

ltx-talk – A class for typesetting presentations*

Joseph Wright[†]

Released 2026-07-02

Contents

I	ltx-talk – Overall set up	1
1	ltx-talk implementation	1
1.1	Set up	1
1.2	Additions for expl3	2
1.3	Extra variants	2
1.4	Scratch space	2
1.5	Option handling	3
1.6	Setting up	4
1.7	Math support	5
1.8	Font selection	5
1.9	Text scripts	5
1.10	Hyperlinks	6
1.11	Tagging	6
II	ltx-talk-color – Color definitions	7
1	ltx-talk-color implementation	7
1.1	Existing definitions	7
1.2	Document (and interface) commands	7
1.3	Color definition	9
1.4	Semantic colors	9
III	ltx-talk-decode – Decoding overlay specs	10
1	ltx-talk-decode implementation	10
IV	ltx-talk-frame – The structure of frames	17

*This file describes v0.5.1, last revised 2026-07-02.

[†]E-mail: joseph@texdev.net

1	ltx-talk-frame implementation	17
1.1	Slides in frames	17
1.2	Counters	20
1.3	Frame options	21
1.4	Tagging for headers	22
1.5	Wallpaper	22
1.6	The <code>frame</code> environment	27
V	ltx-talk-frame – The structure of frames	31
1	ltx-talk-frame-structure implementation	31
1.1	Columns	31
1.2	Floats	33
1.3	Footnotes	35
VI	ltx-talk-mode – Modes	37
1	ltx-talk-mode implementation	37
VII	ltx-talk-overlay – Overlays	38
1	ltx-talk-overlay implementation	38
1.1	Utilities	38
1.2	Opacity utilities	39
1.3	Action commands and environments	39
1.4	Non-action commands and environments	44
1.5	Fixed-size areas	45
1.6	Adding overlays to existing commands	47
1.7	Overlay patching of third-party environments	49
VIII	ltx-talk-required – “Required” definitions	51
1	ltx-talk-required implementation	51
1.1	Standard design settings	51
1.2	List support	52
IX	ltx-talk-structure – Structural commands	53
1	ltx-talk-structure implementation	53
1.1	Frame title	53
1.2	Headings	54
1.3	Table of contents	58
1.4	Block environments	60
1.5	Lists	61
1.6	Theorems, <i>etc.</i>	65

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

Part I

ltx-talk – Overall set up

1 ltx-talk implementation

Start the DocStrip guards.

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

1.1 Set up

Identify the package and give the over all version information.

```
3 \ProvidesExplClass {ltx-talk} {2026-07-02} {0.5.1}
4   {A class for typesetting presentations}
    Get the right type of message.
5 \prop_gput:Nnn \g_msg_module_name_prop { talk } { ltx-talk }
6 \prop_gput:Nnn \g_msg_module_type_prop { talk } { Class }
    Require the latest LATEX structures.
7 \IfFormatAtLeastF { 2025-11-01 }
8   {
9     \msg_new:nnnn { ltx-talk } { kernel-too-old }
10    { The~ltx-talk~class~requires~LaTeX~2025-11-01~or~later. }
11    {
12      You~have~tried~to~use~the~ltx-talk~class~with~a~LaTeX~kernel~release~
13      prior~to~2025-11-01;~the~required~functionality~is~missing.
14    }
15    \msg_fatal:nn { ltx-talk } { kernel-too-old }
16  }
17 \NeedsDocumentMetadata
    Warn if not an engine that is tested.
18 \bool_lazy_or:nnF
19   { \sys_if_engine luatex_p: }
20   { \sys_if_engine pdftex_p: }
21   {
22     \msg_new:nnn { ltx-talk } { unsupported-engine }
23     {
24       The~engine~"\c_sys_engine_str"~
25       is~not~supported~by~the~ltx-talk~class.
26     }
27     \msg_warning:nn { ltx-talk } { unsupported-engine }
28   }
```

1.2 Additions for expl3

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

```

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

Some things from `xbox` that would be useful.

```

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

1.3 Extra variants

```

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

1.4 Scratch space

`__talk_tmp:w` For one-off processing.

```

61 \cs_new_protected:Npn \__talk_tmp:w { }
```

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

`\l__talk_tmp_box`

```

62 \box_new:N \l__talk_tmp_box
```

(End of definition for `\l__talk_tmp_box`.)

`\l__talk_tmp_tl`

```

63 \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
64 \keys_define:nn { talk }
65 {
66   aspect-ratio .str_set:N =
67     \l__talk_aspect_ratio_str ,
68   font-size .dim_set:N =
69     \l__talk_fontsize_dim ,
70   frame-title-arg .bool_set:N =
71     \l__talk_frame_title_bool ,
72   handout .code:n =
73     { \str_set:Nn \l__talk_mode_str { handout } } ,
74   handout .value_forbidden:n = true ,
75   mode .choices:nn =
76     { handout , projector }
77     { \str_set:NV \l__talk_mode_str \l_keys_choice_tl }
78 }
79 \str_new:N \l__talk_mode_str

(End of definition for \l__talk_aspect_ratio_str and others.)
Scope for options.
80 \keys_define:nn { talk }
81 {
82   aspect-ratio .usage:n = load ,
83   font-size .usage:n = load ,
84   frame-title-arg .usage:n = load ,
85   mode .usage:n = load
86 }

Compatibility keys for classical font size setting.
87 \clist_map_inline:nn { 10pt , 11pt , 12pt }
88 {
89   \keys_define:nn { talk }
90   {
91     #1 .meta:n = { font-size = #1 } ,
92     #1 .value_forbidden:n = true ,
93     #1 .usage:n = load
94   }
95 }

Initial values.
96 \keys_set:nn { talk }
97 {
98   aspect-ratio = 16:9 ,
99   font-size = 11pt ,
100   frame-title-arg = false ,
101   mode = projector
102 }
103 \ProcessKeyOptions [ talk ]

```

1.6 Setting up

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

```

104 \file_if_exist_input:nF { size \dim_to_decimal:n \l__talk_fontsize_dim .clo }
105 {
106   \file_input:n { size10.clo }
107   \RequirePackage { relsize }
108   \hook_gput_code:nne { begindocument } { talk }
109   { \exp_not:N \relsize { \fp_eval:n { \l__talk_fontsize_dim / 10pt } } } }
110 }

```

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`.

```

111 \dim_const:Nn \c__talk_paper_height_dim { 100mm }
112 \use:e
113 {
114   \cs_set_protected:Npn \exp_not:N \__talk_tmp:w
115     #1 \tl_to_str:n { : } #2 \tl_to_str:n { : } #3 \exp_not:N \q_stop
116   {
117     \dim_const:Nn \exp_not:N \c__talk_paper_width_dim
118     {
119       \exp_not:N \fp_to_dim:n
120       { (#1 / #2) * \exp_not:N \c__talk_paper_height_dim }
121     }
122   }
123   \exp_not:N \__talk_tmp:w \l__talk_aspect_ratio_str
124   \tl_to_str:n { : } 100 \exp_not:N \q_stop
125 }
126 \use:e
127 {
128   \exp_not:N \RequirePackage
129   [
130     papersize =
131     {
132       \dim_use:N \c__talk_paper_width_dim ,
133       \dim_use:N \c__talk_paper_height_dim
134     } ,
135     tmargin    = 10mm ,
136     bmargin    = 8mm ,
137     lmargin    = 10mm ,
138     rmargin    = 10mm ,
139     headheight = 10mm ,
140     headsep    = 2mm ,
141     footskip   = 6mm
142   ]
143   { geometry }
144 }

```

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

Turn off justification

```

145 \raggedright

```

1.7 Math support

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

```
146 \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.

```
147 \sys_if_engine_opentype:TF
148 {
149   \RequirePackage { fontspec }
150   \RequirePackage { mathtools }
151   \sys_if_engine luatex:TF
152   {
153     \RequirePackage { lua-unicode-math }
154     \tagpdfsetup { math / mathml / luamml / load = true }
155   }
156   { \RequirePackage { unicode-math } }
157   \setmainfont { NewCMSans10-Regular.otf }
158   \setsansfont { NewCMSans10-Regular.otf }
159   \setmathfont { NewCMSansMath-Regular.otf }
160 }
161 {
162   \RequirePackage { sansmathfonts }
163   \RequirePackage [ nomath ] { lmodern }
164   \cs_set_eq:NN \rmdefault \sfdefault
165 }
```

1.9 Text scripts

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

<code>\textsubscript@offset</code>	Offset values as in ConTeXt, no additional spacing added after script items.
<code>\textsubscript@space</code>	
<code>\textsuperscript@offset</code>	166 \cs_set_nopar:Npn \textsubscript@offset { 0.48ex }
<code>\textsuperscript@space</code>	167 \cs_set_nopar:Npn \textsubscript@space { }
	168 \cs_set_nopar:Npn \textsuperscript@offset { 0.86ex }
	169 \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.

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

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

A requirement.

```
171 \RequirePackage { hyperref }
```

```
172 \hypersetup { hidelinks }
```

1.11 Tagging

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

```
173 \tagpdfsetup
```

```
174 {
```

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

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

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

```
178 }
```

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

```
8 \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`.

```
9 \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.

```
10 \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`

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

13 \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 14 \bool_new:N \l__talk_decode_actions_bool

\l__talk_decode_actions_str 15 \clist_new:N \l__talk_decode_actions_clist

16 \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.)

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:n

__talk_decode_parse_auxi:n

__talk_decode_parse_auxii:n

__talk_decode_parse:w

17 \cs_new_protected:Npn __talk_decode_parse:n #1

18 { \exp_args:Ne __talk_decode_parse_auxi:n {#1} }

19 \cs_new_protected:Npn __talk_decode_parse_auxi:n #1

20 {

21 \str_clear:N \l__talk_decode_action_str

22 \bool_lazy_or:nnTF

23 { \tl_if_blank_p:n {#1} }

24 { \str_if_eq_p:nn {#1} { all } }

25 { \bool_set_true:N \l__talk_decode_overlays_bool }

26 {

27 \str_set:Nn \l__talk_decode_arg_str {#1}

28 \bool_set_false:N \l__talk_decode_actions_bool

29 \bool_set_false:N \l__talk_decode_overlays_bool

30 \bool_set_false:N \l__talk_decode_trailing_bool

31 \bool_set_false:N \l__talk_decode_modes_bool

32 \exp_args:No __talk_decode_parse_auxii:n { \l__talk_decode_arg_str }

33 }

34 }

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.

35 \cs_new_protected:Npn __talk_decode_parse_auxii:n #1

36 {

37 \bool_set_false:N \l__talk_decode_step_bool

38 __talk_decode_parse:w #1 | \q_recursion_tail | \q_recursion_stop

39 \bool_if:NT \l__talk_decode_step_bool

40 {

41 \legacy_if:nF { measuring@ }

42 { \int_gincr:N \g__talk_pauses_int }

43 }

44 }

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

45 \cs_new_protected:Npn __talk_decode_parse:w #1 |

46 {

```

47 \quark_if_recursion_tail_stop_do:nn {#1}
48 {
49   \bool_lazy_and:nnT
50   { \str_if_empty_p:N \l__talk_decode_overlays_str }
51   { ! \l__talk_decode_modes_bool }
52   { \bool_set_true:N \l__talk_decode_overlays_bool }
53 }
54 \exp_args:Ne \__talk_decode_mode:n
55 { \tl_trim_spaces:n {#1} }
56 \__talk_decode_parse:w
57 }

```

(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.

