

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

Joseph Wright[†]

Released 2026-10-13

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 v0.6.10, last revised 2026-10-13.

[†]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	36
1	ltx-talk-frame-structure implementation	36
1.1	Columns	36
1.2	Floats	38
1.3	Footnotes	40
VI	ltx-talk-mode – Modes	42
1	ltx-talk-mode implementation	42
VII	ltx-talk-overlay – Overlays	43
1	ltx-talk-overlay implementation	43
1.1	Utilities	43
1.2	Opacity utilities	44
1.3	Action commands and environments	44
1.4	Non-action commands and environments	49
1.5	Fixed-size areas	50
1.6	Adding overlays to existing commands	52
1.7	Overlay patching of third-party environments	54
VIII	ltx-talk-required – “Required” definitions	56
1	ltx-talk-required implementation	56
1.1	Standard design settings	56
1.2	List support	57
IX	ltx-talk-structure – Structural commands	58

1	ltx-talk-structure implementation	58
1.1	Frame title	58
1.2	Headings	60
1.3	Table of contents	64
1.4	Block environments	66
1.5	Lists	67
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} {2026-10-13} {0.6.10}
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 \</class>
```

Part III

ltx-talk-decode – Decoding overlay specs

1 ltx-talk-decode implementation

Start the DocStrip guards.

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

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

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

(End of definition for \l__talk_decode_overlays_bool.)

`\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
\l__talk_decode_actions_str

```
15 \bool_new:N \l__talk_decode_actions_bool
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_actions_str
39       \clist_clear:N \l__talk_decode_overlays_clist
40       \clist_clear:N \l__talk_decode_overlays_str
41       \exp_args:No __talk_decode_parse_auxii:n { \l__talk_decode_arg_str }
42     }
43 }
```

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.

```

44 \cs_new_protected:Npn \__talk_decode_parse_auxii:n #1
45 {
46   \bool_set_false:N \l__talk_decode_step_bool
47   \__talk_decode_parse:w #1 | \q_recursion_tail | \q_recursion_stop
48   \bool_if:NT \l__talk_decode_step_bool
49   {
50     \legacy_if:nF { measuring@ }
51     { \int_gincr:N \g__talk_pauses_int }
52   }
53 }

```

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

```

54 \cs_new_protected:Npn \__talk_decode_parse:w #1 |
55 {
56   \quark_if_recursion_tail_stop_do:nn {#1}
57   {
58     \bool_lazy_and:nnT
59     { \str_if_empty_p:N \l__talk_decode_overlays_str }
60     { ! \l__talk_decode_modes_bool }
61     {
62       \bool_set_true:N \l__talk_decode_mode_bool
63       \bool_set_true:N \l__talk_decode_overlays_bool
64     }
65   }
66   \exp_args:Ne \__talk_decode_mode:n
67   { \tl_trim_spaces:n {#1} }
68   \__talk_decode_parse:w
69 }

```

(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.

```

70 \clist_const:Ne \c__talk_modes_clist
71 {
72   \tl_to_str:n { article } ,
73   \tl_to_str:n { handout } ,
74   \tl_to_str:n { projector }
75 }

```

(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.

```

76 \cs_new_protected:Npe \__talk_decode_mode:n #1
77 {
78   \clist_if_in:NnTF \exp_not:N \c__talk_modes_clist {#1}
79   {

```

```

80     \bool_set_true:N \exp_not:N \l__talk_decode_modes_bool
81     \exp_not:N \str_if_eq:VnT
82     \exp_not:N \l__talk_mode_str {#1}
83     {
84         \bool_set_true:N \exp_not:N \l__talk_decode_mode_bool
85         \bool_set_true:N \exp_not:N \l__talk_decode_overlays_bool
86     }
87 }
88 {
89     \exp_not:N \bool_if:NF \exp_not:N \l__talk_frame_break_bool
90     {
91         \exp_not:N \__talk_decode_mode:w #1 \tl_to_str:n { : : }
92         \exp_not:N \q_stop
93     }
94 }
95 }
96 \use:e
97 {
98     \cs_new_protected:Npe \exp_not:N \__talk_decode_mode:w
99     #1 \token_to_str:N :
100     #2 \token_to_str:N :
101     #3 \exp_not:N \q_stop
102 }
103 {
104     \exp_not:N \tl_if_blank:nTF {#2}
105     {
106         \exp_not:N \__talk_decode_mode:nn
107         { \tl_to_str:n { projector } } {#1}
108     }
109     { \exp_not:N \__talk_decode_mode:nn {#1} {#2} }
110 }
111 \cs_new_protected:Npn \__talk_decode_mode:nn #1#2
112 {
113     \str_if_eq:VnTF \l__talk_mode_str {#1}
114     {
115         \__talk_decode_action:n {#2}
116         \str_if_empty:NT \l__talk_decode_overlays_str
117         { \__talk_decode_overlays:nn { overlays } { * } }
118     }
119     {
120         \tl_if_blank:nT {#2}
121         { \bool_set_true:N \l__talk_decode_modes_bool }
122     }
123 }

```

(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.

```

124 \cs_new_protected:Npe \__talk_decode_action:n #1
125 {
126     \exp_not:N \__talk_decode_action:w
127     #1 \tl_to_str:n { @ @ } \exp_not:N \q_stop

```

```

128 }
129 \use:e
130 {
131   \cs_new_protected:Npn \exp_not:N \__talk_decode_action:w
132     #1 \tl_to_str:n { @ } #2 \tl_to_str:n { @ } #3 \exp_not:N \q_stop
133 }
134 {
135   \tl_if_blank:nTF {#2}
136   {
137     \str_if_empty:NTF \l__talk_decode_action_str
138     { \__talk_decode_overlays:nn { overlays } {#1} }
139     {
140       \msg_error:nnV { talk } { misplaced-action-spec }
141       \l__talk_decode_arg_str
142     }
143   }
144   {
145     \cs_if_exist:cTF { __talk_action_ #1 :N }
146     {
147       \str_if_empty:NTF \l__talk_decode_action_str
148       {
149         \str_set:Nn \l__talk_decode_action_str {#1}
150         \tl_if_blank:nF {#2}
151         {
152           \__talk_decode_overlays:nn { actions } {#2}
153           \clist_if_empty:NT \l__talk_decode_overlays_clist
154           { \bool_set_true:N \l__talk_decode_overlays_bool }
155         }
156       }
157       {
158         \msg_error:nnV { talk } { duplicate-action-spec }
159         \l__talk_decode_arg_str
160       }
161     }
162     {
163       \msg_error:nnV { talk } { bad-action-spec }
164       \l__talk_decode_arg_str
165     }
166   }
167 }

```

(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
168 \cs_new_protected:Npn \__talk_decode_overlays:nn #1#2
169 {
170   \str_clear:c { l__talk_decode_ #1 _str }
171   \bool_lazy_and:nnT
172     { \str_if_eq_p:nn {#1} { overlays } }
173     { ! \str_if_eq_p:nn {#2} { 0 } }
174     { \bool_set_true:N \l__talk_decode_mode_bool }

```

```

175     \__talk_decode_overlays:nN {#1} #2 \q_recursion_tail \q_recursion_stop
176     \__talk_decode_check:n {#1}
177   }
178 \cs_new_protected:Npn \__talk_decode_overlays:nN #1#2
179 {
180   \quark_if_recursion_tail_stop:N #2
181   \cs_if_exist_use:cF { __talk_decode_overlay_ #2 :nw }
182   {
183     \str_put_right:cn { l__talk_decode_ #1 _str } {#2}
184     \__talk_decode_overlays:nN
185   }
186   {#1}
187 }
188 \cs_new_protected:cpn { __talk_decode_overlay_+:nw } #1
189 {
190   \bool_set_true:N \l__talk_decode_step_bool
191   \__talk_decode_overlay_aux:nNN {#1} 1
192 }
193 \cs_new_protected:cpn { __talk_decode_overlay_.:nw } #1
194 { \__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 (.

```

195 \cs_new_protected:Npn \__talk_decode_overlay_aux:nNN #1#2#3
196 {
197   \quark_if_recursion_tail_stop_do:Nn #3
198   {
199     \__talk_decode_overlay_offset:nNn {#1} #2 { 0 }
200     \q_recursion_tail \q_recursion_stop
201   }
202   \token_if_eq_meaning:NNTF #3 ( % )
203   { \__talk_decode_overlay_offset:nNn {#1} #2 { } }
204   { \__talk_decode_overlay_offset:nNn {#1} #2 { 0 } #3 }
205 }

```

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.

```

206 \cs_new_protected:Npn \__talk_decode_overlay_offset:nNnN #1#2#3#4
207 {
208   \quark_if_recursion_tail_stop_do:Nn #4
209   {
210     \msg_error:nnV { talk } { bad-action-spec }
211     \l__talk_decode_arg_str
212   } % (
213   \token_if_eq_meaning:NNTF #4 )
214   { \__talk_decode_overlay_offset:nNn {#1} #2 {#3} }
215   { \__talk_decode_overlay_offset:nNnN {#1} #2 {#3#4} }
216 }

```

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

```

217 \cs_new_protected:Npn \__talk_decode_overlay_offset:nNn #1#2#3
218 {

```

```

219 \str_put_right:ce { l__talk_decode_ #1 _str }
220 { \int_max:nn { 0 } { #3 + \g__talk_pauses_int + #2 } }
221 \__talk_decode_overlays:nN {#1}
222 }

```

(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.

