, 357, 387, 395, 402, 447, 452, 487
<code>\msg_fatal:nn</code>	15	<code>\parsep</code>	53, 61, 62, 68, 69
<code>\g_msg_module_name_prop</code>	5	<code>\parskip</code>	501
<code>\g_msg_module_type_prop</code>	6	<code>\partopsep</code>	71
<code>\msg_new:nnn</code>	22, 348	<code>\pause</code>	46, 250
<code>\msg_new:nnnn</code>	9, 273, 279, 285, 647, 653, 659	pdfxform commands:	
<code>\msg_warning:nn</code>	27, 344, 364	<code>\pdfxform_new:nnn</code>	441
N		<code>\pdfxform_use:n</code>	445
<code>\NeedsDocumentMetadata</code>	17	prg commands:	
<code>\NewCommandCopy</code>	4, 5, 6, 173, 368, 388, 407, 522, 532, 636	<code>\prg_generate_conditional_variant:Nnn</code>	10
<code>\newcounter</code>	23, 130, 131, 132, 150	<code>\prg_new_protected_conditional:Nppn</code>	3, 3
<code>\NewDocumentCommand</code>	5, 10, 11, 34, 39, 65, 110, 122, 132, 139, 144, 145, 150, 217, 223, 227, 240, 250, 278, 583, 628	<code>\prg_return_false:</code>	8, 9
<code>\NewDocumentEnvironment</code>	11, 78, 117, 129, 137, 141, 152, 268, 285, 363, 364, 365, 366, 384, 558, 580, 602	<code>\prg_return_true:</code>	7, 8
<code>\NewEnvironmentCopy</code>	371, 544	<code>\ProcessedArgument</code>	14, 15
<code>\newlabel</code>	416	<code>\ProcessKeyOptions</code>	114
<code>\newlength</code>	156, 157	prop commands:	
<code>\NewSocketPlug</code>	181	<code>\prop_gput:Nnn</code>	5, 6
<code>\NewTaggingSocket</code>	235	property commands:	
<code>\NewTaggingSocketPlug</code>	236	<code>\property_new:nnnn</code>	11, 507, 510, 513
<code>\NewTemplateType</code>	15, 44, 93, 94, 104, 281, 317, 375	<code>\property_record:nn</code>	152, 519
<code>\newtheorem</code>	69, 532	<code>\property_ref:nn</code>	17, 509, 512, 515
<code>\newwrite</code>	265	<code>\protect</code>	223
<code>\noindent</code>	31, 339, 344, 397	<code>\providecommand</code>	195, 196, 197, 198
<code>\normalfont</code>	42, 50, 322, 392, 528	<code>\ProvidesExplClass</code>	3
<code>\normalsize</code>	166	<code>\put</code>	57
<code>\number</code>	20, 22	Q	
		<code>\quad</code>	131
		quark commands:	
		<code>\quark_if_recursion_tail_stop:N</code>	178

<code>\quark_if_recursion_tail_stop_-do:Nn</code>	195, 206	<code>\sfcode</code>	574
<code>\quark_if_recursion_tail_stop_-do:nn</code>	54	<code>\sfdefault</code>	179
<code>\q_recursion_stop</code>	45, 173, 198	<code>\SimpleBlockEnv</code>	519
<code>\q_recursion_tail</code>	45, 173, 198	<code>\skip</code>	30
<code>\q_stop</code> .	90, 99, 125, 126, 130, 135, 161, 165, 233, 236, 317, 319, 416, 421	skip commands:	
quotation (env.)	363	<code>\skip_horizontal:n</code>	32, 344, 366, 402, 431, 448, 454
quote (env.)	363	<code>\skip_if_eq_p:nn</code>	22
		<code>\skip_new:N</code>	17
		<code>\skip_set:Nn</code>	291, 301
		<code>\skip_set_eq:NN</code>	20
		<code>\skip_vertical:n</code> ...	33, 159, 161, 165, 167, 171, 173, 177, 179, 499, 500
		<code>\l_tmpa_skip</code>	496, 497, 499, 501
		sloppy	570
		socket commands:	
		<code>\socket_use:n</code>	506
		<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>	407, 411, 412
		<code>\stdmathcolor</code>	4
		<code>\stdnewtheorem</code>	532
		<code>stdquotation (env.)</code>	363
		<code>stdquote (env.)</code>	363
		<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 (env.)</code>	363
		<code>\stepcounter</code>	23
		str commands:	
		<code>\str_case:nn</code>	539
		<code>\str_clear:N</code>	22, 168, 574, 583, 596, 605, 632
		<code>\str_if_empty:N</code> .	114, 135, 145, 535
		<code>\str_if_empty_p:N</code>	57
		<code>\str_if_eq:nnTF</code>	20, 79, 111, 169, 182, 184
		<code>\str_if_eq_p:nn</code>	6, 7, 25, 170, 171, 427
		<code>\str_new:N</code> ..	11, 13, 14, 17, 87, 88, 141