```

58 \clist_const:Ne \c__talk_modes_clist
59 {
60   \tl_to_str:n { article } ,
61   \tl_to_str:n { handout } ,
62   \tl_to_str:n { projector }
63 }

```

(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.

```

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

```

```

91     { \exp_not:N \__talk_decode_mode:nn {#1} {#2} }
92   }
93 \cs_new_protected:Npn \__talk_decode_mode:nn #1#2
94 {
95   \str_if_eq:VnTF \l__talk_mode_str {#1}
96   {
97     \__talk_decode_action:n {#2}
98     \str_if_empty:NT \l__talk_decode_overlays_str
99     { \__talk_decode_overlays:nn { overlays } { * } }
100   }
101   {
102     \tl_if_blank:nT {#2}
103     { \bool_set_true:N \l__talk_decode_modes_bool }
104   }
105 }

```

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

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

```

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

```



```

139     }
140     {
141         \msg_error:nnV { talk } { bad-action-spec }
142         \l__talk_decode_arg_str
143     }
144 }
145 }

```

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

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

```

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

```

169 \cs_new_protected:Npn \__talk_decode_overlay_aux:nNn #1#2#3
170 {
171   \quark_if_recursion_tail_stop_do:Nn #3
172   {
173     \__talk_decode_overlay_offset:nNn {#1} #2 { 0 }
174     \q_recursion_tail \q_recursion_stop
175   }
176   \token_if_eq_meaning:NNTF #3 ( % )
177   { \__talk_decode_overlay_offset:nNn {#1} #2 { } }
178   { \__talk_decode_overlay_offset:nNn {#1} #2 { 0 } #3 }
179 }

```

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.

```

180 \cs_new_protected:Npn \__talk_decode_overlay_offset:nNnN #1#2#3#4
181 {
182   \quark_if_recursion_tail_stop_do:Nn #4
183   {
184     \msg_error:nnV { talk } { bad-action-spec }
185     \l__talk_decode_arg_str
186   } % (
187   \token_if_eq_meaning:NNTF #4 )
188   { \__talk_decode_overlay_offset:nNn {#1} #2 {#3} }
189   { \__talk_decode_overlay_offset:nNnN {#1} #2 {#3#4} }
190 }

```

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

```

191 \cs_new_protected:Npn \__talk_decode_overlay_offset:nNn #1#2#3
192 {
193   \str_put_right:ce { l__talk_decode_ #1 _str }
194   { \int_max:nn { 0 } { #3 + \g__talk_pauses_int + #2 } }
195   \__talk_decode_overlays:nN {#1}
196 }

```

(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.

```

197 \cs_new_protected:Npn \__talk_decode_check:n #1
198 {
199   \clist_set:cv { l__talk_decode_ #1 _clist } { l__talk_decode_ #1 _str }
200   \clist_if_in:cnTF { l__talk_decode_ #1 _clist } { * }
201   { \bool_set_true:c { l__talk_decode_ #1 _bool } }
202   {
203     \clist_map_inline:cn { l__talk_decode_ #1 _clist }
204     { \__talk_decode_check:nw {#1} 0 ##1 - - \q_stop }
205   }
206 }

```

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.

```

207 \cs_new_protected:Npn \__talk_decode_check:nw #1#2 - #3 - #4 \q_stop
208 {
209   \tl_if_empty:nTF {#4}
210   { \__talk_decode_check_single:nn {#1} {#2} }
211   {
212     \tl_if_blank:nTF {#3}
213     { \__talk_decode_check_range:nnn {#1} {#2} { \c_max_int } }
214     { \__talk_decode_check_range:nnn {#1} {#2} {#3} }
215   }

```

```

216 }
217 \cs_new_protected:Npn \__talk_decode_check_single:nn #1#2
218 {
219   \int_compare:nNnTF \g__talk_slide_int = {#2}
220   { \bool_set_true:c { l__talk_decode_ #1 _bool } }
221   {
222     \int_compare:nNnTF {#2} > \g__talk_slide_int
223     { \bool_gset_true:N \g__talk_slide_continue_bool }
224     { \bool_set_true:N \l__talk_decode_trailing_bool }
225   }
226 }

```

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.

```

227 \cs_set_protected:Npn \__talk_decode_check_range:nnn #1#2#3
228 {
229   \int_compare:nNnTF \g__talk_slide_int > {#3}
230   { \bool_set_true:N \l__talk_decode_trailing_bool }
231   {
232     \int_compare:nNnTF \g__talk_slide_int < {#2}
233     { \bool_gset_true:N \g__talk_slide_continue_bool }
234     {
235       \bool_set_true:c { l__talk_decode_ #1 _bool }
236       \bool_lazy_and:nnT
237       { \int_compare_p:nNn \g__talk_slide_int < {#3} }
238       { \int_compare_p:nNn {#3} < \c_max_int }
239       { \bool_gset_true:N \g__talk_slide_continue_bool }
240       \clist_map_break:
241     }
242   }
243 }

```

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

```

244 \msg_new:nnnn { talk } { bad-action-spec }
245 { Bad-overlay-specification~"#1". }
246 {
247   The~overlay-specification-given~doesn't-follow~the~pattern-described-in~
248   the~ltx-talk-manual:~it~has~been~ignored.
249 }
250 \msg_new:nnnn { talk } { duplicate-action-spec }
251 { Duplicate~action-in~overlay-specification~"#1". }
252 {
253   The~overlay-specification~contains~more~than~one~action:
254   ltx-talk-only~supports~a~single~action~in~one~overlay.
255 }
256 \msg_new:nnnn { talk } { misplaced-action-spec }
257 { Misplaced~action-in~overlay-specification~"#1". }
258 {
259   The~overlay-specification~contains~an~action~before~the~overlay~
260   part(s):~this~is~not~supported.
261 }
262 </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`.)

```
\l__talk_slide_box
```

```
5 \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 L^AT_EX counter-style access.

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

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

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

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

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

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

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

```

11 {
12   \group_begin:
13     \tl_set:N\l__talk_tmp_tl
14     {
15       \property_ref:ee { frame . \int_use:N \g__talk_frame_int }
16       { slides }
17     }
18     \str_if_eq:VnTF \l__talk_frame_tagging_str { n }
19     { \str_set:N\l__talk_frame_tagging_str \l__talk_tmp_tl }
20     {
21       \str_replace_all:NnV \l__talk_frame_tagging_str { ,n }
22       \l__talk_tmp_tl
23       \str_replace_all:NnV \l__talk_frame_tagging_str { ,~n }
24       \l__talk_tmp_tl
25     }
26     \int_gzero:N \g__talk_slide_int
27     \RenewCommandCopy \frame \__talk_latex_frame:n
28     \bool_do_while:Nn \g__talk_slide_continue_bool
29     { \__talk_slide_aux:nn {#1} {#2} }
30     \property_record:ee { frame . \int_use:N \g__talk_frame_int }
31     { slides }
32   \group_end:
33 }

```

. At the start of each slide, we are in vertical mode. That is switched back . to horizontal mode before we deal with any tagging whatsits, then back to . vertical mode once this is sorted.

```

34 \cs_new_protected:Npn \__talk_slide_aux: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   {
40     \__talk_slide_begin:
41     \hbox:n
42     {
43       \tl_gclear:N \g__talk_onslide_tl
44       \bool_set_eq:NN \l__talk_slide_continue_bool
45       \g__talk_slide_continue_bool
46       \__talk_if_overlay:VTF \l__talk_frame_tagging_str
47       {
48         \bool_gset_eq:NN \g__talk_slide_continue_bool
49         \l__talk_slide_continue_bool
50         \__talk_frame_tag:n
51       }
52       {
53         \bool_gset_eq:NN \g__talk_slide_continue_bool
54         \l__talk_slide_continue_bool
55         \__talk_frame_notag:n
56       }
57     }
58     \vbox:n
59     {
60       \bool_if:NTF \l__talk_frame_verb_bool

```

```

61             { \__talk_slide_aux:n }
62             { \use:n }
63             {#2}
64         }
65     }
66     \tl_use:N \g__talk_onslide_tl
67 }
68 \__talk_slide_end:
69 }
70 }
71 \cs_new_protected:Npn \__talk_slide_aux:n #1
72 {
73     \group_begin:
74     \cs_set:Npn \obeyedline { ^^J }
75     \use:e
76     {
77         \group_end:
78         \tl_retokenize:n {#1}
79     }
80 }

```

(End of definition for __talk_slide:nn, __talk_slide__aux:nn, and __talk_slide_aux:n.)

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

```

81 \AddToHook { enddocument / afterlastpage }
82 {
83     \property_record:ee { frame . \int_use:N \g__talk_frame_int }
84     { slides }
85 }

```

\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).

```

86 \int_new:N \g__talk_frame_struct_int

```

(End of definition for \g__talk_frame_struct_int.)