```

223 \cs_new_protected:Npn \__talk_decode_check:n #1
224 {
225   \clist_if_empty:cF { l__talk_decode_ #1 _clist }
226   {
227     \msg_error:nnV { talk } { bad-action-spec }
228     \l__talk_decode_arg_str
229   }
230   \clist_set:cv { l__talk_decode_ #1 _clist } { l__talk_decode_ #1 _str }
231   \clist_if_in:cnTF { l__talk_decode_ #1 _clist } { * }
232   { \bool_set_true:c { l__talk_decode_ #1 _bool } }
233   {
234     \clist_map_inline:cn { l__talk_decode_ #1 _clist }
235     { \__talk_decode_check:nw {#1} 0 ##1 - - \q_stop }
236   }
237 }

```

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.

```

238 \cs_new_protected:Npn \__talk_decode_check:nw #1#2 - #3 - #4 \q_stop
239 {
240   \tl_if_empty:nTF {#4}
241   { \__talk_decode_check_single:nn {#1} {#2} }
242   {
243     \tl_if_blank:nTF {#3}
244     { \__talk_decode_check_range:nnn {#1} {#2} { \c_max_int } }
245     { \__talk_decode_check_range:nnn {#1} {#2} {#3} }
246   }
247 }
248 \cs_new_protected:Npn \__talk_decode_check_single:nn #1#2
249 {
250   \int_compare:nNnTF \g__talk_slide_int = {#2}
251   { \bool_set_true:c { l__talk_decode_ #1 _bool } }
252   {
253     \int_compare:nNnTF {#2} > \g__talk_slide_int
254     { \bool_gset_true:N \g__talk_slide_continue_bool }
255     { \bool_set_true:N \l__talk_decode_trailing_bool }
256   }
257 }

```

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.

```

258 \cs_set_protected:Npn \__talk_decode_check_range:nnn #1#2#3
259 {
260   \int_compare:nNnTF \g__talk_slide_int > {#3}
261   { \bool_set_true:N \l__talk_decode_trailing_bool }
262   {
263     \int_compare:nNnTF \g__talk_slide_int < {#2}
264     { \bool_gset_true:N \g__talk_slide_continue_bool }
265     {
266       \bool_set_true:c { l__talk_decode_ #1 _bool }
267       \bool_lazy_and:nnT
268       { \int_compare_p:nNn \g__talk_slide_int < {#3} }
269       { \int_compare_p:nNn {#3} < \c_max_int }
270       { \bool_gset_true:N \g__talk_slide_continue_bool }
271       \clist_map_break:
272     }
273   }
274 }

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

275 \msg_new:nnnn { talk } { bad-action-spec }
276 { Bad~overlay~specification~"#1". }
277 {
278   The~overlay~specification~given~doesn't~follow~the~pattern~described~in~
279   the~ltx-talk-manual:~it~has~been~ignored.
280 }
281 \msg_new:nnnn { talk } { duplicate-action-spec }
282 { Duplicate~action~in~overlay~specification~"#1". }
283 {
284   The~overlay~specification~contains~more~than~one~action:
285   ltx-talk~only~supports~a~single~action~in~one~overlay.
286 }
287 \msg_new:nnnn { talk } { misplaced-action-spec }
288 { Misplaced~action~in~overlay~specification~"#1". }
289 {
290   The~overlay~specification~contains~an~action~before~the~overlay~
291   part(s):~this~is~not~supported.
292 }
293 </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:Nx \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       \tl_if_empty:NF \g__talk_frame_subtitle_tl
361       {
362         \msg_warning:nnV { talk } { subtitle-not-implemented }
363         \g__talk_frame_subtitle_tl
364       }
365     }
366   }
367   \dim_compare:nNnT { \box_ht:N \l__talk_tmp_box + \box_dp:N \l__talk_tmp_box }
368   > \l__talk_header_ht_dim
369   { \msg_warning:nn { talk } { overheight-header } }
370   \box_use_drop:N \l__talk_tmp_box
371   \skip_horizontal:n { \l__talk_header_right_skip - \Gm@rmargin }
372 }
373 \DeclareInstance { header } { std } { talk } { }
374 \AddToHook { begindocument }
375 {
376   \DeclareInstanceCopy { header } { wallpaper } { std }
377   \EditInstance { header } { wallpaper }
378   { print-frame-title = false }
379 }

```

(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.

```

380 \NewTemplateType { footer } { 0 }
381 \DeclareTemplateInterface { footer } { talk } { 0 }
382 {
383   background-color : tokenlist ,
384   color             : tokenlist ,
385   element-order    : commalist ,

```

```

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

```

```

440 {
441   \DeclareInstanceCopy { footer } { wallpaper } { std }
442   \EditInstance { footer } { wallpaper }
443     { element-order = }
444 }

```

(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.

```

445 \cs_new:Npn \__talk_metadata_name:n #1
446 {
447   \tl_if_exist:cTF { @ short #1 }
448     { short #1 }
449     {#1}
450 }

```

(End of definition for __talk_metadata_name:n.)

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

```

451 \cs_new_protected:Npn \__talk_wallpaper_hrule:Nnn #1#2#3
452 {
453   \skip_horizontal:n { -\Gm@lmargin }
454   \tl_if_empty:NF #1
455   {
456     \group_begin:
457       \color_select:V #1
458       \rule:nnn { \paperwidth } {#2} {#3}
459       \skip_horizontal:n { -\paperwidth }
460     \group_end:
461   }
462 }

```

(End of definition for __talk_wallpaper_hrule:Nnn.)

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

```

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

```

```

478         \hfil
479     }
480     \cs_set_nopar:Npn \@oddfoot
481     {
482         \UseInstance { footer } { wallpaper }
483         \hfil
484     }
485     \cs_set_eq:NN \@evenhead \@oddhead
486     \cs_set_eq:NN \@evenfoot \@oddfoot
487 }
488 \cs_new_nopar:Npn \ps@talk
489 {
490     \cs_set_nopar:Npn \@oddhead
491     {
492         \UseInstance { header } { std }
493         \hfil
494     }
495     \cs_set_nopar:Npn \@oddfoot { \UseInstance { footer } { std } }
496     \cs_set_eq:NN \@evenhead \@oddhead
497     \cs_set_eq:NN \@evenfoot \@oddfoot
498 }
499 \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.

```

500 \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.

```

501 \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).

```

502 \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).

```

503 \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 504 \int_new:N \g__talk_frame_int
\theframe 505 \cs_new_eq:NN \c@frame \g__talk_frame_int
\@framenumber 506 \cs_new:Npn \theframe { \@arabic \c@frame }
507 \cs_new:Npn \@framenumber { \arabic { frame } }

```

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

```
\g__talk_appendixframe_int The overall frame number, including LATEX counter-like access.
    \c@appendixframe      508 \int_new:N \g__talk_appendixframe_int
    \theappendixframe      509 \cs_new_eq:NN \c@appendixframe \g__talk_appendixframe_int
    \@appendixframenumber  510 \cs_new:Npn \theappendixframe { \@arabic \c@appendixframe }
                          511 \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.
512 \property_new:nnnn { frames } { shipout } { -1 }
513 { \int_eval:n { \g__talk_frame_int - \g__talk_appendixframe_int } }
514 \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.
515 \property_new:nnnn { totalframes } { shipout } { -1 }
516 { \int_use:N \g__talk_frame_int }
517 \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.
518 \property_new:nnnn { appendixframes } { shipout } { -1 }
519 { \int_use:N \g__talk_appendixframe_int }
520 \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.

```
521 \clist_new:N \l_talk_frame_properties_clist

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

522 \AddToHook { enddocument / afterlastpage }
523 {
524     \property_record:nn { lastpage }
525     { appendixframes , frames , totalframes }
526 }
```

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

```
527 \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
\__talk_frame_process_aux:nn the user or automatically that is handled in an auxiliary.
\__talk_frame_save:nn
\__talk_frame_save:en
528 \cs_new_protected:Npn \__talk_frame_process:nn #1#2
529 {
530   \__talk_if_mode:nT {#1}
531   { \__talk_frame_process_aux:nn {#1} {#2} }
532 }
533 \cs_new_protected:Npn \__talk_frame_process_aux:nn #1#2
534 {
535   \int_gincr:N \g__talk_frame_int
536   \bool_if:NT \l__talk_appendix_bool
537   { \int_gincr:N \g__talk_appendixframe_int }
538   \bool_set_true:N \l__talk_frame_bool
539   \str_if_empty:NF \l__talk_frame_name_str
540   { \__talk_frame_save:en { \l__talk_frame_name_str } {#2} }
541   \bool_if:NTF \l__talk_frame_suppl_bool
542   {
543     \str_case:Vn \l__talk_suppl_mode_str
544     {
545       { appendix }
546       {
547         \int_gincr:N \g__talk_frame_suppl_int
548         \__talk_frame_save:en
549         { appendix . \int_use:N \g__talk_frame_suppl_int }
550         {#2}
551       }
552       { in-place } { \__talk_slide:nn {#1} {#2} }
553     }
554   }
555   { \__talk_slide:nn {#1} {#2} }
556 }
557 \cs_new_protected:Npn \__talk_frame_save:nn #1#2
558 {
559   \tl_if_exist:cT { g__talk_frame_ #1 _tl }
560   { \msg_error:nnn { talk } { duplicate-frame-name } {#1} }
561   \tl_clear_new:c { g__talk_frame_ #1 _tl }
562   \tl_gset:ce { g__talk_frame_ #1 _tl }
563   {
564     {
565       \bool_if:NTF \l__talk_frame_verb_bool
566       { true }
567       { false }
568     }
569     { \exp_not:n {#2} }
570   }
571 }
572 \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.)

\__talk_frame_set:n We need to handle the classical beamer syntax in optional arguments: [<...>]. That
\__talk_frame_set:w can be done while still allowing keyvals not to show values, for example auto-break.
573 \cs_new_protected:Npn \__talk_frame_set:n #1