		<code>\str_put_right:Nn</code>	181, 217
		<code>\str_replace_all:Nnn</code>	23, 25, 63
		<code>\str_set:Nn</code>	21, 31, 78, 147
		<code>\str_set_eq:NN</code>	82, 85, 181
R			
<code>\raggedright</code>	90, 156, 265		
<code>\refstepcounter</code>	161		
<code>\relax</code>	266		
<code>\relsize</code>	120		
<code>\renewcommand</code>	568		
<code>\RenewCommandCopy</code>	29		
<code>\RenewDocumentCommand</code>	11, 15, 21, 22, 27, 30, 43, 174, 369, 389, 403, 408, 428, 533		
<code>\RenewDocumentEnvironment</code>	372, 518, 545, 571, 593		
<code>\RequirePackage</code> .	3, 118, 139, 157, 161, 163, 164, 167, 170, 177, 178, 186, 387		
<code>\reuseframe</code>	612		
<code>\rmdefault</code>	179		
<code>\rule</code>	58		
rule commands:			
<code>\rule:nnn</code>	52, 453		
S			
scan commands:			
<code>\scan_stop:</code>	58		
<code>\scriptsize</code>	84		
<code>\section</code>	130, 139		
seq commands:			
<code>\seq_gpop_left:NN</code>	186		
<code>\seq_gpush:Nn</code>	176		
<code>\seq_gput_right:Nn</code>	229		
<code>\seq_gremove_all:Nn</code>	133, 134, 135		
<code>\seq_gset_from_clist:Nn</code>	202		
<code>\seq_map_indexed_inline:Nn</code>	141		
<code>\seq_map_inline:Nn</code>	209, 216, 221		
<code>\seq_new:N</code>	172, 201		
<code>\seq_set_from_clist:Nn</code>	139		
<code>\l_tmpa_seq</code>	139, 141		
<code>\setcounter</code>	24, 362		
<code>\setlength</code>	25, 26, 27, 28, 29, 31, 32, 33, 43, 44, 45, 46, 47, 71, 158, 159		
<code>\setmainfont</code>	171		
<code>\setmathfont</code>	173		
<code>\setsansfont</code>	172		
<code>\SetTemplateKeys</code>	121		
<code>\settowidth</code>	562		

<code>\string</code>	416	<code>__talk_action_visible:N</code>	48, 56
<code>\subsection</code>	130, 139	<code>__talk_action_visible_end:N</code> .	48, 62
<code>\subsubsection</code>	130, 139	<code>\l__talk_appendix_bool</code>	
<code>\subtitle</code>	34		229, 532, 582, 586, 641
sys commands:		<code>\g__talk_appendixframe_int</code>	
<code>\c_sys_engine_str</code>	24		503, 508, 514, 533
<code>\sys_if_engine luatex:TF</code> ..	165, 448	<code>\l__talk_aspect_ratio_str</code> ..	69, 134
<code>\sys_if_engine luatex_p:</code>	19	<code>\g__talk_break_box</code>	5, 122, 124
<code>\sys_if_engine opentype:TF</code>	158	<code>\l__talk_break_box</code>	
<code>\sys_if_engine pdftex_p:</code>	20		6, 119, 120, 122, 124, 125, 127
T			
<code>\tabbingsep</code>	29	<code>\g__talk_cnt_reset_seq</code>	
<code>\tabcolsep</code>	26		133, 134, 135, 201, 216, 221, 229
<code>table (env.)</code>	139	<code>__talk_cnt_restore:</code>	79, 214, 219
<code>\tablename</code>	150	<code>__talk_cnt_save:</code>	99, 214, 214
<code>\tableofcontents</code>	278	<code>__talk_column_align_bottom:n</code>	54, 54
tag commands:		<code>__talk_column_align_center:n</code>	54, 56
<code>\tag_get:n</code>	184	<code>__talk_column_align_top:n</code> ..	54, 73
<code>\tag_mc_begin:n</code>	192, 271, 278	<code>\l__talk_column_alignment_tl</code> .	31, 98
<code>\tag_mc_end:</code>	198, 269, 276	<code>\g__talk_column_int</code> 9, 15, 16, 27, 81, 82	
<code>\tag_resume:n</code>	85, 197, 268	<code>\l__talk_column_int</code>	9, 15, 27
<code>\tag_struct_begin:n</code> .	91, 183, 243, 270	<code>\l__talk_columns_wd_tl</code>	5, 18, 19
<code>\tag_struct_end:</code> ...	92, 188, 250, 277	<code>__talk_decode_action:n</code> 113, 122, 122	
<code>\tag_suspend:n</code>	73, 193, 279	<code>__talk_decode_action:w</code> 122, 124, 129	
tag internal commands:		<code>\l__talk_decode_action_str</code>	
<code>\g__tag_title_author_tl</code>	19		14, 22, 135, 145, 147, 182, 190
<code>\g__tag_title_title_tl</code>	31	<code>\l__talk_decode_actions_bool</code> ...	