__talk_slide_begin: Start and end the box holding all of the slide: we are in vertical mode here, and need to swap back to horizontal mode above to deal with whatsits.

```

87 \cs_new_protected:Npn \__talk_slide_begin:
88 {
89     \int_gzero:N \g__talk_pauses_int
90     \tl_gclear:N \g__talk_frame_title_tl
91     \tl_gclear:N \g__talk_frame_subtitle_tl
92     \box_gclear:N \g__talk_footnote_box
93     \__talk_cnt_save:
94     \vbox_set:Nw \l__talk_slide_box
95 }
96 \cs_new_protected:Npn \__talk_slide_end:
97 {
98     \vbox_set_end:
99     \bool_if:NT \g__talk_slide_continue_bool
100     { \__talk_cnt_restore: }
101     \vbox_to_ht:nn { \textheight }
102     {

```

```

103         \use:c { __talk_slide_align_ \l__talk_frame_alignment_tl :n }
104         { \vbox_unpack_drop:N \l__talk_slide_box }
105         \box_if_empty:NF \g__talk_footnote_box
106         {
107             \footnoterule
108             \vbox_unpack_drop:N \g__talk_footnote_box
109         }
110     }
111     \clearpage
112 }

```

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

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

```

\__talk_slide_align_center:n
    \__talk_slide_align_stretch:n
\__talk_slide_align_top:n
113 \cs_new_protected:Npn \__talk_slide_align_bottom:n #1
114 {
115     \skip_vertical:n { Opt~plus~1fil }
116     #1
117     \skip_vertical:n { Opt }
118 }
119 \cs_new_protected:Npn \__talk_slide_align_center:n #1
120 {
121     \skip_vertical:n { Opt~plus~0.5fil }
122     #1
123     \skip_vertical:n { Opt~plus~0.5fil }
124 }
125 \cs_new_protected:Npn \__talk_slide_align_stretch:n #1
126 {
127     \skip_vertical:n { Opt }
128     #1
129     \skip_vertical:n { Opt }
130 }
131 \cs_new_protected:Npn \__talk_slide_align_top:n #1
132 {
133     \skip_vertical:n { Opt }
134     #1
135     \skip_vertical:n { Opt~plus~1fil }
136 }

```

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

1.2 Counters

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

```

137 \seq_new:N \g__talk_cnt_reset_seq
138 \seq_gset_from_clist:Nn \g__talk_cnt_reset_seq
139 {
140     equation      ,
141     footnote      ,
142     mpfootnote    ,
143     parentequation

```

```

144 }
145 \seq_map_inline:Nn \g__talk_cnt_reset_seq
146 {
147   \int_new:c { g__talk_saved_ #1 _int }
148   \int_gset_eq:cc { g__talk_saved_ #1 _int } { c@ #1 }
149 }

```

(End of definition for \g__talk_cnt_reset_seq.)

```

\__talk_cnt_save: A simple save-and-restore pair.
\__talk_cnt_restore:
150 \cs_new_protected:Npn \__talk_cnt_save:
151 {
152   \seq_map_inline:Nn \g__talk_cnt_reset_seq
153   { \int_gset_eq:cc { g__talk_saved_ ##1 _int } { c@ ##1 } }
154 }
155 \cs_new_protected:Npn \__talk_cnt_restore:
156 {
157   \seq_map_inline:Nn \g__talk_cnt_reset_seq
158   { \int_gset_eq:cc { c@ ##1 } { g__talk_saved_ ##1 _int } }
159 }

```

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

```

\@definecounter Track all counters for resetting.
\std@definecounter
160 \cs_new_eq:NN \std@definecounter \@definecounter
161 \cs_gset_protected:Npn \@definecounter #1
162 {
163   \std@definecounter {#1}
164   \int_new:c { g__talk_saved_ #1 _int }
165   \seq_gput_right:Nn \g__talk_cnt_reset_seq {#1}
166 }

```

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

1.3 Frame options

```

\l__talk_frame_alignment_tl

```

```

167 \tl_new:N \l__talk_frame_alignment_tl

```

(End of definition for \l__talk_frame_alignment_tl.)

```

\l__talk_action_spec_str
\l__talk_frame_name_str
\l__talk_frame_tagging_str
168 \keys_define:nn { talk / frame }
169 {
170   action-spec .str_set:N =
171     \l__talk_action_spec_str ,
172   label .meta:n =
173     { name = {#1} } ,
174   name .str_set:N =
175     \l__talk_frame_name_str ,
176   tag-slides .str_set:N =
177     \l__talk_frame_tagging_str ,
178   vertical-alignment .choices:nn =
179     { bottom , center , stretch , top }

```



```

180     {
181         \tl_set_eq:NN \l__talk_frame_alignment_tl
182         \l_keys_value_tl
183     }
184 }
185 \keys_set:nn { talk / frame }
186 {
187     action-spec      =      ,
188     name             =      ,
189     tag-slides       = n     ,
190     vertical-alignment = center
191 }

```

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

1.4 Tagging for headers

`__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:`

```

192 \cs_new_protected:Npn \__talk_header_tag_begin:n #1
193 {
194     \tag_resume:n { header }
195     \tag_mc_end:
196     \tag_struct_begin:n {#1}
197     \tag_mc_begin:n { }
198 }
199 \cs_generate_variant:Nn \__talk_header_tag_begin:n { e }
200 \cs_new_protected:Npn \__talk_header_tag_end:
201 {
202     \tag_mc_end:
203     \tag_struct_end:
204     \tag_mc_begin:n { artifact }
205     \tag_suspend:n { header }
206 }

```

(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
207 \NewTemplateType { footer-element } { 1 }
208 \DeclareTemplateInterface { footer-element } { talk } { 1 }
209 {
210     color      : tokenlist ,
211     font       : tokenlist = ,
212     left-hspace : length = 0em ,
213     right-hspace : length = 0em
214 }
215 \DeclareTemplateCode { footer-element } { talk } { 1 }
216 {
217     color      = \l__talk_footelem_color_tl ,
218     font       = \l__talk_footelem_font_tl ,
219     left-hspace = \l__talk_footelem_left_skip ,

```

```

220 right-hspace = \l__talk_footelem_right_skip
221 }
222 {
223   \tl_if_empty:nF {#1}
224   {
225     \hspace { \l__talk_footelem_left_skip }
226     \hbox:n
227     {
228       \tl_if_empty:NF \l__talk_footelem_color_tl
229       { \color_select:V \l__talk_footelem_color_tl }
230       \l__talk_footelem_font_tl
231       #1
232     }
233     \hspace { \l__talk_footelem_right_skip }
234   }
235 }
236 \DeclareInstance { footer-element } { date } { talk } { }
237 \DeclareInstance { footer-element } { author } { talk } { }
238 \DeclareInstance { footer-element } { title } { talk } { }
239 \DeclareInstance { footer-element } { subtitle } { talk } { }
240 \DeclareInstance { footer-element } { institute } { talk } { }
241 \DeclareInstance { footer-element } { framenumbers } { talk } { }
242 \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>	<p>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.</p> <pre> 243 \NewTemplateType { header } { 0 } 244 \DeclareTemplateInterface { header } { talk } { 0 } 245 { 246 background-color : tokenlist, 247 color : tokenlist = structure , 248 font : tokenlist = \normalfont , 249 height : length = \Gm@tmargin + \headsep , 250 left-hspace : skip = \Gm@lmargin , 251 print-frame-title : boolean = true , 252 right-hspace : skip = \Gm@rmargin 253 } 254 \DeclareTemplateCode { header } { talk } { 0 } 255 { 256 background-color = \l__talk_header_bg_tl , 257 color = \l__talk_header_fg_tl , 258 font = \l__talk_header_font_tl , 259 height = \l__talk_header_ht_dim , 260 left-hspace = \l__talk_header_left_skip , 261 print-frame-title = \l__talk_header_frametitle_bool , 262 right-hspace = \l__talk_header_right_skip 263 } 264 { 265 \noindent 266 __talk_wallpaper_hrulerule:Nnn </pre>
--	---

```

267 \l__talk_header_bg_tl
268 { \l__talk_header_ht_dim - \headsep }
269 { \headsep }
270 \skip_horizontal:n { \l__talk_header_left_skip }
271 \group_begin:
272 \tl_if_empty:NF \l__talk_header_fg_tl
273 { \color_select:V \l__talk_header_fg_tl }
274 \l__talk_header_font_tl
275 \bool_if:NT \l__talk_header_frametitle_bool
276 {
277 \ExpandArgs { nnV }
278 \UseInstance { frametitle } { header }
279 \g__talk_frame_title_tl
280 }
281 \group_end:
282 }
283 \DeclareInstance { header } { std } { talk } { }
284 \AddToHook { begindocument }
285 {
286 \DeclareInstanceCopy { header } { wallpaper } { std }
287 \EditInstance { header } { wallpaper }
288 { print-frame-title = false }
289 }

```

(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
290 \NewTemplateType { footer } { 0 }
291 \DeclareTemplateInterface { footer } { talk } { 0 }
292 {
293 background-color : tokenlist ,
294 color : tokenlist ,
295 element-order : commalist ,
296 font : tokenlist = \tiny ,
297 left-hspace : length = \Gm@lmargin ,
298 right-hspace : length = \Gm@rmargin ,
299 separator : tokenlist = \hfil
300 }
301 \DeclareTemplateCode { footer } { talk } { 0 }
302 {
303 background-color = \l__talk_footer_bg_tl ,
304 color = \l__talk_footer_fg_tl ,
305 element-order = \l__talk_footer_order_clist ,
306 font = \l__talk_footer_font_tl ,
307 left-hspace = \l__talk_footer_left_skip ,
308 right-hspace = \l__talk_footer_right_skip ,
309 separator = \l__talk_footer_sep_tl
310 }
311 {
312 \noindent
313 \__talk_wallpaper_hruler:Nnn
314 \l__talk_footer_bg_tl

```

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.

```

315 { \footskip }
316 { \Gm@bmargin - \footskip }
317 \skip_horizontal:n { \l__talk_footer_left_skip }
318 \vbox_set_to_wd:Nnn \l__talk_tmp_box
319 {
320     \paperwidth
321     - \l__talk_footer_left_skip
322     - \l__talk_footer_right_skip
323 }
324 {
325     \tl_if_empty:NF \l__talk_footer_fg_tl
326     { \color_select:V \l__talk_footer_fg_tl }
327     \l__talk_footer_font_tl
328     \clist_pop:NNT \l__talk_footer_order_clist \l__talk_tmp_tl
329     {
330         \ExpandArgs { nVv } \UseInstance { footer-element } \l__talk_tmp_tl
331         { @ \__talk_metadata_name:n { \l__talk_tmp_tl } }
332         \clist_map_inline:Nn \l__talk_footer_order_clist
333         {
334             \tl_if_empty:cF { @ \__talk_metadata_name:n { ##1 } }
335             {
336                 \l__talk_footer_sep_tl
337                 \ExpandArgs { nnv }
338                 \UseInstance { footer-element } {##1}
339                 { @ \__talk_metadata_name:n { ##1 } }
340             }
341         }
342     }
343     \hfil
344 }
345 \box_use_drop:N \l__talk_tmp_box
346 \skip_horizontal:n { \l__talk_footer_right_skip - \Gm@rmargin }
347 }
348 \DeclareInstance { footer } { std } { talk } { }
349 \AddToHook { begindocument }
350 {
351     \DeclareInstanceCopy { footer } { wallpaper } { std }
352     \EditInstance { footer } { wallpaper }
353     { element-order = }
354 }

```

(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.