```

```

574 {
575   \tl_if_head_eq_charcode:nNTF {#1} { < }
576   { \__talk_frame_set:w #1 \q_nil > \q_nil \q_stop {#1} }
577   { \keys_set:nn { talk / frame } {#1} }
578 }
579 \cs_new_protected:Npn \__talk_frame_set:w < #1 > \q_nil #2 \q_stop #3
580 {
581   \tl_if_empty:nTF {#2}
582   { \keys_set:nn { talk / frame } {#3} }
583   { \keys_set:nn { talk / frame } { action-spec = { < #1 > } } }
584 }

```

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

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

```

585 \bool_if:NTF \l__talk_frame_title_bool
586 {
587   \RenewDocumentEnvironment { frame }
588   { D <> { all } 0 { } +m +b }
589   {
590     \str_clear:N \l__talk_frame_name_str
591     \__talk_frame_set:n {#2}
592     \bool_set_false:N \l__talk_frame_verb_bool
593     \__talk_frame_process:nn {#1} { \frametitle {#3} #4 }
594   }
595   { }
596   \NewDocumentEnvironment { frame* }
597   { D <> { all } 0 { } +m c }
598   {
599     \str_clear:N \l__talk_frame_name_str
600     \__talk_frame_set:n {#2}
601     \bool_set_true:N \l__talk_frame_verb_bool
602     \tl_gset:Nn \g__talk_frame_title_tl {#3}
603     \exp_args:Nne \__talk_frame_process:nn {#1}
604     { \tl_to_str:n { \frametitle } \exp_not:n { {#3} #4 } }
605   }
606   { }
607 }
608 {
609   \RenewDocumentEnvironment { frame }
610   { !D <> { all } !0 { } +b }
611   {
612     \str_clear:N \l__talk_frame_name_str
613     \__talk_frame_set:n {#2}
614     \bool_set_false:N \l__talk_frame_verb_bool
615     \__talk_frame_process:nn {#1} {#3}
616   }
617   { }
618   \NewDocumentEnvironment { frame* }
619   { !D <> { all } !0 { } c }
620   {
621     \str_clear:N \l__talk_frame_name_str
622     \__talk_frame_set:n {#2}

```

```

623         \bool_set_true:N \l__talk_frame_verb_bool
624         \__talk_frame_process:nn {#1} {#3}
625     }
626     { }
627 }

```

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

Reusing a frame is largely a question of passing the correct stored information along after setting the flag for re-scanning.

```

\__talk_frame_reuse:nn
\__talk_frame_reuse_aux:nn
\__talk_frame_reuse_aux:nnn
\reuseframe
\againframe
628 \cs_new_protected:Npn \__talk_frame_reuse:nn #1#2
629 {
630     \tl_if_exist:cTF { g__talk_frame_ #2 _tl }
631     {
632         \exp_args:Nv \__talk_frame_reuse_aux:nn
633         { g__talk_frame_ #2 _tl } {#1}
634     }
635     { \msg_error:nnn { talk } { unknown-frame-name } {#2} }
636 }
637 \cs_new_protected:Npn \__talk_frame_reuse_aux:nn #1#2
638 { \__talk_frame_reuse_aux:nnn #1 {#2} }
639 \cs_new_protected:Npn \__talk_frame_reuse_aux:nnn #1#2#3
640 {
641     \use:c { bool_set_ #1 :N } \l__talk_frame_verb_bool
642     \__talk_frame_process:nn {#3} {#2}
643 }
644 \NewDocumentCommand \reuseframe { D <> { all } O { } m }
645 {
646     \group_begin:
647     \__talk_frame_set:n {#2}
648     \str_clear:N \l__talk_frame_name_str
649     \__talk_frame_reuse:nn {#1} {#3}
650     \group_end:
651 }
652 \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.

```

653 \AddToHook { enddocument }
654 {
655     \int_compare:nNnT \g__talk_frame_suppl_int > 0
656     {
657         \bool_if:NF \l__talk_appendix_bool
658         { \appendix }
659         \int_step_inline:nn \g__talk_frame_suppl_int
660         { \__talk_frame_reuse:nn { } { appendix . #1 } }
661     }
662 }
663 \msg_new:nnnn { talk } { overheight-header }
664 { Overheight~header~on~frame~ \int_use:N \g__talk_frame_int ! }
665 {
666     Typesetting~the~frame~header,~there~was~too~much~vertical~space~used.

```

```

667     You~likely~need~to~adjsst~the~overall~space~available~for~frame~titles.
668   }
669   \msg_new:nnnn { talk } { subtitle-not-implemented }
670   { Frame~subtitle~not~printed~on~frame~ \int_use:N \g__talk_frame_int ! }
671   {
672     Typesetting~the~frame~header,~the~frame~subtitle~"#1"~was~given;~
673     at~present,~printing~frame~subtitles~is~not~implemented~in~the~standard~
674     template.~The~text~will~therefore~be~lost.
675   }
676   \msg_new:nnnn { talk } { unknown-frame-name }
677   { Unknown~frame~name~"#1"! }
678   {
679     You~asked~to~reuse~the~contents~of~a~frame~with~the~name~"#1",~
680     but~that~name~was~never~defined.
681   }
682   \msg_new:nnnn { talk } { duplicate-frame-name }
683   { Duplicate~frame~name~"#1"! }
684   {
685     You~asked~to~save~the~contents~of~this~frame~with~the~name~"#1",~
686     but~that~name~has~already~been~used.
687   }
688 </class>

```

Part V

ltx-talk-frame – The structure of frames

1 ltx-talk-frame-structure implementation

Start the DocStrip guards.

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

1.1 Columns

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

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

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

(End of definition for \l__talk_columns_wd_tl.)

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

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

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

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

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

```

28 \par
29 \__talk_action_end:
30 }

```

\l__talk_column_alignment_tl

```

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

```

(End of definition for \l__talk_column_alignment_tl.)

__talk_column_align_bottom:n
 __talk_column_align_center:n
 __talk_column_align_top:n

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

```

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

```

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

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

```

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

```

(End of definition for `\l__talk_vcenter_offset_tl`.)

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

```

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

```

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

```

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

```

1.2 Floats

Well really “not floats at all” but the idea is clear.

`\l__talk_float_alignment_tl` We only worry about horizontal alignment here.

```

103 \tl_new:N \l__talk_float_alignment_tl

```

(End of definition for \l__talk_float_alignment_tl.)

A bit similar to the current approach to lists: we need a template at the start but a common function at the end. The float-placement key is at present just there to allow mopping up of any argument that is given by accident, hence maps to a temporary variable.

```

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

```

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

```

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

```

\endfloatenv And the common end function.

```

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

```

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

figure (env.) Unlike beamer, we allow for overlays for the environments as a whole.

```

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

```

```

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

```

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

`\thefigure`

`\c@table`

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

(End of definition for __talk_if_mode:nTF.)

\mode

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

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

```
13 </class>
```

Part VII

ltx-talk-overlay — Overlays

1 ltx-talk-overlay implementation

Start the DocStrip guards.

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

1.1 Utilities

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

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

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

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

\l__talk_shuffle_skip For tracking.

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

(End of definition for \l__talk_shuffle_skip.)

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

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

```

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

```

(End of definition for __talk_shuffle_skip:n.)

1.2 Opacity utilities

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

```

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

```

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

1.3 Action commands and environments

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

```

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

```

(End of definition for __talk_action_alert:N.)

```

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

```

```

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

```

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

```

\_talk_action_only:N
\_talk_action_only_end:N
\_talk_saved_ifendpe:

```

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

```

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

```

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

```
\l__talk_uncover_hidden_fp
```

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

```

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

```

(End of definition for \l__talk_uncover_hidden_fp.)

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

```

100 \cs_new_protected:Npn \__talk_action_uncover:N #1
101   {
102     \bool_if:NTF #1
103       { \__talk_opacity_begin:n { 1 } }
104       { \UseInstance { hidden } { std } }
105   }
106 \cs_new_protected:Npn \__talk_action_uncover_end:N #1
107   { \__talk_opacity_end: }

```

(End of definition for __talk_action_uncover:N and __talk_action_uncover_end:N.)

\invisible All generated automatically using the above implementations.

```

\uncover 108 \clist_map_inline:nn { invisible , uncover , visible }
\visible 109   {
110     \ExpandArgs { cne } \NewDocumentCommand { #1 }
111       { > { \__talk_overlay_arg:n } D <> { all } +m }
112       {
113         \exp_not:c { __talk_action_ #1 :N } ##1
114         ##2
115         \exp_not:c { __talk_action_ #1 _end:N } ##1
116       }

```

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

invisibleenv (env.) And the environment versions.

```

uncoverenv (env.) 117 \ExpandArgs { nnee } \NewDocumentEnvironment { #1 env }
visibleenv (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 tabular cells.

```

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

```

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

\g__talk_onslide_tl

```

239 \tl_new:N \g__talk_onslide_tl

```

(End of definition for \g__talk_onslide_tl.)

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

```

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

```

```

248     }
249 }

```

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

\pause A thin wrapper.

```

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

```

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

1.5 Fixed-size areas

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

```

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

```

(End of definition for __talk_overprint_begin:n.)

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

```

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

```

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

```

279 \int_new:N \l__talk_overprint_int

```

(End of definition for \l__talk_overprint_int.)

`\__talk_frame_overprint:` To refer to the current overprint environment within the document: needed in the `.aux` so avoids using non-letters.

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

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

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

`\__talk_overprint_check_ht:n`

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

As there is no clear end-point for overprinting, we need to be careful to keep the current width separate from the saved one. The rest is then about saving to the `.aux` file and helping out the user.

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

```

323 \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 Lua_T_EX due to restrictions on Xform structures in pdf_T_EX. Here, we apply the collection code to the commands not the environment forms to deal with “direct” usage.