<code>\tagpdfparaOff</code>	86		15, 32, 184, 192
<code>\tagpdfsetup</code>	168, 188	<code>\l__talk_decode_actions_clist</code> 15, 37	
talk commands:		<code>\l__talk_decode_actions_str</code>	15
<code>\l__talk_frame_properties_clist</code> .		<code>\l__talk_decode_arg_str</code>	
.....	149, 516	... 11, 31, 39, 139, 157, 162, 209, 226	
talk internal commands:		<code>__talk_decode_check:n</code> .	174, 221, 221
<code>__talk_action_alert:N</code> 42, 42, 125, 130		<code>__talk_decode_check:nw</code> 221, 233, 236	
<code>__talk_action_begin:n</code> 13, 91, 143,		<code>__talk_decode_check_range:nnn</code> .	
144, 147, 153, 155, 374, 386, 438, 547			221, 242, 243, 256
<code>__talk_action_begin:w</code> .	144, 160, 165	<code>__talk_decode_check_single:nn</code> .	
<code>__talk_action_begin_auxi:n</code>			221, 239, 246
.....	144, 163, 166, 167	<code>__talk_decode_mode:n</code>	64, 74, 74
<code>__talk_action_begin_auxii:n</code> ...		<code>__talk_decode_mode:nn</code> .	104, 107, 109
.....	144, 173, 176	<code>__talk_decode_mode:w</code>	74, 89, 96
<code>__talk_action_end:</code> 37, 29, 96, 144,		<code>__talk_decode_mode_aux:n</code>	74
148, 149, 154, 198, 380, 403, 434, 555		<code>\l__talk_decode_mode_bool</code>	
<code>__talk_action_invisible:N</code> 48, 48, 196			8, 27, 35, 60, 82, 172, 526
<code>__talk_action_invisible_end:N</code> .		<code>\l__talk_decode_modes_bool</code>	
.....	48, 54, 213		9, 36, 58, 78, 119
<code>__talk_action_only:N</code>	64, 64, 138	<code>__talk_decode_overlay_.:nw</code>	166
<code>__talk_action_only_end:N</code> 64, 78, 139		<code>__talk_decode_overlay_aux:nNn</code> .	