```

355 \cs_new:Npn \__talk_metadata_name:n #1
356 {
357     \tl_if_exist:cTF { @ short #1 }
358     { short #1 }
359     {#1}
360 }

```

(End of definition for __talk_metadata_name:n.)

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

```

361 \cs_new_protected:Npn \__talk_wallpaper_hrule:Nnn #1#2#3
362 {
363   \skip_horizontal:n { -\Gm@lmargin }
364   \tl_if_empty:NF #1
365   {
366     \group_begin:
367     \color_select:V #1
368     \rule:nnn { \paperwidth } {#2} {#3}
369     \skip_horizontal:n { -\paperwidth }
370     \group_end:
371   }
372 }

```

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

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

`\ps@wallpaper`
`\ps@talk`

```

373 \cs_set_nopar:Npn \ps@plain
374 {
375   \cs_set_nopar:Npn \@oddhead
376   {
377     \hfil
378   }
379   \cs_set_nopar:Npn \@oddfoot { }
380   \cs_set_eq:NN \@evenhead \@oddhead
381   \cs_set_eq:NN \@evenfoot \@oddfoot
382 }
383 \cs_set_nopar:Npn \ps@wallpaper
384 {
385   \cs_set_nopar:Npn \@oddhead
386   {
387     \UseInstance { header } { wallpaper }
388     \hfil
389   }
390   \cs_set_nopar:Npn \@oddfoot
391   {
392     \UseInstance { footer } { wallpaper }
393     \hfil
394   }
395   \cs_set_eq:NN \@evenhead \@oddhead
396   \cs_set_eq:NN \@evenfoot \@oddfoot
397 }
398 \cs_new_nopar:Npn \ps@talk
399 {
400   \cs_set_nopar:Npn \@oddhead
401   {
402     \UseInstance { header } { std }
403     \hfil
404   }
405   \cs_set_nopar:Npn \@oddfoot { \UseInstance { footer } { std } }
406   \cs_set_eq:NN \@evenhead \@oddhead

```

```

407 \cs_set_eq:NN \@evenfoot \@oddfoot
408 }
409 \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.
410 \bool_new:N \l__talk_frame_bool

(End of definition for \l__talk_frame_bool.)

\g__talk_frame_tag_bool To track when a frame is being tagged: mainly needed for the header (and as a result
global).
411 \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).
412 \bool_new:N \l__talk_frame_verb_bool

(End of definition for \l__talk_frame_verb_bool.)

\g__talk_frame_int The overall frame number, including LATEX counter-like access.
\c@frame 413 \int_new:N \g__talk_frame_int
\theframe 414 \cs_new_eq:NN \c@frame \g__talk_frame_int
\@framenum 415 \cs_new:Npn \theframe { \@arabic \c@frame }
416 \cs_new:Npn \@framenum { \arabic { frame } }

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

\@totalframes The total frames can be handled using the kernel properties.
417 \property_new:nnnn { totalframes } { shipout } { -1 }
418 { \int_use:N \g__talk_frame_int }
419 \AddToHook { enddocument / afterlastpage }
420 { \property_record:nn { lastpage } { totalframes } }
421 \cs_new:Npn \@totalframes { \property_ref:nn { lastpage } { totalframes } }

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

\__talk_latex_frame:n As we will need to re-define \frame but have it available inside frames, a copy is made
here.
422 \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.

```
423 \cs_new_protected:Npn \__talk_frame_process:nn #1#2
424 {
425   \int_gincr:N \g__talk_frame_int
426   \bool_set_true:N \l__talk_frame_bool
427   \str_if_empty:NF \l__talk_frame_name_str
428   {
429     \tl_if_exist:cT { g__talk_frame_ \l__talk_frame_name_str _tl }
430     {
431       \msg_error:nnV { talk } { duplicate-frame-name }
432       \l__talk_frame_name_str
433     }
434     \tl_clear_new:c { g__talk_frame_ \l__talk_frame_name_str _tl }
435     \tl_gset:ce { g__talk_frame_ \l__talk_frame_name_str _tl }
436     {
437       {
438         \bool_if:NTF \l__talk_frame_verb_bool
439         { true }
440         { false }
441       }
442       { \exp_not:n {#2} }
443     }
444   }
445   \__talk_slide:nn {#1} {#2}
446 }
```

(End of definition for __talk_frame_process:nn.)

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

```
447 \cs_new_protected:Npn \__talk_frame_tag:n #1
448 {
449   \tag_struct_begin:n { tag = frame }
450   \int_gset:Nn \g__talk_frame_struct_int { \tag_get:n { struct_num } }
451   \bool_gset_true:N \g__talk_frame_tag_bool
452   #1
453   \tag_struct_end:
454 }
```

(End of definition for __talk_frame_tag:n.)

__talk_frame_notag:n The alternative: turn off tagging and suppress the function that would tag the frame title.

```
455 \cs_new_protected:Npn \__talk_frame_notag:n #1
456 {
457   \tag_mc_begin:n { artifact }
458   \tag_suspend:n { frame }
459   \bool_gset_false:N \g__talk_frame_tag_bool
460   #1
461   \par
462   \tag_resume:n { frame }
463   \tag_mc_end:
464 }
```

(End of definition for __talk_frame_notag:n.)

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

```

465 \bool_if:NTF \l__talk_frame_title_bool
466 {
467   \RenewDocumentEnvironment { frame }
468     { D <> { all } = { action-spec } 0 { } +m +b }
469     {
470       \str_clear:N \l__talk_frame_name_str
471       \keys_set:nn { talk / frame } {#2}
472       \bool_set_false:N \l__talk_frame_verb_bool
473       \__talk_frame_process:nn {#1} { \frametitle {#3} #4 }
474     }
475   { }
476   \NewDocumentEnvironment { frame* }
477     { D <> { all } = { action-spec } 0 { } +m c }
478     {
479       \str_clear:N \l__talk_frame_name_str
480       \keys_set:nn { talk / frame } {#2}
481       \bool_set_true:N \l__talk_frame_verb_bool
482       \tl_gset:Nn \g__talk_frame_title_tl {#3}
483       \exp_args:Nne \__talk_frame_process:nn {#1}
484         { \tl_to_str:n { \frametitle } \exp_not:n { {#3} #4 } }
485     }
486   { }
487 }
488 {
489   \RenewDocumentEnvironment { frame }
490     { !D <> { all } = { action-spec } !0 { } +b }
491     {
492       \str_clear:N \l__talk_frame_name_str
493       \keys_set:nn { talk / frame } {#2}
494       \bool_set_false:N \l__talk_frame_verb_bool
495       \__talk_frame_process:nn {#1} {#3}
496     }
497   { }
498   \NewDocumentEnvironment { frame* }
499     { !D <> { all } = { action-spec } !0 { } c }
500     {
501       \str_clear:N \l__talk_frame_name_str
502       \keys_set:nn { talk / frame } {#2}
503       \bool_set_true:N \l__talk_frame_verb_bool
504       \__talk_frame_process:nn {#1} {#3}
505     }
506   { }
507 }

```

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

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

```

508 \cs_new_protected:Npn \__talk_frame_reuse:nn #1#2
509 {
510   \tl_if_exist:cTF { g__talk_frame_ #2 _tl }
511   {

```



```

512     \exp_args:Nv \__talk_frame_reuse_aux:nn
513     { g__talk_frame_ #2 _t1 } {#1}
514   }
515   { \msg_error:nnn { talk } { unknown-frame-name } {#2} }
516 }
517 \cs_new_protected:Npn \__talk_frame_reuse_aux:nn #1#2
518 { \__talk_frame_reuse_aux:nnn #1 {#2} }
519 \cs_new_protected:Npn \__talk_frame_reuse_aux:nnn #1#2#3
520 {
521   \use:c { bool_set_ #1 :N } \l__talk_frame_verb_bool
522   \__talk_frame_process:nn {#3} {#2}
523 }
524 \NewDocumentCommand \reuseframe { D <> { all } = { action-spec } 0 { } m }
525 {
526   \group_begin:
527     \keys_set:nn { talk / frame } {#2}
528     \str_clear:N \l__talk_frame_name_str
529     \__talk_frame_reuse:nn {#1} {#3}
530   \group_end:
531 }
532 \NewCommandCopy \againframe \reuseframe

```

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

```

533 \msg_new:nnnn { talk } { unknown-frame-name }
534 { Unknown~frame~name~"#1"! }
535 {
536   You~asked~to~reuse~the~contents~of~a~frame~with~the~name~"#1",~
537   but~that~name~was~never~defined.
538 }
539 \msg_new:nnnn { talk } { duplicate-frame-name }
540 { Duplicate~frame~name~"#1"! }
541 {
542   You~asked~to~save~the~contents~of~this~frame~with~the~name~"#1",~
543   but~that~name~has~already~been~used.
544 }
545 </class>

```

Part V

ltx-talk-frame – The structure of frames

1 ltx-talk-frame-structure implementation

Start the DocStrip guards.

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

1.1 Columns

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

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

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

(End of definition for `\l__talk_columns_wd_tl`.)

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

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

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

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

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

```

28   \par
29   \__talk_action_end:
30 }

```

\l__talk_column_alignment_tl

```

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

```

(End of definition for \l__talk_column_alignment_tl.)

__talk_column_align_bottom:n
 __talk_column_align_center:n
 __talk_column_align_top:n

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

```

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

```

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

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

```

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

```

(End of definition for `\l__talk_vcenter_offset_tl`.)

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

```

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

```

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

```

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

```

1.2 Floats

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

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

```

103 \tl_new:N \l__talk_float_alignment_tl

```

(End of definition for \l__talk_float_alignment_tl.)