```

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

```

Part VIII

ltx-talk-required – “Required” definitions

1 ltx-talk-required implementation

Start the DocStrip guards.

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

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

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

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

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

1.1 Standard design settings

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

```

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

```

1.2 List support

```

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

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

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

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

<code>\section</code> <code>\subsection</code> <code>\subsubsection</code> <code>\insertsection</code> <code>\insertsubsection</code> <code>\insertsubsubsection</code> <code>__talk_head_section:Nnn</code> <code>__talk_head_subsection:Nnn</code> <code>__talk_head_subsubsection:Nnn</code>	The sectioning commands all have essentially the same form: we therefore create using a generator with the necessary conditionals in place. As we do not typeset sections at this stage, the code is quite different from article. This also means that the bookmark links need to point forward to the next slide: if that doesn't appear, the bookmarks will be out. Using the general scratch sequence here should be OK: it really is a one-off setting. We need a sequence to allow indexed mapping to avoid any extra setup for the depth value.	<pre> 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_mode:nT {##2} 156 { \exp_not:c { __talk_head_ #1 :Nnn } ##1 {##3} {##4} } 157 } 158 } 159 } </pre>
--	---	--

```

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 _ t1 } } {#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:nNnF {#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
442 \AddToHook { item / after } { \l__talk_list_end_tl }

```

(End of definition for `\l__talk_list_end_tl`.)

`\__talk_block_set:n`
`\__talk_block_set:w`

We need to handle the classical beamer syntax in optional arguments: `[<...>]`. The code here uses the same idea as the function for frames, but with only one key set up and the need to return a value, things are be done slightly differently.

```

443 \cs_new_protected:Npn \__talk_block_set:n #1
444 {
445   \tl_clear:N \l__talk_tmp_tl
446   \tl_if_head_eq_charcode:nNT {#1} { < }
447   { \__talk_block_set:w #1 \q_nil > \q_nil \q_stop }
448 }
449 \cs_new_protected:Npn \__talk_block_set:w < #1 > \q_nil #2 \q_stop
450 {
451   \tl_if_empty:nF {#2}
452   {
453     \keys_set_known:nnN { talk / block } { action-spec = { < #1 > } }
454     \l__talk_tmp_tl
455   }
456 }

```

(End of definition for `\__talk_block_set:n` and `\__talk_block_set:w`.)

`itemize (env.)` Allow for the classical beamer syntax.
`enumerate (env.)` 457 \AddToHook { begindocument / before }
`description (env.)`

```

458 {
459   \clist_map_inline:nn { itemize , enumerate , description }
460   {
461     \RenewDocumentEnvironment {#1} { !o }
462     {
463       \IfNoValueTF {##1}
464       { \SimpleBlockEnv {#1} { } }
465       {
466         \__talk_block_set:n {##1}
467         \ExpandArgs { nV } \SimpleBlockEnv {#1} \l__talk_tmp_tl
468       }
469     }
470     { \BlockEnvEnd }
471   }
472 }

```

And add the structural color to item labels.

```

473 \AddToHook { begindocument / before }
474 {
475   \EditInstance { item } { basic }
476   { label-format = \color { structure } #1 }
477   \EditInstance { item } { description }
478   { label-format = \normalfont \bfseries \color { structure } #1 }
479 }

```

`\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.

```

480 \keys_define:nn { talk / block }
481 { 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.

```

482 \NewCommandCopy \stdnewtheorem \newtheorem
483 \RenewDocumentCommand \newtheorem { s m o m o }
484 {
485   \use:e
486   {
487     \exp_not:N \stdnewtheorem
488     \IfBooleanT #1 { * }
489     { \exp_not:n {#2} }
490     \IfNoValueF {#3} { \exp_not:n { [ {#3} ] } }
491     { \exp_not:n {#4} }
492     \IfNoValueF {#5} { \exp_not:n { [ {#5} ] } }
493   }
494   \NewEnvironmentCopy { std #2 } {#2}
495   \RenewDocumentEnvironment {#2} { D <> { all } o }
496   {
497     \__talk_action_begin:n {##1}

```

```

498     \IfNoValueTF {##2}
499     { \begin { std #2 } }
500     { \begin { std #2 } [ {##2} ] }
501     \ignorespaces
502   }
503   {
504     \end { std #2 }
505     \__talk_action_end:
506   }
507 }

```

(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.

```

508 \NewDocumentEnvironment { thebibliography } { m }
509 {
510   \list { \@biblabel { \@arabic \c@enumiv } }
511   {
512     \settowidth \labelwidth { \@biblabel {#1} }
513     \leftmargin \labelwidth
514     \advance \leftmargin \labelsep
515     \@openbib@code
516     \usecounter { enumiv }
517     \let \p@enumiv \empty
518     \renewcommand \theenumiv { \@arabic \c@enumiv }
519   }
520   \sloppy
521   \clubpenalty 4000
522   \@clubpenalty \clubpenalty
523   \widowpenalty 4000
524   \sfcode `\. \@m
525 }
526 {
527   \def \@noitemerr
528   { \@latex@warning { Empty~`thebibliography'~environment } }
529   \endlist
530 }
531 \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`

```

532 \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.

```

533 \NewDocumentCommand \appendix { D <> { all } }
534 {
535   \__talk_if_overlay:nT {#1}

```

```

536     { \bool_set_true:N \l__talk_appendix_bool }
537     \addtocontents { toc } { \appendixmarker }
538     \cs_gset_eq:NN \contentsline \use_none:nnnn
539 }

```

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

```

540 \</class>

```

Part X

ltx-talk-title – Title pages

1 ltx-talk-title implementation

Start the DocStrip guards.

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

```
\@author We create a set of keys and variables in one go. Following the classical kernel approach,
\@date    all of the underlying storage is global. The short values will always be set in the following
\@institute code so can be used automatically anywhere we might want them.
\@subtitle 3 \clist_map_inline:nn
\@title    4 { author , date , institute , subtitle , title }
\@shortauthor 5 {
\@shortdate 6 \keys_define:nn { talk / metadata }
\@shortinstitute 7 {
\@shortsubtitle 8 #1 .tl_gset:c = @ #1 ,
\@shorttitle 9 short- #1 .tl_gset:c = @short #1
10 }
11 }
```

Allow empty values for author and title.

```
12 \tl_gclear:N \@author
13 \tl_gclear:N \@title
```

As the date has a standard value, that has to be propagated.

```
14 \tl_gset_eq:NN \@shortdate \@date
```

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

```
\author Slightly repetitive but as we need to handle the tagging aspects, this is easier than using
\date    a loop. The main aim is to add the short metadata concept. Notice that keys are set
\title   before the main data storage in case someone set the value as a key as well as a mandatory
         argument.
```

```
15 \RenewDocumentCommand \author { = { short-author } 0 { {#2} } m }
16 {
17   \keys_set:nn { talk / metadata } {#1}
18   \tl_gset:Nn \@author {#2}
19   \tl_gset_eq:NN \g__tag_title_author_tl \@author
20   \keys_set_known:nn { hyp } {#1}
21 }
22 \RenewDocumentCommand \date { = { short-date } 0 { {#2} } m }
23 {
24   \keys_set:nn { talk / metadata } {#1}
25   \tl_gset:Nn \@date {#2}
26 }
27 \RenewDocumentCommand \title { = { short-title } 0 { {#2} } m }
28 {
29   \keys_set:nn { talk / metadata } {#1}
```

```

30 \tl_gset:Nn \@title {#2}
31 \tl_gset_eq:NN \g__tag_title_title_tl \@title
32 \keys_set_known:nn { hyp } {#1}
33 }

```

(End of definition for \author, \date, and \title. These functions are documented on page ??.)

\institute Simple storage at present: unlike some of the kernel data, there is not a lot to do here.

```

\subtitle
34 \NewDocumentCommand \institute { = { short-institute } 0 { {#2} } m }
35 {
36 \keys_set:nn { talk / metadata } {#1}
37 \tl_gset:Nn \@institute {#2}
38 }
39 \NewDocumentCommand \subtitle { = { short-subtitle } 0 { {#2} } m }
40 {
41 \keys_set:nn { talk / metadata } {#1}
42 \tl_gset:Nn \@subtitle {#2}
43 }

```

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

\l__talk_titlelem_after_skip \l__talk_titlelem_before_skip \l__talk_titlelem_color_tl \l__talk_titlelem_font_tl \l__talk_titlelem_tag_begin_tl \l__talk_titlelem_tag_end_tl As the various elements of the titlepage share certain characteristics, we use a single template and split them as instances.

```

44 \NewTemplateType { titlepage-element } { 1 }
45 \DeclareTemplateInterface { titlepage-element } { talk } { 1 }
46 {
47 after-skip : length = 0em ,
48 before-skip : length = 0em ,
49 color : tokenlist = . ,
50 font : tokenlist = \normalfont ,
51 tag-begin : tokenlist = ,
52 tag-end : tokenlist =
53 }
54 \DeclareTemplateCode { titlepage-element } { talk } { 1 }
55 {
56 after-skip = \l__talk_titlelem_after_skip ,
57 before-skip = \l__talk_titlelem_before_skip ,
58 color = \l__talk_titlelem_color_tl ,
59 font = \l__talk_titlelem_font_tl ,
60 tag-begin = \l__talk_titlelem_tag_begin_tl ,
61 tag-end = \l__talk_titlelem_tag_end_tl
62 }
63 {
64 \tl_if_empty:nF {#1}
65 {
66 \vspace { \l__talk_titlelem_before_skip }
67 \group_begin:
68 \tl_if_empty:NF \l__talk_titlelem_color_tl
69 { \color_select:V \l__talk_titlelem_color_tl }
70 \l__talk_titlelem_font_tl
71 \l__talk_titlelem_tag_begin_tl
72 #1
73 \par
74 \l__talk_titlelem_tag_end_tl

```

```

75         \group_end:
76         \vspace { \l__talk_titlelem_after_skip }
77     }
78 }

```

Standard settings are taken from beamer with minor adjustments.