<code>\l__talk_action_spec_str</code>			166, 189, 192, 193
.....	161, 232, 416, 530	<code>__talk_decode_overlay_offset:nNn</code>	
<code>__talk_action_uncover:N</code> 100, 100, 189			166, 197, 202, 212, 215
<code>__talk_action_uncover_end:N</code> ...		<code>__talk_decode_overlay_offset:nNnN</code>	
.....	100, 106, 191		166, 201, 204, 213
		<code>__talk_decode_overlays:nN</code>	
			166, 173, 176, 182, 219

__talk_decode_overlays:nn		__talk_frame_reuse:nn	
.....	115, 136, 150, 166, 166		612, 612, 633, 644
\l__talk_decode_overlays_bool	. 3,	__talk_frame_reuse_aux:nn	
6, 28, 33, 61, 83, 152, 180, 187, 189, 191			612, 616, 621
\l__talk_decode_overlays_clist	.	__talk_frame_reuse_aux:nnn	
.....	12, 38, 151		612, 622, 623
\l__talk_decode_overlays_str	...	__talk_frame_save:nn	
.....	12, 57, 114		523, 536, 544, 553, 568
__talk_decode_parse:n		\g__talk_frame_struct_int	81, 184, 200
.....	5, 18, 18, 178, 188, 525	\g__talk_frame_subtitle_tl	3, 13, 97
__talk_decode_parse:w	. 18, 45, 52, 66	\l__talk_frame_suppl_bool	. 232, 537
__talk_decode_parse_auxi:n		\g__talk_frame_suppl_int	
.....	18, 19, 20		496, 543, 545, 639, 643
__talk_decode_parse_auxii:n	...	\g__talk_frame_tag_bool	
.....	18, 39, 42	. 46, 68, 71, 72, 83, 103, 104, 257, 497	
\l__talk_decode_step_bool		\l__talk_frame_tagging_str	
.....	10, 44, 46, 188		20, 21, 23, 25, 102, 232
\l__talk_decode_trailing_bool	..	__talk_frame_title:n	 15, 36, 69
.....	4, 34, 245, 253, 259	\l__talk_frame_title_bool	69, 145, 569
\l__talk_float_alignment_tl		__talk_frame_title_tagged:n	...
.....	103, 115, 116, 117, 126, 136		15, 72, 76
\l__talk_fontsize_dim	... 69, 115, 120	\g__talk_frame_title_tl	
\l__talk_footelem_color_tl	 281		3, 8, 83, 96, 359, 586
\l__talk_footelem_font_tl	 281	\l__talk_frame_verb_bool	
\l__talk_footelem_left_skip	 281	. 50, 498, 561, 576, 585, 598, 607, 625	
\l__talk_footelem_right_skip	... 281	\l__talk_frametitle_after_skip	.
\l__talk_footer_bg_tl	 375		25, 53, 63
\l__talk_footer_fg_tl	 375	\l__talk_frametitle_before_skip	
\l__talk_footer_font_tl	 375		26, 43, 58
\l__talk_footer_left_skip	 375	\l__talk_frametitle_color_tl	...
\l__talk_footer_order_clist	 375		27, 45, 48
\l__talk_footer_right_skip	 375	\l__talk_frametitle_font_tl	.. 28, 34
\l__talk_footer_sep_tl	 375	__talk_head_marks:nnnnn	
\g__talk_footnote_box			186, 199, 199, 217
.....	98, 111, 141, 144, 171, 183, 185	__talk_head_marks_aux:Nnn	 199
\g__talk_footnote_overlay_seq	..	__talk_head_marks_aux:Nnnn	
.....	172, 176, 186		206, 213, 218
\l__talk_frame_alignment_tl		__talk_head_section:Nnn	 139
.....	94, 113, 139, 231, 252	__talk_head_subsection:Nnn	 139
\l__talk_frame_bool	141, 153, 495, 534	__talk_head_subsubsection:Nnn	. 139
\l__talk_frame_break_bool	57, 87, 232	__talk_head_tag:nn	 235, 238, 241
\l__talk_frame_break_fp	121, 128, 232	\l__talk_header_bg_tl	 317
\g__talk_frame_int	 17,	\l__talk_header_fg_tl	 317
136, 152, 282, 499, 508, 511, 531, 648		\l__talk_header_font_tl	 317
\l__talk_frame_name_str		\l__talk_header_frametitle_bool	317
232, 535, 536, 574, 583, 596, 605, 632		\l__talk_header_ht_dim	 317