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

```

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

```

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

```

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

```

`\endfloatenv` And the common end function.

```

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

```

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

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

```





```

```

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

```

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

```

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

```

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

The spacing values needed for the standard function.

```

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

```

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

```

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

```

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

1.3 Footnotes

`\g__talk_footnote_box` Holds footnotes as they are constructed.

```

171 \box_new:N \g__talk_footnote_box

```

(End of definition for \g__talk_footnote_box.)

`\g__talk_footnote_overlay_seq` For tracking the overlays to apply.

```

172 \seq_new:N \g__talk_footnote_overlay_seq

```

(End of definition for \g__talk_footnote_overlay_seq.)

\stdfootnote

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

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

\footnote Sort-of overlay aware!

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

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

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

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

\@makefntext Use a copy of the standard setup.

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

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

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

Part VI

ltx-talk-mode — Modes

1 ltx-talk-mode implementation

Start the DocStrip guards.

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

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

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

(End of definition for __talk_mode:nT.)

`\mode`

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

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

```
13 </class>
```


Part VII

ltx-talk-overlay — Overlays

1 ltx-talk-overlay implementation

Start the DocStrip guards.

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

1.1 Utilities

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

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

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

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

\l__talk_shuffle_skip For tracking.

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

(End of definition for \l__talk_shuffle_skip.)

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

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

```

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

```

(End of definition for __talk_shuffle_skip:n.)

1.2 Opacity utilities

Currently, opacity is applied using what's 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

```

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.

```

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         \vbox_set:Nw \l__talk_tmp_box
71         \hbox_set:Nw \l__talk_tmp_box
72         \tag_suspend:n { onlyenv }
73       }
74       \vbox_set:Nw \l__talk_tmp_box
75     }
76   }
77 \cs_new_protected:Npn \_talk_action_only_end:N #1
78   {
79     \bool_if:NF #1
80     {
81       \vbox_set_end:
82       \bool_if:NT \g__talk_frame_tag_bool
83       {
84         \tag_resume:n { onlyenv }
85         \hbox_set_end:
86         \vbox_set_end:
87       }
88     }
89   }

```

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

```
\l__talk_uncover_hidden_fp
```

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

```

90 \NewTemplateType { hidden } { 0 }
91 \DeclareTemplateInterface { hidden } { talk } { 0 }
92   { opacity : real = 0 }
93 \DeclareTemplateCode { hidden } { talk } { 0 }
94   { opacity = \l__talk_uncover_hidden_fp }
95   { \_talk_opacity_begin:n { \l__talk_uncover_hidden_fp } }
96 \DeclareInstance { hidden } { std } { talk } { }

```

(End of definition for \l__talk_uncover_hidden_fp.)

```

\_talk_action_uncover:N
\_talk_action_uncover_end:N

```

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

```

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

```

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

invisibleenv (env.) And the environment versions.

```

\uncoverenv (env.) 114 \ExpandArgs { nnee } \NewDocumentEnvironment { #1 env }
\visibleenv (env.) 115 { > { \__talk_overlay_arg:n } D <> { all } }
116 { \exp_not:c { __talk_action_ #1 :N } ##1 }
117 { \exp_not:c { __talk_action_ #1 _end:N } ##1 }
118 }

```

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

```

119 \NewDocumentCommand \alert { > { \__talk_overlay_arg:n } D <> { all } +m }
120 {
121   \group_begin:
122     \__talk_action_alert:N #1
123     #2
124   \group_end:
125 }

```

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

alertenv (env.) As does the environment.

```

126 \NewDocumentEnvironment { alertenv } { > { \__talk_overlay_arg:n } D <> { all } }
127 { \__talk_action_alert:N #1 }
128 { }

```

\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).

```

129 \NewDocumentCommand \only { D <> { all } +m }

```

```

130 {
131   \__talk_if_overlay:nT {#1}
132   {#2}
133 }

```

(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.

```

134 \NewDocumentEnvironment { onlyenv } { > { \__talk_overlay_arg:n } D <> { all } }
135 { \__talk_action_only:N #1 }
136 { \__talk_action_only_end:N #1 }

```

```

\l__talk_saved_overlays_bool
\l__talk_saved_action_str
\l__talk_saved_actions_bool
137 \bool_new:N \l__talk_saved_overlays_bool
138 \str_new:N \l__talk_saved_action_str
139 \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
140 \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:
141 \NewDocumentCommand \action { d <> +m }
142 {
143   \group_begin:
144     \__talk_action_begin:n {#1}
145     #2
146     \__talk_action_end:
147   \group_end:
148 }
149 \NewDocumentEnvironment { actionenv } { d <> }
150 { \__talk_action_begin:n {#1} }
151 { \__talk_action_end: }
152 \cs_new_protected:Npn \__talk_action_begin:n #1
153 {
154   \group_begin:
155     \tl_if_novalue:nTF {#1}
156     {
157       \exp_after:wN \__talk_action_begin:w
158       \l__talk_action_spec_str < all > \q_stop
159     }
160     { \__talk_action_begin_auxi:n {#1} }
161   }
162 \cs_new_protected:Npn \__talk_action_begin:w #1 < #2 > #3 \q_stop

```

```

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

```

(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.

```
214 \NewDocumentCommand \alt { D <> { all } +m +m }
215 {
216   \__talk_if_overlay:nTF {#1}
217   {#2}
218   {#3}
219 }
```

(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

```
220 \NewDocumentCommand \onslide { D <> { all } }
221 {
222   \__talk_onslide:n {#1}
223   \ignorespaces
224 }
225 \cs_new_protected:Npn \__talk_onslide:n #1
226 {
227   \tl_use:N \g__talk_onslide_tl
228   \tl_gclear:N \g__talk_onslide_tl
229   \__talk_if_overlay:nF {#1}
230   {
231     \__talk_opacity_begin:n { 0 }
232     \tl_gput_right:Nn \g__talk_onslide_tl
233     { \__talk_opacity_end: }
234   }
235 }
```

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

\g__talk_onslide_tl

```
236 \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.

```
237 \NewDocumentCommand \temporal { D <> { all } +m +m +m }
238 {
239   \__talk_if_overlay:nTF {#1}
240   {#3}
241   {
242     \bool_if:NTF \l__talk_decode_trailing_bool
243     {#4}
244     {#2}
245   }
246 }
```

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

`\pause` A thin wrapper.

```

247 \NewDocumentCommand \pause { o }
248 {
249   \legacy_if:nF { measuring@ }
250   {
251     \IfNoValueTF {#1}
252     { \int_gincr:N \g__talk_pauses_int }
253     { \int_gset:Nn \g__talk_pauses_int {#1} }
254     \exp_args:Ne \__talk_onslide:n
255     { \int_eval:n { \g__talk_pauses_int + 1 } - }
256   }
257 }

```

(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`.

```

258 \cs_new_protected:Npn \__talk_overprint_begin:n #1
259 {
260   \par
261   \vbox_set_to_wd:Nnw \l__talk_tmp_box {#1}
262   \raggedright
263   \ignorespaces
264 }

```

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

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

```

265 \NewDocumentEnvironment { overlayarea } { m m }
266 { \__talk_overprint_begin:n {#1} }
267 {
268   \vbox_set_end:
269   \vbox_to_ht:nn {#2}
270   {
271     \box_use_drop:N \l__talk_tmp_box
272     \vfil
273   }
274   \par
275 }

```

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

```

276 \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.

```

277 \cs_new:Npn \__talk_frame_overprint:
278 {
279   \int_to_Roman:n \g__talk_frame_int
280   \int_to_roman:n \l__talk_overprint_int
281 }

```


(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.

```

282 \NewDocumentEnvironment { overprint } { 0 { \textwidth } }
283 { \__talk_overprint_begin:n {#1} }
284 {
285   \vbox_set_end:
286   \int_incr:N \l__talk_overprint_int
287   \__talk_overprint_save_ht:
288   \cs_if_exist:cTF
289     { overprint@ \__talk_frame_overprint: }
290     {
291       \dim_compare:vNnTF
292         { overprint@ \__talk_frame_overprint: }
293         > { \box_ht:N \l__talk_tmp_box }
294         {
295           \vbox_to_ht:vn
296             { overprint@ \__talk_frame_overprint: }
297             {
298               \box_use_drop:N \l__talk_tmp_box
299               \vfil
300             }
301           }
302         { \box_use_drop:N \l__talk_tmp_box }
303       }
304     { \box_use_drop:N \l__talk_tmp_box }
305   \par
306 }

```

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.

```

307 \cs_new_protected:Npn \__talk_overprint_save_ht:
308 {
309   \tl_if_exist:cF { g__talk_overprint_ \__talk_frame_overprint: _tl }
310   {
311     \tl_new:c { g__talk_overprint_ \__talk_frame_overprint: _tl }
312     \tl_gset:cn { g__talk_overprint_ \__talk_frame_overprint: _tl }
313       { Opt }
314   }
315   \tl_gset:ce { g__talk_overprint_ \__talk_frame_overprint: _tl }
316   {
317     \dim_max:vn { g__talk_overprint_ \__talk_frame_overprint: _tl }
318     { \box_ht:N \l__talk_tmp_box }
319   }
320   \legacy_if:nT { @files }
321   {
322     \iow_now:Ne \@auxout
323     {
324       \gdef \exp_not:c { overprint@ \__talk_frame_overprint: }
325       {
326         \exp_not:v { g__talk_overprint_ \__talk_frame_overprint: _tl }
327       }

```

```

328     }
329   }
330   \hook_gput_code:nne { enddocument / afterlastpage } { talk }
331   { \__talk_overprint_check_ht:n { \__talk_frame_overprint: } }
332 }
333 \cs_new_protected:Npn \__talk_overprint_check_ht:n #1
334 {
335   \bool_lazy_and:nnF
336     { \exp_not:N \cs_if_exist_p:c { overprint@ #1 } }
337     {
338       \dim_compare_p:vNv { overprint@ #1 } = { g__talk_overprint_ #1 _tl }
339     }
340     {
341       \msg_warning:nn { talk } { overprint-ht }
342       \cs_gset_protected:Npn \__talk_overprint_check_ht:n ##1 { }
343     }
344 }
345 \msg_new:nnn { talk } { overprint-ht }
346 {
347   Overprint~area~height~has~changed:\\
348   rerun~LaTeX.
349 }

```

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

1.6 Adding overlays to existing commands

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