```

79 \DeclareInstance { titlepage-element } { author } { talk }
80 { before-skip = 1em }
81 \DeclareInstance { titlepage-element } { date } { talk }
82 { after-skip = 0.5em }
83 \DeclareInstance { titlepage-element } { institute } { talk }
84 { font = \scriptsize }
85 \DeclareInstance { titlepage-element } { subtitle } { talk }
86 { before-skip = 0.25em , color = structure }
87 \DeclareInstance { titlepage-element } { title } { talk }
88 {
89     color = structure ,
90     font = \Large ,
91     tag-begin = \tag_struct_begin:n { tag = Title } ,
92     tag-end = \tag_struct_end:
93 }

```

(End of definition for \l__talk_titlelem_after_skip and others.)

Here, we deal with the overall style: notice that frame vertical alignment actually applies elsewhere, which is why it doesn't show up in the template code part. As a result, we have a slightly repetitive key interface.

```

\l__talk_titlepage_order_clist
\l__talk_titlepage_alignment_tl
\l__talk_titlepage_framestyle_tl
\l__talk_frame_alignment_tl
79 \NewTemplateType { titlepage } { 0 }
80 \DeclareTemplateInterface { titlepage } { talk } { 0 }
81 {
82     element-order : commalist =
83     {
84         title      ,
85         subtitle   ,
86         author     ,
87         institute  ,
88         date
89     } ,
90     frame-style : tokenlist = talk ,
91     horizontal-alignment : choice { left , center , right } = center ,
92     vertical-alignment : choice { bottom , center , stretch , top } = center
93 }
94 \DeclareTemplateCode { titlepage } { talk } { 0 }
95 {
96     element-order = \l__talk_titlepage_order_clist ,
97     frame-style = \l__talk_titlepage_framestyle_tl ,
98     horizontal-alignment =
99     {
100         left = \tl_set:Nn \l__talk_titlepage_alignment_tl { flushleft } ,
101         center = \tl_set:Nn \l__talk_titlepage_alignment_tl { center } ,
102         right = \tl_set:Nn \l__talk_titlepage_alignment_tl { flushright }
103     } ,
104     vertical-alignment =
105     {
106         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		bool commands:	
\(	77	\bool_do_until:nn	125
\)	77	\bool_do_while:Nn	30
\.	524	\bool_gset_eq:NN	105
@ commands:		\bool_gset_false:N	37, 104
\@_decode_overlay_+:nw	168	\bool_gset_true:N	103, 254, 264, 270
@starttoc@aux internal commands:		\bool_if:NTF	6,
\starttoc@aux__talknn	253		6, 44, 46, 48, 50, 50, 58, 66, 68, 71,
\\	350		72, 78, 80, 83, 89, 102, 141, 145,
			147, 153, 174, 187, 200, 204, 229,
			245, 257, 355, 536, 541, 565, 585, 657
A		\bool_lazy_and:nnTF	
\abovecaptionskip	156, 158		21, 56, 58, 171, 267, 338
\action	144	\bool_lazy_and_p:nn	323, 328
actionenv (env.)	144	\bool_lazy_or:nnTF	18, 23, 321, 425
\addcontentsline	214	\bool_lazy_or_p:nn	24
\addcontentslinebookmark	195, 207	\bool_new:N	3, 3, 4, 8, 9, 10, 15,
\addcontentslinebookmarkOff	195, 212		140, 142, 143, 232, 500, 502, 503, 532
\addcontentslinebookmarkOn	195	\bool_set_eq:NN	100, 179, 183
\addcontentslinebookmarkReset	197	\bool_set_false:N	32,
\addtocontents	537		33, 34, 35, 36, 46, 172, 179, 592, 614
\AddToHook	54, 263, 367, 374, 405,	\bool_set_true:N	27, 28, 62, 63, 80,
	439, 442, 450, 451, 457, 473, 522, 653		84, 85, 121, 154, 170, 174, 190, 232,
\addtolength	48	\c_false_bool	251, 255, 261, 266, 536, 538, 601, 623
\advance	514	\c_true_bool	15
\againframe	628		14, 196, 213
\alert	46, 122	box commands:	
alertenv (env.)	129	\box_dp:N	63, 63, 367
\alt	217	\box_gclear:N	98
\and	131	\box_gset_eq:NN	122
\appendix	533, 658	\box_ht:N	
\appendixmarker	227, 537		58, 59, 62, 68, 111, 296, 321, 367
\arabic	507, 511	\box_if_empty:NTF	141
\arraycolsep	25	\box_if_empty_p:N	125
\arrayrulewidth	27	\box_new:N	5, 6, 7, 66, 171
\AssignSocketPlug	194	\box_set_dp:Nn	60, 66
\AssignStructureRole	125, 127	\box_set_eq:NN	119, 124
\AssignTaggingSocketPlug	252	\box_set_ht:Nn	55, 60
\author	15	\box_use:N	401
B		\box_use_drop:N	26, 66, 70,
\begin	122, 126, 130, 144, 375, 435, 499, 500		71, 76, 130, 274, 301, 305, 307, 370, 435
\begingroup	163, 255, 412	\box_wd:N	40, 45
\belowcaptionskip	157, 159	box internal commands:	
\bfseries	21, 39, 285, 392, 478	__box_dim_eval:n	37, 40, 45, 48
\bigskipamount	18	__box_set_to_wd:	44, 49
block (env.)	384	__box_set_to_wd:NNnn	30, 32, 33
\BlockEnvEnd	470		

C

`\clearpage` 155
 clist commands:
 `\clist_clear:N` 37, 38, 39, 40
 `\clist_const:Nn` 70
 `\clist_if_empty:NnTF` 153, 225
 `\clist_if_in:NnTF` 78, 231
 `\clist_map_break:` 271
 `\clist_map_inline:Nn` ... 132, 234, 422
 `\clist_map_inline:nn`
 3, 97, 108, 139, 369, 459
 `\clist_new:N` 12, 16, 67, 521
 `\clist_pop:NNTF` 418
 `\clist_put_left:Nn` 150
 `\clist_remove_duplicates:N` 151
 `\clist_set:Nn` 230
 `\clist_set_eq:NN` 149
`\clubpenalty` 521, 522
`\color` 4, 11, 56, 285, 393, 476, 478
 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, 416, 457
 `\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, 538
 cs commands:
 `\cs_generate_variant:Nn` 7,
 8, 9, 10, 60, 62, 63, 64, 217, 273, 572
 `\cs_gset:Npn` 136, 137, 138
 `\cs_gset_eq:NN` 56, 88, 230, 231, 234, 538
 `\cs_gset_nopar:Npn` 49, 57, 64
 `\cs_gset_protected:Npn`
 29, 31, 33, 42, 52, 225, 253, 345
 `\cs_if_exist:NnTF` 145, 291
 `\cs_if_exist_p:N` 339
 `\cs_if_exist_use:NnTF` ... 181, 189, 206
 `\cs_new:Npn` 7, 10, 34, 35,
 36, 37, 38, 39, 40, 41, 42, 185, 280,
 445, 506, 507, 510, 511, 514, 517, 520
 `\cs_new_eq:NN` 6,
 9, 92, 195, 224, 233, 427, 505, 509
 `\cs_new_nopar:Npn` 3, 488
 `\cs_new_protected:Npe` 76, 98, 124

`\cs_new_protected:Npn` 11, 12, 18, 18,
 20, 34, 38, 40, 41, 42, 44, 48, 52, 54,
 54, 54, 56, 56, 62, 64, 65, 68, 69, 73,
 76, 78, 82, 92, 100, 106, 108, 111,
 117, 131, 134, 134, 155, 157, 159,
 163, 165, 167, 168, 169, 175, 176,
 178, 181, 186, 188, 190, 193, 195,
 195, 198, 199, 206, 214, 217, 218,
 219, 223, 228, 238, 241, 248, 261,
 266, 272, 274, 284, 286, 296, 307,
 310, 314, 319, 336, 339, 378, 410,
 421, 423, 434, 436, 443, 449, 451,
 528, 533, 557, 573, 579, 628, 637, 639
`\cs_set:Npn` 14, 15, 85
`\cs_set_eq:NN` ... 61, 62, 63, 64, 70,
 179, 470, 471, 485, 486, 496, 497, 531
`\cs_set_nopar:Npn`
 130, 181, 182, 183, 184,
 463, 465, 469, 473, 475, 480, 490, 495
`\cs_set_protected:Npn`
 44, 125, 131, 160, 258
`\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, 373, 438
`\DeclareInstanceCopy` 376, 441
`\DeclareTemplateCode`
 ... 23, 54, 96, 109, 110, 289, 328, 391
`\DeclareTemplateInterface`
 ... 16, 45, 94, 95, 105, 282, 318, 381
`\def` 527
`\definecolor` 69
`description (env.)` 363, 457
 dim commands:
 `\dim_compare:nNnTF` 294, 367
 `\dim_compare_p:nNn` 59, 341
 `\dim_const:Nn` 122, 128
 `\dim_eval:n` 55, 56, 57, 57, 62
 `\dim_max:nn` 62, 320
 `\dim_set:Nn` 19, 86
 `\dim_set_eq:NN` 20, 87
 `\dim_to_decimal:n` 115
 `\dim_use:N` 143, 144
`\DocumentMetadata` 7
`\doublerulesep` 28