__talk_frame_overprint:		\l__talk_header_left_skip	 317
.....	280, 280, 292, 295, 299,	\l__talk_header_right_skip	 317
312, 314, 315, 318, 320, 327, 329, 334		__talk_header_tag_begin:n	
__talk_frame_process:nn			78, 266, 266, 273
.... 523, 523, 577, 587, 599, 608, 626		__talk_header_tag_end:	. 89, 266, 274
__talk_frame_process_aux:nn	...	__talk_if_overlay:n	 3, 10
.....	523, 527, 529		

__talk_if_overlay:nTF ..	3 , 7 , 12 , 13 , 13 , 23 , 34 , 38 , 45 , 102 , 134 , 155 , 219 , 232 , 242 , 372 , 391 , 406 , 430 , 585	\l__talk_saved_overlays_bool ...	140 , 179 , 204
__talk_item_parse_spec:n	405 , 418 , 422 , 423	\l__talk_sec_bookmark_tl	97 , 176 , 188
__talk_item_parse_spec:w	405 , 415 , 421	\l__talk_sec_label_tl	97
__talk_label:n	403 , 407 , 410	\l__talk_sec_nameref_tl	97 , 189
__talk_latex_frame:n ..	29 , 522 , 522	\l__talk_sec_numbered_bool .	97 , 179
\l__talk_list_end_tl	429 , 435 , 441 , 454 , 490 , 512	\l__talk_sec_quote_tl	97
__talk_metadata_name:n	416 , 419 , 424 , 440 , 440	\l__talk_sec_running_tl	97 , 177
__talk_mode:n	3	\l__talk_sec_subtitle_tl	97
__talk_mode:nTF	3 , 12	\l__talk_sec_toc_tl	97 , 178 , 187
\l__talk_mode_str	7 , 69 , 80 , 111	\g__talk_section_tl	91
\c__talk_modes_clist	68 , 76	\l__talk_section_tl	91
__talk_onslide:n .	223 , 225 , 228 , 257	\l__talk_shuffle_skip ..	17 , 20 , 22 , 34
\g__talk_onslide_tl	54 , 94 , 185 , 202 , 230 , 231 , 235 , 239	__talk_shuffle_skip:n .	18 , 18 , 39 , 41
__talk_opacity_begin:n	38 , 38 , 51 , 52 , 59 , 60 , 98 , 103 , 234	__talk_slide:nn	12 , 12 , 548 , 551
__talk_opacity_end:	38 , 40 , 55 , 63 , 107 , 211 , 236	__talk_slide_align_bottom:n	157 , 157
__talk_opacity_group_begin: ...	434 , 434 , 450	__talk_slide_align_center:n	157 , 163
__talk_opacity_group_end: ...	434 , 436 , 451	__talk_slide_align_stretch:n ..	157 , 169
\g__talk_opacity_group_int	433 , 440 , 442 , 446	__talk_slide_align_top:n .	157 , 175
\l__talk_overlay_all_bool	143 , 170 , 172 , 200	__talk_slide_auxi:nn	12 , 31 , 34
__talk_overlay_arg:n	3 , 11 , 111 , 118 , 122 , 129 , 137	__talk_slide_auxii:n	12 , 39 , 41
__talk_overprint_begin:n	261 , 261 , 269 , 286	__talk_slide_auxiii: .	12 , 65 , 68 , 123
__talk_overprint_check_ht:n ...	285 , 334 , 336 , 345	__talk_slide_auxiv:n	12 , 51 , 82
\l__talk_overprint_int .	279 , 283 , 289	\l__talk_slide_box	5 , 7 , 44 , 45 , 49 , 59 , 70 , 76 , 110 , 114 , 119 , 120 , 127 , 129 , 130 , 140
__talk_overprint_save_ht: ...	285 , 290 , 310	__talk_slide_break:	62 , 117 , 117
__talk_pagecolor:n	43 , 48 , 49 , 52	\g__talk_slide_continue_bool ..	3 , 30 , 37 , 78 , 101 , 105 , 147 , 252 , 262 , 268
\c__talk_paper_height_dim	122	\l__talk_slide_continue_bool ...	3 , 100 , 106
\c__talk_paper_width_dim	122	__talk_slide_fill:	64 , 108 , 108
\g__talk_pauses_int	10 , 5 , 49 , 95 , 218 , 255 , 256 , 258	__talk_slide_init:	43 , 92 , 92
\l__talk_saved_action_str	140 , 181 , 207	\g__talk_slide_int	8 , 11 , 28 , 36 , 248 , 251 , 258 , 261 , 266
\l__talk_saved_actions_bool	140 , 183 , 209	__talk_slide_output:	21 , 80 , 131 , 134 , 134
__talk_saved_contentline:nnnn .	227 , 230 , 233 , 234	__talk_slide_tag_off_begin: ...	48 , 190 , 190
__talk_saved_ifendpe: .	64 , 70 , 88 , 92	__talk_slide_tag_off_end:	74 , 190 , 195
		__talk_slide_tag_on_begin:	47 , 181 , 181