```

\textnormal 350 \tl_map_inline:nn
\textrm      351 {
\textsc      352   \textbf
\textsf      353   \textit
\textsl      354   \textmd
\texttt      355   \textnormal
\textup      356   \textrm
\emph        357   \textsc
\stdtextbf   358   \textsf
\stdtextit   359   \textsl
\stdtextmd   360   \texttt
\stdtextup   361   \textup
\stdtextnormal 362   \emph
\stdtextrm   363 }
\stdtextsc   364 {
\stdtextsf   365   \ExpandArgs { c } \NewCommandCopy { std \cs_to_str:N #1 } #1
\stdtextsl   366   \ExpandArgs { Nne } \RenewDocumentCommand #1
\stdtexttt   367     { D <> { all } +m }
\stdtextup   368     {
369       \exp_not:N \__talk_if_overlay:nTF {##1}
370       { \exp_not:c { std \cs_to_str:N #1 } }
371       { \exp_not:N \__talk_textcmd_equiv:n }
372       {##2}
373     }

```

`__talk_textcmd_equiv:n`

```

374 }
375 \cs_new_protected:Npn \__talk_textcmd_equiv:n #1
376 {
377   \mode_if_math:TF
378   { { \mbox {#1} } }
379   {
380     \mode_leave_vertical:
381     {#1}
382   }
383 }

```

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

```

(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
400 \RenewDocumentCommand \label { D <> { 1 } m }
401 {
402   \@bsphack
403   \__talk_if_overlay:nT {#1}
404   { \__talk_label:n {#2} }
405   \@esphack
406 }
407 \cs_new_protected:Npn \__talk_label:n #1
408 {
409   \begingroup
410   \UseHookWithArguments { label } { 1 } {#1}
411   \protected@write \@auxout { }
412   {
413     \string \newlabel {#1}
414     {
415       \@currentlabel }

```

```

416         { \thepage }
417         { \@currentlabelname }
418         { \@currentHref }
419         { \@kernel@reserved@label@data }
420     }
421 }
422 \endgroup
423 }

```

(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!

```

424 \cs_new_eq:NN \std@label@in@display \label@in@display
425 \RenewDocumentCommand \label@in@display { D <> { 1 } m }
426 {
427     \_talk_if_overlay:nT {#1}
428     { \std@label@in@display {#2} }
429 }

```

(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.

```

430 \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:

```

431 \cs_new_protected:Npn \_talk_opacity_group_begin:
432 { \begin { lrbox } \l__talk_tmp_box }
433 \cs_new_protected:Npn \_talk_opacity_group_end:
434 {
435     \end { lrbox }
436     \mode_leave_vertical:
437     \int_gincr:N \g__talk_opacity_group_int
438     \exp_args:Ne \pdfxform_new:nnn
439     { talk.opacity \int_use:N \g__talk_opacity_group_int }
440     { /Group << /S /Transparency /K ~ false /I ~ false >> }
441     { \usebox \l__talk_tmp_box }
442     \exp_args:Ne \pdfxform_use:n
443     { talk.opacity \int_use:N \g__talk_opacity_group_int }
444 }

```

(End of definition for `_talk_opacity_group_begin:` and `_talk_opacity_group_end:`.)

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

```
445 \sys_if_engine_luatex:T
446 {
447   \AddToHook { cmd / pspicture / before } { \_talk_opacity_group_begin: }
448   \AddToHook { cmd / endpspicture / after } { \_talk_opacity_group_end: }
449 }
450 </class>
```

Part VIII

ltx-talk-required – “Required” definitions

1 ltx-talk-required implementation

Start the DocStrip guards.

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

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

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

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

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

1.1 Standard design settings

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

```

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

```

1.2 List support

```

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

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

```

Part IX

ltx-talk-structure – Structural commands

1 ltx-talk-structure implementation

Start the DocStrip guards.

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

1.1 Frame title

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

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

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

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

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

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



```

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

```

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

1.2 Headings

Two versions of the data store: one set locally (but at the top level) for general use, one set (and more importantly cleared) globally to allow insertion in the header area just once per name.

```

\l__talk_section_tl
\g__talk_section_tl
\l__talk_subsection_tl
\g__talk_subsection_tl
\l__talk_subsubsection_tl
\g__talk_subsubsection_tl
66 \tl_new:N \l__talk_section_tl
67 \tl_new:N \g__talk_section_tl
68 \tl_new:N \l__talk_subsection_tl
69 \tl_new:N \g__talk_subsection_tl
70 \tl_new:N \l__talk_subsubsection_tl
71 \tl_new:N \g__talk_subsubsection_tl

```

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

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

```

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

```

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

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

```

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

```

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

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

```

114 \seq_set_from_clist:Nn \l_tmpa_seq
115   { section , subsection , subsubsection }
116 \seq_map_indexed_inline:Nn \l_tmpa_seq
117   {
118     \use:e
119     {
120       \NewDocumentCommand \exp_not:c { insert #2 } { }
121       {
122         \exp_not:N \tl_use:N
123         \exp_not:c { l__talk_ #2 _tl }
124       }
125       \NewDocumentCommand \exp_not:c {#2}
126       { s D <> { all } = { shorttitle } 0 {##4} m }
127       {
128         \exp_not:N \bool_if:NF \exp_not:N \l__talk_frame_bool
129         {
130           \__talk_if_overlay:nT {##2}
131           { \exp_not:c { __talk_head_ #1 :Nnn } ##1 {##3} {##4} }
132         }
133       }
134       \cs_new_protected:Npn \exp_not:c { __talk_head_ #1 :Nnn } ##1##2##3
135       {
136         \exp_not:N \refstepcounter {#2}
137         \UseTaggingSocket { sec / end } { \use:c { toplevel@ #2 } }
138         \UseTaggingSocket { sec / begin }
139         {
140           { \use:c { toplevel@ #2 } }
141           {
142             tag =
143             \exp_not:N \UseStructureName
144             { sec / \use:c { toplevel@ #2 } }
145           }
146         }
147         \tl_set:Nn \exp_not:c { l__talk_ #2 _tl } {##3}
148         \keys_set:nn { talk / heading } { shorttitle = {##3} , ##2 }
149         \exp_not:N \bool_if:NT ##1
150         {
151           \tl_clear:N \exp_not:N \l__talk_sec_bookmark_tl
152           \tl_clear:N \exp_not:N \l__talk_sec_running_tl
153           \tl_clear:N \exp_not:N \l__talk_sec_toc_tl
154           \bool_set_false:N \exp_not:N \l__talk_sec_numbered_bool
155         }
156         \UseTaggingSocket { talk / sec / title } {#2}
157         \str_if_eq:nnT {#2} { section }
158         { \tl_clear:N \exp_not:N \l__talk_subsubsection_tl }
159         \str_if_eq:nnF {#2} { subsubsection }

```

```

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

1.3 Table of contents

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

```

220 \cs_gset_protected:Npn \@starttoc #1
221 {
222     \begingroup
223     \makeatletter
224     \bool_if:NTF \g__talk_frame_tag_bool
225     { \@starttoc@aux@nn }
226     { \use_i:nn }
227     {#1} { \input { \jobname . #1 } }
228     \legacy_if:nT { @filesw }
229     {
230         \AddToHook { enddocument / afterlastpage }
231         {
232             \expandafter \newwrite \csname tf@ #1 \endcsname
233             \immediate \openout \csname tf@ #1 \endcsname \jobname .#1 \relax
234         }
235     }
236     \@nobreakfalse
237     \endgroup
238 }
239 \cs_new_protected:Npn \@starttoc@aux@nn #1#2
240 {
241     \UseTaggingSocket { toc / starttoc / before } {#1}
242     #2
243     \UseTaggingSocket { toc / starttoc / after } {#1}
244 }

```

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

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

```

245 \NewDocumentCommand \tableofcontents { 0 { } }
246 {
247   \group_begin:
248     \@starttoc { toc }
249   \group_end:
250 }

```

(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 251 \cs_new_protected:Npn \l@section #1#2
\__talk_toc_dest:n 252 { \__talk_toc_aux:nnnn { 1 } { \bfseries \color { structure } } {#1} {#2} }
\__talk_toc_dest:w 253 \cs_new_protected:Npn \l@subsection #1#2
\__talk_toc_level:nnnn 254 {
255   \__talk_toc_aux:nnnn
256   { 2 }
257   {
258     \skip_set:Nn \leftskip { 2em }
259     \color_ensure_current:
260   }
261   {#1} {#2}
262 }
263 \cs_new_protected:Npn \l@subsubsection #1#2
264 {
265   \__talk_toc_aux:nnnn
266   { 3 }
267   {
268     \skip_set:Nn \leftskip { 4em }
269     \color_ensure_current:
270     \footnotesize
271   }
272   {#1} {#2}
273 }
274 \cs_new_protected:Npn \__talk_toc_aux:nnnn #1#2#3#4
275 {
276   \int_compare:nNnTF { \value { section } } < 1
277   { \use:n }
278   { \__talk_toc_dest:n }
279   { \__talk_toc_level:nnnn {#1} {#2} {#3} {#4} }
280 }

```

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.

```

281 \cs_new_protected:Npn \__talk_toc_dest:n
282 {
283   \exp_after:wN \__talk_toc_dest:w \@contentsline@destination
284   . 0 . 0 . 0 . \q_stop
285 }

```

```

286 \cs_new_protected:Npn \__talk_toc_dest:w #1 . #2 . #3 . #4 . #5 \q_stop #6
287 {
288   \bool_lazy_or:nnTF
289   {
290     \bool_lazy_and_p:nn
291     { \int_compare_p:nNn { \value { section } } = {#2} }
292     { \int_compare_p:nNn { \value { subsection } } = 0 }
293   }
294   {
295     \bool_lazy_and_p:nn
296     { \int_compare_p:nNn { \value { section } } = {#2} }
297     { \int_compare_p:nNn { \value { subsection } } = {#3} }
298   }
299   {#6}
300   {
301     \opacity_begin:n { 0.2 }
302     #6
303     \opacity_end:
304   }
305 }
306 \cs_new_protected:Npn \__talk_toc_level:nnnn #1#2#3#4
307 {
308   \int_compare:nNnF {#1} > { \value { tocdepth } }
309   {
310     \group_begin:
311     \noindent
312     #2
313     \UseHookWithArguments { contentsline / text / before } { 4 }
314     {#1} {#3} {#4} { \@contentsline@destination }
315     #3
316     \UseHookWithArguments { contentsline / text / after } { 4 }
317     {#1} {#3} {#4} { \@contentsline@destination }
318     \UseHookWithArguments { contentsline / page / before } { 4 }
319     {#1} {#3} {#4}
320     { \@contentsline@destination }
321     \UseHookWithArguments { contentsline / page / after } { 4 }
322     {#1} {#3} {#4}
323     { \@contentsline@destination }
324     \par
325     \group_end:
326     \vfil
327   }
328 }