E

`\EditInstance` 377, 442, 475, 477

<code>\emph</code>	353	<code>\ExpandArgs</code>	110, 117, 134, 139, 357, 368, 369, 420, 427, 467
<code>\end</code>	136, 137, 137, 147, 379, 438, 504		
<code>\endcsname</code>	167, 265, 266		
<code>\endfloatenv</code>	134, 147		
<code>\endgroup</code>	169, 270, 425		
<code>\endlist</code>	529		
<code>enumerate (env.)</code>	457		
environments:			
<code>actionenv</code>	144		
<code>alertenv</code>	129		
<code>block</code>	384		
<code>column</code>	78		
<code>columns</code>	11		
<code>description</code>	363, 457		
<code>enumerate</code>	457		
<code>figure</code>	139		
<code>invisibleenv</code>	117		
<code>itemize</code>	457		
<code>onlyenv</code>	137		
<code>overlayarea</code>	268		
<code>overprint</code>	285		
<code>quotation</code>	363		
<code>quote</code>	363		
<code>stdquotation</code>	363		
<code>stdquote</code>	363		
<code>stdverse</code>	363		
<code>table</code>	139		
<code>uncoverenv</code>	117		
<code>verse</code>	363		
<code>visibleenv</code>	117		
exp commands:			
<code>\exp_after:wN</code>	160, 316, 415		
<code>\exp_args:Ne</code>			
.....	19, 66, 136, 238, 257, 441, 445		
<code>\exp_args:Nne</code>	603		
<code>\exp_args:No</code>	41		
<code>\exp_args:NV</code>	129, 188		
<code>\exp_args:Nv</code>	632		
<code>\exp_args_generate:n</code>	61		
<code>\exp_not:N</code> ..	40, 42, 43, 50, 52, 53, 55, 60, 78, 80, 81, 82, 84, 85, 89, 91, 92, 98, 101, 104, 106, 109, 113, 115, 119, 120, 120, 125, 126, 126, 127, 128, 130, 131, 131, 132, 134, 135, 139, 141, 142, 144, 145, 146, 147, 147, 148, 150, 153, 156, 159, 161, 168, 172, 174, 176, 177, 178, 179, 183, 185, 186, 187, 188, 189, 327, 339, 372, 373, 374, 395, 431, 435, 487		
<code>\exp_not:n</code>	48, 329, 397, 398, 489, 490, 491, 492, 569, 604		
<code>\exp_stop_f:</code>	55, 56, 57		
<code>\expandafter</code>	265		
		F	
		<code>\fboxrule</code>	32
		<code>\fboxsep</code>	31
		<code>\fi</code>	18, 165
		<code>figure (env.)</code>	139
		<code>\figurename</code>	150
		file commands:	
		<code>\file_if_exist_input:nTF</code>	115
		<code>\file_input:n</code>	117
		fnote commands:	
		<code>\fnote_makefnstext:n</code>	195
		<code>\font</code>	77
		<code>\fontchardp</code>	77
		<code>\fontcharht</code>	77
		<code>\footins</code>	30
		<code>\footnote</code>	173, 174
		<code>\footnoterule</code>	21, 143
		<code>\footnotesize</code>	303
		<code>\footskip</code>	405, 406
		fp commands:	
		<code>\fp_eval:n</code>	120
		<code>\fp_to_dim:n</code>	121, 128, 130
		<code>\frame</code>	31, 29, 527
		<code>frame</code>	585
		<code>frame*</code>	585
		<code>\framesubtitle</code>	10
		<code>\frametitle</code>	5, 593, 604
		G	
		<code>\gdef</code>	327
		<code>\geometry</code>	4
		group commands:	
		<code>\group_begin:</code> ..	14, 33, 44, 67, 84, 85, 124, 146, 157, 280, 343, 389, 456, 646
		<code>\c_group_begin_token</code>	47
		<code>\group_end:</code> ..	32, 41, 51, 75, 88, 88, 127, 150, 215, 282, 358, 397, 460, 650
		<code>\group_insert_after:N</code>	49
		H	
		hbox commands:	
		<code>\hbox:n</code>	130, 300
		<code>\hbox_set:Nw</code>	45, 72
		<code>\hbox_set_end:</code>	25, 75, 86
		<code>\hbox_set_to_wd:Nnw</code>	18
		<code>\headsep</code>	323, 342, 343
		<code>\hfil</code>	83, 389, 433, 467, 478, 483, 493
		hook commands:	
		<code>\hook_gput_code:nnn</code>	60, 119, 333
		<code>\hook_new:n</code>	192
		<code>\hook_use:n</code>	190

\hspace	299, 307	\keys_define:nn	3, 5, 6, 31, 69, 89, 97, 99, 233, 480
\hypersetup	187	\keys_set:nn	7, 17, 17, 24, 29, 36, 41, 50, 84, 106, 173, 256, 577, 582, 583
I			
\IfBooleanT	396, 488	\keys_set_known:nn	20, 32
\ifcase	5	\keys_set_known:nnN	453
\IfFormatAtLeastF	7	\l_keys_value_tl	46, 253
\IfNoValueF	397, 398, 490, 492	L	
\IfNoValueTF	15, 25, 36, 47, 67, 177, 254, 410, 413, 463, 498	\label	102, 403
\ignorespaces	19, 21, 92, 167, 226, 266, 376, 501	\labelenumi	34
\immediate	266	\labelenumii	35
\includegraphics	387	\labelenumiii	36
\insertsection	139	\labelenumiv	37
\insertsubsection	139	\labelitemfont	38, 39, 40, 41, 42
\insertsubsubsection	139	\labelitemi	38
\institute	34	\labelitemii	39
int commands:		\labelitemiii	40
\int_compare:nNnTF	82, 222, 250, 253, 260, 263, 309, 341, 431, 655	\labelitemiv	41
\int_compare_p:nNn	268, 269, 324, 325, 329, 330	\labelsep	29, 33, 46, 48, 514
\int_eval:n	258, 432, 513	\labelwidth	47, 48, 512, 513
\int_gincr:N	36, 51, 81, 255, 440, 535, 537, 547	\Large	21, 90
\int_gset:Nn	184, 256	\large	392
\int_gset_eq:NN	27, 212, 217, 222	\leavevmode	89, 391
\int_gzero:N	16, 28, 95	\leftmargin	51, 59, 66, 513, 514
\int_incr:N	289	\leftmargini	43, 47, 51
\int_max:nn	220	\leftmarginii	44, 59
\int_new:N	5, 8, 9, 10, 200, 211, 228, 279, 433, 501, 504, 508	\leftmarginiii	45, 66
\int_set_eq:NN	15	\leftskip	291, 301
\int_step_inline:nn	659	legacy commands:	
\int_to_Roman:n	282	\legacy_if:nTF	50, 252, 261, 323
\int_to_roman:n	283	\let	517
\int_use:N	11, 17, 81, 136, 152, 442, 446, 516, 519, 549, 664, 670	\list	510
\c_max_int	244, 269	M	
\invisible	108	\makeatletter	256
invisibleenv (env.)	117	\maketitle	139
iow commands:		\mathcolor	5, 11
\iow_now:Nn	325	\mbox	381
\item	69, 405	\medskip	390, 396
itemize (env.)	457	\mode	11
\itemsep	54, 62, 69	mode commands:	
J		\mode_if_horizontal:TF	31
\jobname	260, 266	\mode_if_horizontal_p:	25
K		\mode_if_math:TF	380
keys commands:		\mode_if_vertical_p:	26
\l_keys_choice_str	82, 85	\mode_leave_vertical:	32, 383, 439
		\month	5
		msg commands:	
		\msg_error:nnn	140, 158, 163, 210, 227, 560, 635
		\msg_fatal:nn	15
		\g_msg_module_name_prop	5
		\g_msg_module_type_prop	6

\msg_new:nnn	22, 348	\parsep	53, 61, 62, 68, 69
\msg_new:nnnn		\partopsep	71
..	9, 275, 281, 287, 663, 669, 676, 682	\pause	47, 250
\msg_warning:nn	27, 344, 369	pdfxform commands:	
\msg_warning:nnn	362	\pdfxform_new:nnn	441
		\pdfxform_use:n	445
N			
\NeedsDocumentMetadata	17	prg commands:	
\NewCommandCopy	4, 5, 6, 173, 368, 388, 407, 482, 527, 652	\prg_generate_conditional_-variant:Nnn	10, 10
\newcounter	23, 130, 131, 132, 150	\prg_new_protected_conditional:Npnn	3, 3
\NewDocumentCommand	5, 10, 11, 34, 39, 65, 110, 122, 132, 139, 144, 145, 150, 217, 223, 227, 240, 250, 278, 533, 644	\prg_return_false:	8, 8
\NewDocumentEnvironment	11, 78, 117, 129, 137, 141, 152, 268, 285, 363, 364, 365, 366, 384, 508, 596, 618	\prg_return_true:	7, 7
\NewEnvironmentCopy	371, 494	\ProcessedArgument	14, 15
\newlabel	416	\ProcessKeyOptions	114
\newlength	156, 157	prop commands:	
\NewSocketPlug	181	\prop_gput:Nnn	5, 6
\NewTaggingSocket	235	property commands:	
\NewTaggingSocketPlug	236	\property_new:nnnn	11, 512, 515, 518
\NewTemplateType	15, 44, 93, 94, 104, 281, 317, 380	\property_record:nn	152, 524
\newtheorem	69, 482	\property_ref:nn	17, 514, 517, 520
\newwrite	265	\protect	223
\noindent	31, 339, 344, 402	\providecommand	195, 196, 197, 198
\normalfont	42, 50, 322, 392, 478	\ProvidesExplClass	3
\normalsize	166	\put	57
\number	20, 22		
\numberline	223	Q	
		\quad	131
O			
\obeyedline	19, 85	quark commands:	
\only	51, 132	\q_nil	447, 449, 576, 579
onlyenv (env.)	137	\quark_if_recursion_tail_stop:N	180
\onslide	47, 223	\quark_if_recursion_tail_stop_-do:Nn	197, 208
opacity commands:		\quark_if_recursion_tail_stop_-do:nn	56
\opacity_begin:n	39, 334	\q_recursion_stop	47, 175, 200
\opacity_end:	41, 336	\q_recursion_tail	47, 175, 200
\openout	266	\q_stop	92, 101, 126, 127, 132, 135, 161, 165, 235, 238, 317, 319, 416, 421, 447, 449, 576, 579
\or ..	5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16	quotation (env.)	363
overlayarea (env.)	268	quote (env.)	363
overprint (env.)	285		
P			
\pagecolor	43	R	
\pagestyle	499	\raggedright	90, 156, 265
\paperheight	57, 58	\refstepcounter	161
\paperwidth	58, 347, 410, 458, 459	\relax	266
\par	36, 38, 14, 28, 38, 71, 73, 80, 95, 100, 162, 168, 263, 277, 308, 357, 387, 395, 402	\relsize	120
		\renewcommand	518
		\RenewCommandCopy	29
		\RenewDocumentCommand	11, 15, 21, 22, 27, 30, 43, 174, 369, 389, 403, 408, 428, 483
		\RenewDocumentEnvironment	372, 461, 495, 587, 609