		__talk_slide_tag_on_end: 73 , 181 , 186	
		\g__talk_subsection_tl	91
		\l__talk_subsection_tl	91 , 183
		\g__talk_subsubsection_tl	91
		\l__talk_subsubsection_tl ..	91 , 185
		\l__talk_suppl_mode_str	69 , 539
		__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

<code>\l__talk_titlelem_font_tl</code>	44	<code>\ifpackageloaded</code>	160
<code>\l__talk_titlelem_tag_begin_tl</code> . .	44	<code>\ignore</code>	37
<code>\l__talk_titlelem_tag_end_tl</code> . . .	44	<code>\ignoretrue</code>	101
<code>\l__talk_titlepage_alignment_tl</code> .	94	<code>\inmatherr</code>	489
<code>\l__talk_titlepage_framestyle_tl</code>	94	<code>\input</code>	260
<code>\l__talk_titlepage_order_clist</code> . .	94	<code>\institute</code>	3 , 37
<code>__talk_tmp:w</code>	65 , 65 , 125 , 134	<code>\itempenalty</code>	457
<code>\l__talk_tmp_box</code>	18 , 26 , 32 , 40 , 50 , 55 , 58 , 59 , 60 , 60 , 62 , 63 , 63 , 66 , 66 , 66 , 68 , 71 , 71 , 72 , 75 , 85 , 99 , 264 , 274 , 296 , 301 , 305 , 307 , 321 , 345 , 362 , 365 , 388 , 401 , 403 , 430 , 435 , 444	<code>\kernel@reserved@label@data</code> . . .	422
<code>\l__talk_tmp_clist</code>	67 , 149 , 150 , 151 , 153	<code>\latex@warning</code>	578
<code>\l__talk_tmp_tl</code>	15 , 21 , 24 , 26 , 68 , 112 , 187 , 188 , 413 , 415 , 416	<code>\listI</code>	56
<code>__talk_toc_aux:nnnn</code>	284 , 285 , 288 , 298 , 307	<code>\listi</code>	49 , 56
<code>__talk_toc_dest:n</code>	284 , 311 , 314	<code>\listii</code>	57
<code>__talk_toc_dest:w</code>	284 , 316 , 319	<code>\listiii</code>	64
<code>__talk_toc_level:nnnn</code>	284 , 312 , 339	<code>\m</code>	574
<code>\l__talk_uncover_hidden_fp</code>	93	<code>\makecaption</code>	167
<code>\l__talk_vcenter_offset_tl</code>	64 , 69 , 75	<code>\makefntext</code>	195
<code>__talk_wallpaper_hruler:Nnn</code> . . .	340 , 398 , 446 , 446	<code>\mpfootins</code>	30
<code>talk/sec/title</code>	235	<code>\nobreakfalse</code>	269
<code>\temporal</code>	10 , 240	<code>\noitemerr</code>	482 , 577
T _E X and L ^A T _E X 2 _ε commands:		<code>\oddfoot</code>	464 , 466 , 475 , 481 , 490 , 492
<code>\appendixframenumber</code>	503	<code>\oddhead</code>	460 , 465 , 470 , 480 , 485 , 491
<code>\appendixframes</code>	513	<code>\openbib@code</code>	565 , 581
<code>\arabic</code>	7 , 10 , 136 , 137 , 138 , 185 , 501 , 505 , 560 , 568	<code>\outerparskip</code>	501
<code>\author</code>	3 , 18 , 19	<code>\parboxrestore</code>	88 , 164
<code>\auxout</code>	325 , 414	<code>\setminipage</code>	165
<code>\biblabel</code>	560 , 562	<code>\shortauthor</code>	3
<code>\bsphack</code>	405	<code>\shortdate</code>	3
<code>\caption</code>	160	<code>\shortinstitute</code>	3
<code>\capytype</code>	130	<code>\shortsubstitute</code>	3
<code>\clubpenalty</code>	572	<code>\shorttitle</code>	3
<code>\contentsline@destination</code>	64 , 316 , 347 , 350 , 353 , 356	<code>\skiphyperreftrue</code>	38 , 131
<code>\currentHref</code>	421	<code>\starttoc</code>	253 , 281
<code>\currentlabel</code>	418	<code>\starttoc@aux@nn</code>	258 , 272