```

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

```

329 \setcounter { tocdepth } { 2 }

```

1.4 Block environments

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

quote (*env.*) 330 \NewDocumentEnvironment { description } { } { } { }

quotation (*env.*) 331 \NewDocumentEnvironment { quote } { } { } { }

verse (*env.*) 332 \NewDocumentEnvironment { quotation } { } { } { }

stdquote (*env.*) 333 \NewDocumentEnvironment { verse } { } { } { }

stdquotation (*env.*)

stdverse (*env.*)

```

334 \AddToHook { begindocument / before }
335 {
336   \clist_map_inline:nn { quote , quotation , verse }
337   {
338     \NewEnvironmentCopy { std #1 } {#1}
339     \RenewDocumentEnvironment {#1} { d <> !0 { } }
340     {
341       \__talk_action_begin:n {##1}
342       \begin { std #1 } [ {##2} ]
343       \ignorespaces
344     }
345     {
346       \end { std #1 }
347       \__talk_action_end:
348     }
349   }
350 }

```

`block (env.)`

```

351 \NewDocumentEnvironment { block } { d <> m }
352 {
353   \__talk_action_begin:n {#1}
354   \par
355   \vbox_set:Nw \l__talk_tmp_box
356   \group_begin:
357   \medskip
358   \leavevmode
359   \normalfont \large \bfseries
360   \color { structure }
361   #2
362   \par
363   \medskip
364   \group_end:
365 }
366 {
367   \vbox_set_end:
368   \box_use:N \l__talk_tmp_box
369   \par
370   \__talk_action_end:
371 }

```

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.

```

372 \AddToHook { begindocument / before }
373 {
374   \NewCommandCopy \stditem \item
375   \RenewDocumentCommand \item { d <> o }
376   {

```



```

377     \IfNoValueTF {#2}
378     { \stditem }
379     { \stditem [ {#2} ] }
380   \IfNoValueTF {#1}
381   {
382     \exp_after:wN \__talk_item_parse_spec:w
383     \l__talk_action_spec_str < all > \q_stop
384   }
385   { \__talk_item_parse_spec:n {#1} }
386 }
387 }

```

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.

```

388 \cs_new_protected:Npn \__talk_item_parse_spec:w #1 < #2 > #3 \q_stop
389 { \__talk_item_parse_spec:n {#2} }
390 \cs_new_protected:Npn \__talk_item_parse_spec:n #1
391 {
392   \bool_lazy_or:nnF
393   { \tl_if_blank_p:n {#1} }
394   { \str_if_eq_p:nn {#1} { all } }
395   {
396     \tl_set:Nx \l__talk_list_end_tl
397     {
398       \exp_not:N \int_compare:nNnT \tex_currentgrouplevel:D =
399       { \int_eval:n { \tex_currentgrouplevel:D + 1 } }
400       {
401         \__talk_action_end:
402         \tl_clear:N \exp_not:N \l__talk_list_end_tl
403       }
404     }
405     \__talk_action_begin:n {#1}
406   }
407 }

```

(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`

```

408 \tl_new:N \l__talk_list_end_tl

```

(End of definition for `\l__talk_list_end_tl`.)

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

```

409 \cs_if_exist:NTF \l__block_topsepadd_skip
410 {
411   \cs_gset_protected:Npn \__block_inter_item:
412   {
413     \legacy_if:nT { @inlabel }

```

```

414     { \indent \par }
415 \mode_if_horizontal:T
416 {
417     \__block_skip_remove_last:
418     \__block_skip_remove_last:
419     \par
420 }
421 \l__talk_list_end_tl
422 \__kernel_list_item_end:
423 \__kernel_list_item_begin:
424 \addpenalty \@itempenalty
425 \addvspace \itemsep
426 }

```

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

```

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

```

```

467         {
468             \skip_vertical:n { - \l_tmpa_skip }
469             \skip_vertical:n
470             { \l_tmpa_skip + \parskip - \@outerparskip }
471         }
472         \addpenalty \@endparpenalty
473         \addvspace \l_block_topsepadd_skip
474     }
475     \socket_use:n { block / endpe }
476 }
477 }
478 }

```

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

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

```

479 {
480     \AddToHook { item / after }
481     { \l_talk_list_end_tl }
482 }

```

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

`enumerate (env.)`

`description (env.)`

```

483 \AddToHook { begindocument / before }
484 {
485     \clist_map_inline:nn { itemize , enumerate , description }
486     {
487         \IfFormatAtLeastTF { 2026-06-01 }
488         {
489             \RenewDocumentEnvironment {#1} { = { action-spec } !o }
490             { \SimpleBlockEnv {#1} {##1} }
491             { \BlockEnvEnd }
492         }
493         {
494             \RenewDocumentEnvironment {#1} { = { action-spec } !o }
495             {
496                 \IfNoValueTF {##1}
497                 { \UseInstance { blockenv } {#1} { } }
498                 { \UseInstance { blockenv } {#1} {##1} }
499             }
500             { \endblockenv }
501         }
502     }
503 }

```

And add the structural color to item labels.

```

504 \AddToHook { begindocument / before }
505 {
506     \EditInstance { item } { basic }
507     { label-format = \color { structure } #1 }
508     \EditInstance { item } { description }

```

```

509     { label-format = \normalfont \bfseries \color { structure } #1 }
510   }

```

`\l__talk_action_spec_str`

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

```

511 \IfFormatAtLeastTF { 2026-06-01 }
512 {
513   \keys_define:nn { template / block / std }
514     { action-spec .str_set:N = \l__talk_action_spec_str }
515 }
516 {
517   \keys_define:nn { template / block / display }
518     { action-spec .str_set:N = \l__talk_action_spec_str }
519 }

```

(End of definition for `\l__talk_action_spec_str`.)

1.6 Theorems, *etc.*

`\newtheorem`
`\stdnewtheorem`

We need to extend the creation of theorems by adding the overlay argument. This means 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.

```

520 \NewCommandCopy \stdnewtheorem \newtheorem
521 \RenewDocumentCommand \newtheorem { s m o m o }
522 {
523   \use:e
524   {
525     \exp_not:N \stdnewtheorem
526     \IfBooleanT #1 { * }
527     { \exp_not:n {#2} }
528     \IfNoValueF {#3} { \exp:not:n { [ {#3} ] } }
529     { \exp_not:n {#4} }
530     \IfNoValueF {#5} { \exp:not:n { [ {#5} ] } }
531   }
532   \NewEnvironmentCopy { std #2 } {#2}
533   \RenewDocumentEnvironment {#2} { D <> { all } o }
534   {
535     \__talk_action_begin:n {##1}
536     \IfNoValueTF {##2}
537     { \begin { std #2 } }
538     { \begin { std #2 } [ {##2} ] }
539     \ignorespaces
540   }
541   {
542     \end { std #2 }
543     \__talk_action_end:
544   }
545 }

```

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

```

546 </class>

```

Part X

ltx-talk-title – Title pages

1 ltx-talk-title implementation

Start the DocStrip guards.

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

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

Allow empty values for author and title.

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

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

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

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

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

```

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

```

Standard settings are taken from beamer with minor adjustments.

```

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

```

(End of definition for \l__talk_titlelem_after_skip and others.)

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

```

\l__talk_titlepage_order_clist
\l__talk_titlepage_alignment_tl
\l__talk_titlepage_framestyle_tl
\l__talk_frame_alignment_tl
94 \NewTemplateType { titlepage } { 0 }
95 \DeclareTemplateInterface { titlepage } { talk } { 0 }
96 {
97     element-order : commalist =
98     {
99         title      ,
100        subtitle   ,
101        author     ,
102        institute  ,
103        date
104    } ,
105    framestyle : tokenlist = talk ,
106    horizontal-alignment : choice { left , center , right } = center ,
107    vertical-alignment : choice { bottom , center , stretch , top } = center
108 }
109 \DeclareTemplateCode { titlepage } { talk } { 0 }
110 {
111     element-order = \l__talk_titlepage_order_clist ,
112     framestyle = \l__talk_titlepage_framestyle_tl ,
113     horizontal-alignment =
114     {
115         left = \tl_set:Nn \l__talk_titlepage_alignment_tl { flushleft } ,
116         center = \tl_set:Nn \l__talk_titlepage_alignment_tl { center } ,
117         right = \tl_set:Nn \l__talk_titlepage_alignment_tl { flushright }
118     } ,
119     vertical-alignment =
120     {
121         bottom = \tl_set:Nn \l__talk_frame_alignment_tl { bottom } ,

```

```

122     center = \tl_set:Nn \l__talk_frame_alignment_tl { center } ,
123     stretch = \tl_set:Nn \l__talk_frame_alignment_tl { stretch } ,
124     top = \tl_set:Nn \l__talk_frame_alignment_tl { top }
125   }
126 }
127 {
128   \tl_if_empty:NF \l__talk_titlepage_framestyle_tl
129   { \exp_args:NV \thispagestyle \l__talk_titlepage_framestyle_tl }
130   \begin { \l__talk_titlepage_alignment_tl }
131     \cs_set_protected:Npn \and { \quad }
132     \clist_map_inline:Nn \l__talk_titlepage_order_clist
133     {
134       \ExpandArgs { nnv } \UseInstance { titlepage-element }
135       {##1} { @ ##1 }
136     }
137   \end { \l__talk_titlepage_alignment_tl }
138 }

```

(End of definition for \l__talk_titlepage_order_clist and others.)