tag internal commands:	
\g__tag_title_author_tl	19
\g__tag_title_title_tl	31
\tagpdfparaOff	86
\tagpdfsetup	168, 188
talk commands:	
\l__talk_frame_properties_clist .	149, 521
talk internal commands:	
__talk_action_alert:N 42, 42, 125, 130	
__talk_action_begin:n 13, 91, 143,	144, 147, 153, 155, 374, 386, 438, 497
__talk_action_begin:w . 144, 160, 165	
__talk_action_begin_auxi:n	144, 163, 166, 167
__talk_action_begin_auxii:n	144, 173, 176
__talk_action_end: 38, 29, 96, 144,	148, 149, 154, 198, 380, 403, 434, 505
__talk_action_invisible:N 48, 48, 196	
__talk_action_invisible_end:N .	48, 54, 213
__talk_action_only:N 64, 64, 138	
__talk_action_only_end:N 64, 78, 139	
\l__talk_action_spec_str	161, 232, 416, 480
__talk_action_uncover:N 100, 100, 189	
__talk_action_uncover_end:N	100, 106, 191
__talk_action_visible:N 48, 56	
__talk_action_visible_end:N . 48, 62	
\l__talk_appendix_bool	229, 532, 536, 536, 657
\g__talk_appendixframe_int	508, 513, 519, 537
\l__talk_aspect_ratio_str . . 69, 134	
__talk_block_set:n 443, 443, 466	
__talk_block_set:w 443, 447, 449	
\g__talk_break_box 5, 122, 124	
\l__talk_break_box	6, 119, 120, 122, 124, 125, 127
\g__talk_cnt_reset_seq	133, 134, 135, 201, 216, 221, 229
__talk_cnt_restore: 79, 214, 219	
__talk_cnt_save: 99, 214, 214	
__talk_column_align_bottom:n 54, 54	
__talk_column_align_center:n 54, 56	
__talk_column_align_top:n . . 54, 73	
\l__talk_column_alignment_tl . 31, 98	
\g__talk_column_int 9, 15, 16, 27, 81, 82	
\l__talk_column_int 9, 15, 27	
\l__talk_columns_wd_tl 5, 18, 19	
__talk_decode_action:n 115, 124, 124	
__talk_decode_action:w 124, 126, 131	
\l__talk_decode_action_str	14, 22, 137, 147, 149, 182, 190
\l__talk_decode_actions_bool	15, 32, 184, 192
\l__talk_decode_actions_clist 15, 37	
\l__talk_decode_actions_str . . 15, 38	
\l__talk_decode_arg_str	11, 31, 41, 141, 159, 164, 211, 228
__talk_decode_check:n . 176, 223, 223	
__talk_decode_check:nw 223, 235, 238	
__talk_decode_check_range:nnn .	223, 244, 245, 258
__talk_decode_check_single:nn .	223, 241, 248
__talk_decode_mode:n 66, 76, 76	
__talk_decode_mode:nn . 106, 109, 111	
__talk_decode_mode:w 76, 91, 98	
__talk_decode_mode_aux:n 76	
\l__talk_decode_mode_bool	6, 8, 27, 35, 62, 84, 174
\l__talk_decode_modes_bool	9, 36, 60, 80, 121
__talk_decode_overlay_:nw 168	
__talk_decode_overlay_aux:nNN .	168, 191, 194, 195
__talk_decode_overlay_offset:nNn	168, 199, 204, 214, 217
__talk_decode_overlay_offset:nNnN	168, 203, 206, 215
__talk_decode_overlays:nN	168, 175, 178, 184, 221
__talk_decode_overlays:nn	117, 138, 152, 168, 168
\l__talk_decode_overlays_bool . 3,	6, 28, 33, 63, 85, 154, 180, 187, 189, 191
\l__talk_decode_overlays_clist .	12, 39, 153
\l__talk_decode_overlays_str	12, 40, 59, 116
__talk_decode_parse:n	5, 5, 18, 18, 178, 188
__talk_decode_parse:w . 18, 47, 54, 68	
__talk_decode_parse_auxi:n	18, 19, 20
__talk_decode_parse_auxii:n	18, 41, 44
\l__talk_decode_step_bool	10, 46, 48, 190
\l__talk_decode_trailing_bool . .	4, 34, 245, 255, 261
\l__talk_float_alignment_tl	103, 115, 116, 117, 126, 136
\l__talk_fontsize_dim 69, 115, 120	
\l__talk_footelem_color_tl	281

\l__talk_footelem_font_tl	281	__talk_frame_title_tagged:n	15 , 72 , 76
\l__talk_footelem_left_skip	281	\g__talk_frame_title_tl	3 , 8 , 83 , 96 , 359 , 602
\l__talk_footelem_right_skip	281	\l__talk_frame_verb_bool	50 , 503 , 565 , 592 , 601 , 614 , 623 , 641
\l__talk_footer_bg_tl	380	\l__talk_frametitle_after_skip	25 , 53 , 63
\l__talk_footer_fg_tl	380	\l__talk_frametitle_before_skip	26 , 43 , 58
\l__talk_footer_font_tl	380	\l__talk_frametitle_color_tl	27 , 45 , 48
\l__talk_footer_left_skip	380	\l__talk_frametitle_font_tl	28 , 34
\l__talk_footer_order_clist	380	__talk_head_marks:nnnnn	186 , 199 , 199 , 217
\l__talk_footer_right_skip	380	__talk_head_marks_aux:Nnn	199
\l__talk_footer_sep_tl	380	__talk_head_marks_aux:Nnnn	206 , 213 , 218
\g__talk_footnote_box	98 , 111 , 141 , 144 , 171 , 183 , 185	__talk_head_section:Nnn	139
\g__talk_footnote_overlay_seq	172 , 176 , 186	__talk_head_subsection:Nnn	139
\l__talk_frame_alignment_tl	94 , 113 , 139 , 231 , 252	__talk_head_subsubsection:Nnn	139
\l__talk_frame_bool	141 , 153 , 500 , 538	__talk_head_tag:nn	235 , 238 , 241
\l__talk_frame_break_bool	57 , 89 , 232	\l__talk_header_bg_tl	317
\l__talk_frame_break_fp	121 , 128 , 232	\l__talk_header_fg_tl	317
\g__talk_frame_int	17 , 136 , 152 , 282 , 504 , 513 , 516 , 535 , 664 , 670	\l__talk_header_font_tl	317
\l__talk_frame_name_str	232 , 539 , 540 , 590 , 599 , 612 , 621 , 648	\l__talk_header_frametitle_bool	317
__talk_frame_overprint:	280 , 280 , 292 , 295 , 299 , 312 , 314 , 315 , 318 , 320 , 327 , 329 , 334	\l__talk_header_ht_dim	317
__talk_frame_process:nn	528 , 528 , 593 , 603 , 615 , 624 , 642	\l__talk_header_left_skip	317
__talk_frame_process_aux:nn	528 , 531 , 533	\l__talk_header_right_skip	317
__talk_frame_reuse:nn	628 , 628 , 649 , 660	__talk_header_tag_begin:n	78 , 266 , 266 , 273
__talk_frame_reuse_aux:nn	628 , 632 , 637	__talk_header_tag_end:	89 , 266 , 274
__talk_frame_reuse_aux:nnn	628 , 638 , 639	__talk_if_mode:n	3 , 10
__talk_frame_save:nn	528 , 540 , 548 , 557 , 572	__talk_if_mode:nTF	3 , 12 , 155 , 530
__talk_frame_set:n	573 , 573 , 591 , 600 , 613 , 622 , 647	__talk_if_overlay:n	3 , 10
__talk_frame_set:w	573 , 576 , 579	__talk_if_overlay:nTF	3 , 7 , 12 , 13 , 13 , 23 , 34 , 38 , 45 , 102 , 134 , 219 , 232 , 242 , 372 , 391 , 406 , 430 , 535
\g__talk_frame_struct_int	81 , 184 , 200	__talk_item_parse_spec:n	405 , 418 , 422 , 423
\g__talk_frame_subtitle_tl	3 , 13 , 97 , 360 , 363	__talk_item_parse_spec:w	405 , 415 , 421
\l__talk_frame_suppl_bool	232 , 541	__talk_label:n	403 , 407 , 410
\g__talk_frame_suppl_int	501 , 547 , 549 , 655 , 659	__talk_latex_frame:n	29 , 527 , 527
\g__talk_frame_tag_bool	46 , 68 , 71 , 72 , 83 , 103 , 104 , 257 , 502	\l__talk_list_end_tl	429 , 435 , 441
\l__talk_frame_tagging_str	20 , 21 , 23 , 25 , 102 , 232	__talk_metadata_name:n	421 , 424 , 429 , 445 , 445
__talk_frame_title:n	15 , 36 , 69	\l__talk_mode_str	69 , 82 , 113
\l__talk_frame_title_bool	69 , 145 , 585	\c__talk_modes_clist	70 , 78
		__talk_onslide:n	223 , 225 , 228 , 257
		\g__talk_onslide_tl	54 , 94 , 185 , 202 , 230 , 231 , 235 , 239
		__talk_opacity_begin:n	38 , 38 , 51 , 52 , 59 , 60 , 98 , 103 , 234