<code>\currentlabelname</code>	201 , 420	<code>\subtitle</code>	3 , 42
<code>\currenvir</code>	489	<code>\title</code>	3 , 30 , 31
<code>\date</code>	3 , 25	<code>\totalframes</code>	510
<code>\definecounter</code>	224	<code>\c@appendixframe</code>	503
<code>\empty</code>	567 , 581	<code>\c@enumiv</code>	560 , 568
<code>\endparpenalty</code>	503	<code>\c@figure</code>	150
<code>\esphack</code>	408	<code>\c@frame</code>	499
<code>\evenfoot</code>	466 , 481 , 492	<code>\c@page</code>	185
<code>\evenhead</code>	465 , 480 , 491	<code>\c@pauses</code>	5
<code>\framenumber</code>	499	<code>\c@section</code>	136
<code>\frames</code>	507	<code>\c@slide</code>	8
		<code>\c@subsection</code>	137
		<code>\c@subsubsection</code>	138
		<code>\c@table</code>	150
		<code>\check@mathfonts</code>	58
		<code>\currentgrouplevel</code>	67
		<code>\fnum@figure</code>	150
		<code>\fnum@table</code>	150
		<code>\Gm@bmargin</code>	401
		<code>\Gm@lmargin</code>	324 , 382 , 448
		<code>\Gm@rmargin</code>	326 , 366 , 383 , 431

<code>\Gm@tmargin</code>	323	<code>\textwidth</code>	35, 8, 19, 20, 86, 87, 285
<code>\if@endpe</code>	70, 88, 92	<code>\theappendixframe</code>	503
<code>\if@minipage</code>	165	<code>thebibliography</code>	558
<code>\ifmeasuring@</code>	11	<code>\theenumi</code>	34
<code>\ignorespaces</code>	37	<code>\theenumii</code>	35
<code>\l@section</code>	284	<code>\theenumiii</code>	36
<code>\l@subsection</code>	284	<code>\theenumiv</code>	37, 568
<code>\l@subsubsection</code>	284	<code>\thefigure</code>	150
<code>\label@in@display</code>	53, 427, 428	<code>\theframe</code>	499
<code>\on@line</code>	463	<code>\thepage</code>	6, 185, 419
<code>\p@enumiv</code>	567	<code>\the pauses</code>	5
<code>\protected@write</code>	414	<code>\thesection</code>	130
<code>\ps@plain</code>	458	<code>\theslide</code>	8
<code>\ps@talk</code>	458	<code>\thesubsection</code>	130
<code>\ps@wallpaper</code>	458	<code>\thesubsubsection</code>	130
<code>\reset@color</code>	62, 63	<code>\thetable</code>	150
<code>\set@color</code>	61, 63	<code>\thispagestyle</code>	129
<code>\std@definecounter</code>	224	<code>\thispdfpagelabel</code>	136
<code>\std@label@in@display</code>	427	<code>\tiny</code>	381
<code>\stdreset@color</code>	61	<code>\title</code>	15
<code>\stdset@color</code>	61	tl commands:	
<code>\textsubscript@offset</code>	181	<code>\tl_clear:N</code>	176, 177, 178, 183, 185, 435
<code>\textsubscript@space</code>	181	<code>\tl_clear_new:N</code>	557
<code>\textsuperscript@offset</code>	181	<code>\tl_gclear:N</code> ...	12, 13, 94, 96, 97, 231
<code>\textsuperscript@space</code>	181	<code>\tl_gput_right:Nn</code>	235
tex commands:		<code>\tl_gset:Nn</code>	8,
<code>\tex_currentgrouplevel:D</code> ..	431, 432		13, 18, 25, 30, 37, 42, 315, 318, 558, 586
<code>\tex_hsize:D</code>	37, 48	<code>\tl_gset_eq:NN</code>	14, 19, 31
<code>\tex_lastskip:D</code>	20	<code>\tl_if_blank:nTF</code>	35,
<code>\tex_setbox:D</code>	35, 46		102, 118, 133, 148, 202, 204, 211, 241
<code>\tex_unskip:D</code>	29	<code>\tl_if_blank_p:n</code>	24, 426
<code>\tex_vbox:D</code>	30, 46	<code>\tl_if_empty:NTF</code>	45, 68,
<code>\tex_vrule:D</code>	54		128, 185, 202, 302, 352, 410, 419, 449
<code>\tex_vtop:D</code>	32	<code>\tl_if_empty:nTF</code>	64, 238, 297
<code>\texorpdfstring</code>	198, 214	<code>\tl_if_exist:NTF</code> ..	312, 442, 555, 614
text commands:		<code>\tl_if_novalue:NTF</code>	158
<code>\text_purify:n</code>	83, 239	<code>\tl_map_inline:nn</code>	353
<code>\text_titlecase_first:n</code>	152	<code>\tl_new:N</code>	3,
<code>\textasteriskcentered</code>	40		4, 49, 68, 75, 91, 92, 93, 94, 95, 96,
<code>\textbf</code>	353		103, 129, 151, 153, 231, 239, 314, 441
<code>\textbullet</code>	38	<code>\tl_put_right:N</code>	102
<code>\textcolor</code>	6, 11	<code>\tl_retokenize:n</code>	89
<code>\textendash</code>	39	<code>\tl_set:Nn</code>	15,
<code>\textheight</code>	60, 111, 121, 128, 137		76, 115, 115, 116, 116, 117, 117,
<code>\textit</code>	353		121, 122, 123, 124, 152, 172, 201, 429
<code>\textmd</code>	353	<code>\tl_set_eq:NN</code>	45, 154, 252
<code>\textnormal</code>	353	<code>\tl_to_str:n</code>	70,
<code>\textperiodcentered</code>	41		71, 72, 89, 105, 125, 126, 130, 135, 588
<code>\textrm</code>	353	<code>\tl_trim_spaces:n</code>	65
<code>\textsc</code>	353	<code>\tl_use:N</code>	54, 147, 230
<code>\textsf</code>	353	<code>\today</code>	3
<code>\textsl</code>	353	token commands:	
<code>\texttt</code>	353	<code>\token_if_eq_meaning:NNTF</code> ..	200, 211
<code>\textup</code>	353	<code>\token_to_str:N</code>	97, 98

<code>\topsep</code>	52, 60, 67	<code>\vbox_set:Nn</code>	40, 59, 129
U		<code>\vbox_set:Nw</code>	44, 49, 71, 75, 388
<code>\uncover</code>	108	<code>\vbox_set_end:</code>	
<code>\uncoverenv (env.)</code>	117		55, 77, 82, 87, 97, 271, 288, 400
<code>\unskip</code>	24	<code>\vbox_set_split_to_ht:NNn</code> .	120, 127
use commands:		<code>\vbox_set_to_ht:Nnn</code>	110
<code>\use:N</code>	98,	<code>\vbox_set_to_wd:Nnn</code>	29, 345, 403
113, 139, 162, 165, 169, 223, 246, 625		<code>\vbox_set_to_wd:Nnw</code>	42, 85, 264
<code>\use:n</code>	38, 52, 73, 86, 94,	<code>\vbox_set_top:Nn</code>	32
123, 123, 127, 137, 143, 310, 393, 535		<code>\vbox_set_top_to_wd:Nnn</code>	31
<code>\use_ii:nn</code>	259	<code>\vbox_to_ht:nn</code>	64, 137, 272, 298
<code>\use_none:n</code>	191, 208	<code>\vbox_top:n</code>	74
<code>\use_none:nnnn</code>	231, 233, 588	<code>\vbox_unpack:N</code>	50, 185
<code>\usebox</code>	444	<code>\vbox_unpack_drop:N</code> .	99, 114, 140, 144
<code>\usecounter</code>	566	vcoffin commands:	
<code>\UseHookWithArguments</code>		<code>\vcoffin_set:Nnn</code>	2
.....	346, 349, 351, 354, 413	<code>verse (env.)</code>	363
<code>\UseInstance</code>	104, 134, 144,	<code>\vfil</code>	275, 302, 359
194, 358, 415, 423, 472, 477, 487, 490		<code>\vfill</code>	21
<code>\UseStructureName</code>	128, 168, 246	<code>\visible</code>	108
<code>\UseTaggingSocket</code> .	162, 163, 181, 274, 276	<code>visibleenv (env.)</code>	117
<code>\UseTemplate</code>	142, 146	<code>\vspace</code>	42, 52, 66, 76
V		W	
<code>\value</code> ...	222, 309, 324, 325, 329, 330, 341	<code>\widowpenalty</code>	573
vbox commands:		Y	
<code>\vbox:n</code>	55	<code>\year</code>	22
<code>\vbox_gset:Nn</code>	183		