_talk_opacity_end:	_talk_slide_auxiv:n
..... 38, 40, 55, 63, 107, 211, 236	12, 51, 82
_talk_opacity_group_begin: ...	\l_talk_slide_box
..... 434, 434, 450	 5, 7, 44, 45, 49, 59, 70, 76,
_talk_opacity_group_end:	110, 114, 119, 120, 127, 129, 130, 140
..... 434, 436, 451	_talk_slide_break: 62, 117, 117
\g_talk_opacity_group_int	\g_talk_slide_continue_bool .. 3,
..... 433, 440, 442, 446	30, 37, 78, 101, 105, 147, 254, 264, 270
\l_talk_overlay_all_bool	\l_talk_slide_continue_bool ...
..... 143, 170, 172, 200	 3, 100, 106
_talk_overlay_arg:n	_talk_slide_fill: 64, 108, 108
..... 3, 11, 111, 118, 122, 129, 137	_talk_slide_init: 43, 92, 92
_talk_overprint_begin:n	\g_talk_slide_int
..... 261, 261, 269, 286	11, 28, 36, 250, 253, 260, 263, 268
_talk_overprint_check_ht:n ...	_talk_slide_output:
..... 285, 334, 336, 345	 21, 80, 131, 134, 134
\l_talk_overprint_int . 279, 283, 289	_talk_slide_tag_off_begin: ...
_talk_overprint_save_ht:	 48, 190, 190
..... 285, 290, 310	_talk_slide_tag_off_end:
_talk_pagecolor:n 43, 48, 49, 52	 74, 190, 195
\c_talk_paper_height_dim	_talk_slide_tag_on_begin:
122	 47, 181, 181
\c_talk_paper_width_dim	_talk_slide_tag_on_end: 73, 181, 186
122	\g_talk_subsection_tl
\g_talk_pauses_int	91
..... 10, 5, 51, 95, 220, 255, 256, 258	\l_talk_subsection_tl
\l_talk_saved_action_str	91, 183
..... 140, 181, 207	\g_talk_subsubsection_tl
\l_talk_saved_actions_bool	91
..... 140, 183, 209	\l_talk_subsubsection_tl .. 91, 185
_talk_saved_contentline:nmmn .	\l_talk_suppl_mode_str 69, 543
..... 227, 230, 233, 234	_talk_textcmd_equiv:n . 353, 374, 378
_talk_saved_ifendpe: . 64, 70, 88, 92	\l_talk_titlelem_after_skip 44
\l_talk_saved_overlays_bool ...	\l_talk_titlelem_before_skip ... 44
..... 140, 179, 204	\l_talk_titlelem_color_tl
\l_talk_sec_bookmark_tl 97, 176, 188	44
\l_talk_sec_label_tl	\l_talk_titlelem_font_tl
97	44
\l_talk_sec_nameref_tl 97, 189	\l_talk_titlelem_tag_begin_tl .. 44
\l_talk_sec_numbered_bool . 97, 179	\l_talk_titlelem_tag_end_tl 44
\l_talk_sec_quote_tl	\l_talk_titlepage_alignment_tl . 94
97	\l_talk_titlepage_framestyle_tl 94
\l_talk_sec_running_tl 97, 177	\l_talk_titlepage_order_clist .. 94
\l_talk_sec_subtitle_tl	_talk_tmp:w
97	65, 65, 125, 134
\l_talk_sec_toc_tl	\l_talk_tmp_box 18, 26, 32, 40, 50,
97, 178, 187	55, 58, 59, 60, 60, 62, 63, 63, 66, 66,
\g_talk_section_tl	66, 68, 71, 71, 72, 75, 85, 99, 264,
91	274, 296, 301, 305, 307, 321, 345,
\l_talk_section_tl	367, 370, 388, 401, 408, 435, 435, 444
91	\l_talk_tmp_clist
\l_talk_shuffle_skip .. 17, 20, 22, 34	 67, 149, 150, 151, 153
_talk_shuffle_skip:n . 18, 18, 39, 41	\l_talk_tmp_tl
_talk_slide:nn 12, 12, 552, 555	 15, 21, 24, 26, 68, 112,
_talk_slide_align_bottom:n 157, 157	187, 188, 418, 420, 421, 445, 454, 467
_talk_slide_align_center:n 157, 163	_talk_toc_aux:nmmn
_talk_slide_align_stretch:n ..	 284, 285, 288, 298, 307
..... 157, 169	_talk_toc_dest:n 284, 311, 314
_talk_slide_align_top:n . 157, 175	_talk_toc_dest:w 284, 316, 319
_talk_slide_auxi:nn 12, 31, 34	_talk_toc_level:nmmn . 284, 312, 339
_talk_slide_auxii:n 12, 39, 41	\l_talk_uncover_hidden_fp
_talk_slide_auxiii: . 12, 65, 68, 123	93

<code>\l__talk_vcenter_offset_tl</code>	64, 69, 75	<code>\@shortinstitute</code>	3
<code>__talk_wallpaper_hruler:Nnn</code>	340, 403, 451, 451	<code>\@shortsubtitle</code>	3
<code>talk/sec/title</code>	235	<code>\@shorttitle</code>	3
<code>\temporal</code>	10, 240	<code>\@skiphyperreftrue</code>	39, 131
T _E X and L ^A T _E X 2 _ε commands:		<code>\@starttoc</code>	253, 281
<code>\@appendixframenumber</code>	508	<code>\@starttoc@aux@nn</code>	258, 272
<code>\@appendixframes</code>	518	<code>\@subtitle</code>	3, 42
<code>\@arabic</code>	7, 10, 136, 137, 138, 185, 506, 510, 510, 518	<code>\@title</code>	3, 30, 31
<code>\@author</code>	3, 18, 19	<code>\@totalframes</code>	515
<code>\@auxout</code>	325, 414	<code>\c@appendixframe</code>	508
<code>\@biblabel</code>	510, 512	<code>\c@enumiv</code>	510, 518
<code>\@bsphack</code>	405	<code>\c@figure</code>	150
<code>\@caption</code>	160	<code>\c@frame</code>	504
<code>\@captertype</code>	130	<code>\c@page</code>	185
<code>\@clubpenalty</code>	522	<code>\c@pauses</code>	5
<code>\@contentsline@destination</code>	65, 316, 347, 350, 353, 356	<code>\c@section</code>	136
<code>\@currentHref</code>	421	<code>\c@slide</code>	8
<code>\@currentlabel</code>	418	<code>\c@subsection</code>	137
<code>\@currentlabelname</code>	201, 420	<code>\c@subsubsection</code>	138
<code>\@date</code>	3, 25	<code>\c@table</code>	150
<code>\@definecounter</code>	224	<code>\check@mathfonts</code>	58
<code>\@empty</code>	517, 531	<code>\currentgrouplevel</code>	68
<code>\@esphack</code>	408	<code>\fnum@figure</code>	150
<code>\@evenfoot</code>	471, 486, 497	<code>\fnum@table</code>	150
<code>\@evenhead</code>	470, 485, 496	<code>\Gm@bmargin</code>	406
<code>\@framenumber</code>	504	<code>\Gm@lmargin</code>	324, 387, 453
<code>\@frames</code>	512	<code>\Gm@rmargin</code>	326, 371, 388, 436
<code>\@ifpackageloaded</code>	160	<code>\Gm@tmargin</code>	323
<code>\@ignore</code>	38	<code>\if@endpe</code>	70, 88, 92
<code>\@ignoretrue</code>	101	<code>\if@minipage</code>	165
<code>\@input</code>	260	<code>\ifmeasuring@</code>	12
<code>\@institute</code>	3, 37	<code>\ignorespaces</code>	38
<code>\@kernel@reserved@label@data</code>	422	<code>\l@section</code>	284
<code>\@latex@warning</code>	528	<code>\l@subsection</code>	284
<code>\@listI</code>	56	<code>\l@subsubsection</code>	284
<code>\@listi</code>	49, 56	<code>\label@in@display</code>	54, 427, 428
<code>\@listii</code>	57	<code>\p@enumiv</code>	517
<code>\@listiii</code>	64	<code>\protected@write</code>	414
<code>\@m</code>	524	<code>\ps@plain</code>	463
<code>\@makecaption</code>	167	<code>\ps@talk</code>	463
<code>\@makefntext</code>	195	<code>\ps@wallpaper</code>	463
<code>\@mpfootins</code>	30	<code>\reset@color</code>	62, 63
<code>\@nbreakfalse</code>	269	<code>\set@color</code>	61, 63
<code>\@noitemerr</code>	527	<code>\std@definecounter</code>	224
<code>\@oddfoot</code>	469, 471, 480, 486, 495, 497	<code>\std@label@in@display</code>	427
<code>\@oddhead</code>	465, 470, 475, 485, 490, 496	<code>\stdreset@color</code>	61
<code>\@openbib@code</code>	515, 531	<code>\stdset@color</code>	61
<code>\@parboxrestore</code>	88, 164	<code>\textsubscript@offset</code>	181
<code>\@setminipage</code>	165	<code>\textsubscript@space</code>	181
<code>\@shortauthor</code>	3	<code>\textsuperscript@offset</code>	181
<code>\@shortdate</code>	3	<code>\textsuperscript@space</code>	181
		tex commands:	
		<code>\tex_currentgrouplevel:D</code>	431, 432
		<code>\tex_hsize:D</code>	37, 48