\maketitle A very simple setup.

```

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

```

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

```

149 </class>

```


Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
\(77
\)	77
@ commands:	
\@decode_overlay_+:nw	146
@starttoc@aux internal commands:	
\starttoc@aux__talknn	220
\\	347
A	
\abovecaptionskip	156, 158
\action	141
actionenv (env.)	141
\addcontentsline	189
\addcontentslinebookmark	170, 182
\addcontentslinebookmarkOff	170, 187
\addcontentslinebookmarkOn	170
\addcontentslinebookmarkReset	172
\addpenalty	424, 472
\AddToHook	54, 81, 230, 284, 334, 349, 372, 419, 447, 448, 480, 483, 504
\addtolength	48
\addvspace	425, 473
\againframe	508
\alert	41, 119
alertenv (env.)	126
\alt	214
\and	131
\arabic	416
\arraycolsep	25
\arrayrulewidth	27
\AssignSocketPlug	194
\AssignStructureRole	125, 127
\AssignTaggingSocketPlug	219
\author	15
B	
\begin	122, 126, 130, 144, 342, 432, 537, 538
\begingroup	163, 222, 409
\belowcaptionskip	157, 159
\bfseries	21, 39, 252, 359, 509
\bigskipamount	18
block (env.)	351
block commands:	
\g_block_nesting_depth_int	442
block internal commands:	
__block_debug_typeout:n	432
__block_if_list:TF	450, 461
__block_inter_item:	409, 411
\l_block_level_incr_bool	440
__block_skip_remove_last:	417, 418, 454, 455
__block_skip_set_to_last:N	465
\l_block_topsepadd_skip	62, 64, 409, 473
\l_block_transparent_level_bool	433, 436
\BlockEnvEnd	409, 491
bool commands:	
\bool_do_while:Nn	28
\bool_gset_eq:NN	48, 53
\bool_gset_false:N	37, 459
\bool_gset_true:N	223, 233, 239, 451
\bool_if:NTF	6, 39, 44, 46, 50, 58, 60, 66, 68, 79, 82, 99, 99, 128, 141, 149, 184, 197, 201, 224, 242, 275, 435, 438, 439, 465
\bool_lazy_and:nnTF	21, 49, 236, 335
\bool_lazy_and_p:nn	290, 295
\bool_lazy_or:nnTF	5, 18, 22, 288, 392
\bool_lazy_or_p:nn	24
\bool_new:N	3, 3, 4, 4, 8, 9, 14, 137, 139, 140, 410, 411, 412
\bool_set_eq:NN	44, 176, 180
\bool_set_false:N	28, 29, 30, 31, 37, 154, 169, 472, 494
\bool_set_true:N	25, 52, 68, 71, 103, 164, 167, 201, 220, 224, 230, 235, 426, 481, 503
\c_false_bool	15
\c_true_bool	14, 193, 210
box commands:	
\box_dp:N	36, 63
\box_gclear:N	92
\box_ht:N	62, 68, 293, 318
\box_if_empty:NTF	105
\box_new:N	5, 62, 171
\box_set_dp:Nn	66
\box_set_ht:Nn	60
\box_use:N	368
\box_use_drop:N	26, 71, 271, 298, 302, 304, 345
\box_wd:N	41
box internal commands:	
__box_dim_eval:n	33, 36, 41, 44
__box_set_to_wd:	40, 45

C

`\clearpage` 111
 clist commands:
 `\clist_const:Nn` 58
 `\clist_if_in:NnTF` 66, 200
 `\clist_map_break:` 240
 `\clist_map_inline:Nn` ... 132, 203, 332
 `\clist_map_inline:nn`
 3, 87, 105, 139, 336, 485
 `\clist_new:N` 11, 15
 `\clist_pop:NNTF` 328
 `\clist_set:Nn` 199
`\color` 4, 11, 56, 252, 360, 507, 509
 color commands:
 `\color_ensure_current:` .. 63, 259, 269
 `\color_group_begin:` 34, 46
 `\color_group_end:` 34
 `\color_math:nn` 9, 26
 `\color_math:nnn` 10, 27
 `\color_select:n` 7,
 16, 35, 37, 45, 46, 69, 229, 273, 326, 367
 `\color_select:nn` 8, 17, 38
 color internal commands:
 `_color_backend_reset:` 64
`\colorlet` 68
`column (env.)` 78
`columns (env.)` 11
`\columnwidth` 20, 87, 122
 cs commands:
 `\cs_generate_variant:Nn`
 ... 7, 8, 9, 10, 56, 58, 59, 60, 192, 199
 `\cs_gset:Npe` 429
 `\cs_gset:Npn` 111, 112, 113
 `\cs_gset_eq:NN` 56
 `\cs_gset_nopar:Npn` 49, 57, 64
 `\cs_gset_protected:Npe` 428
 `\cs_gset_protected:Npn`
 29, 38, 48, 161, 220, 342, 411
 `\cs_if_exist:NTF` .. 127, 288, 409, 433
 `\cs_if_exist_p:N` 336
 `\cs_if_exist_use:NTF` ... 155, 186, 203
 `\cs_new:Npn` 7, 8, 34, 35, 36, 37, 38, 39,
 40, 41, 42, 170, 277, 355, 415, 416, 421
 `\cs_new_eq:NN` . 6, 7, 160, 195, 414, 424
 `\cs_new_nopar:Npn` 3, 398
 `\cs_new_protected:Npe` 64, 80, 106
 `\cs_new_protected:Npn`
 10, 11, 17, 18, 19, 34,
 35, 38, 40, 42, 44, 45, 48, 51, 52, 54,
 54, 56, 56, 61, 62, 64, 71, 73, 77, 87,
 93, 96, 97, 103, 113, 113, 119, 125,
 131, 134, 134, 146, 150, 152, 152,
 155, 162, 162, 164, 167, 169, 173,
 174, 180, 191, 192, 193, 195, 197,

200, 207, 208, 217, 225, 239, 251,
 253, 258, 263, 274, 281, 286, 306,
 307, 333, 361, 375, 388, 390, 407,
 423, 431, 433, 447, 455, 508, 517, 519
`\cs_set:Npn` 14, 15, 74
`\cs_set_eq:NN` 61, 62,
 63, 64, 164, 380, 381, 395, 396, 406, 407
`\cs_set_nopar:Npn`
 130, 166, 167, 168, 169,
 373, 375, 379, 383, 385, 390, 400, 405
`\cs_set_protected:Npn`
 40, 114, 131, 160, 227
`\cs_to_str:N` 365, 370
`\csname` 167, 232, 233

D

`\date` 15
`\day` 20
`\DeclareColor` 65, 71, 72, 73
`\DeclareInstance` 43,
 79, 81, 83, 85, 87, 96, 133, 236,
 237, 238, 239, 240, 241, 242, 283, 348
`\DeclareInstanceCopy` 286, 351
`\DeclareTemplateCode`
 ... 23, 54, 93, 109, 110, 215, 254, 301
`\DeclareTemplateInterface`
 16, 45, 91, 95, 105, 208, 244, 291
`\definecolor` 69
`description (env.)` 330, 483
 dim commands:
 `\dim_compare:nNnTF` 291, 466
 `\dim_compare_p:nNn` 338
 `\dim_const:Nn` 111, 117
 `\dim_eval:n` 51, 52, 53
 `\dim_max:nn` 58, 317
 `\dim_set:Nn` 19, 86
 `\dim_set_eq:NN` 20, 87
 `\dim_to_decimal:n` 104
 `\dim_use:N` 132, 133
 `\c_zero_dim` 466
`\DocumentMetadata` 7
`\doublerulesep` 28

E

`\EditInstance` 287, 352, 506, 508
`\emph` 350
`\end` . 136, 137, 137, 146, 346, 435, 458, 542
`\endblockenv` 409, 500
`\endcsname` 167, 232, 233
`\endfloatenv` 134, 147
`\endgroup` 169, 237, 422
`enumerate (env.)` 483
 environments:
 `actionenv` 141

<code>\msg_warning:nn</code>	27, 341	pdfxform commands:	
		<code>\pdfxform_new:nnn</code>	438
		<code>\pdfxform_use:n</code>	442
N			
<code>\NeedsDocumentMetadata</code>	17	prg commands:	
<code>\NewCommandCopy</code>	4, 5, 6, 173, 365, 374, 385, 422, 520, 532	<code>\prg_generate_conditional_-</code> variant:Nnn	10
<code>\newcounter</code>	20, 105, 106, 107, 150	<code>\prg_new_protected_conditional:Npnn</code>	3, 3
<code>\NewDocumentCommand</code>	5, 10, 11, 34, 39, 65, 107, 119, 120, 125, 129, 139, 141, 214, 220, 237, 245, 247, 524	<code>\prg_return_false:</code>	8, 9
<code>\NewDocumentEnvironment</code>	11, 78, 114, 126, 134, 141, 149, 265, 282, 330, 331, 332, 333, 351, 476, 498	<code>\prg_return_true:</code>	7, 8
<code>\NewEnvironmentCopy</code>	338, 532	<code>\ProcessedArgument</code>	14, 15
<code>\newlabel</code>	413	<code>\ProcessKeyOptions</code>	103
<code>\newlength</code>	156, 157	prop commands:	
<code>\NewSocketPlug</code>	181	<code>\prop_gput:Nnn</code>	5, 6
<code>\NewTaggingSocket</code>	202	property commands:	
<code>\NewTaggingSocketPlug</code>	203	<code>\property_new:nnnn</code>	9, 417
<code>\NewTemplateType</code>	15, 44, 90, 94, 104, 207, 243, 290	<code>\property_record:nn</code>	30, 83, 420
<code>\newtheorem</code>	65, 520	<code>\property_ref:nn</code>	15, 421
<code>\newwrite</code>	232	<code>\protect</code>	198
<code>\noindent</code>	31, 265, 311, 312	<code>\providecommand</code>	170, 171, 172, 173
<code>\normalfont</code>	42, 50, 248, 359, 509	<code>\ProvidesExplClass</code>	3
<code>\normalsize</code>	166	<code>\put</code>	57
<code>\number</code>	20, 22	Q	
<code>\numberline</code>	198	<code>\quad</code>	131
O			
<code>\obeyedline</code>	17, 74	quark commands:	
<code>\only</code>	46, 129	<code>\quark_if_recursion_tail_stop:N</code>	154
<code>onlyenv (env.)</code>	134	<code>\quark_if_recursion_tail_stop_-</code> do:Nn	171, 182
<code>\onslide</code>	42, 220	<code>\quark_if_recursion_tail_stop_-</code> do:nn	47
opacity commands:		<code>\q_recursion_stop</code>	38, 149, 174
<code>\opacity_begin:n</code>	39, 301	<code>\q_recursion_tail</code>	38, 149, 174
<code>\opacity_end:</code>	41, 303	<code>\q_stop</code>	75, 83, 109, 114, 115, 124, 158, 162, 204, 207, 284, 286, 383, 388
<code>\openout</code>	233	quotation (env.)	330
<code>\or</code>	5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16	quote (env.)	330
<code>overlayarea (env.)</code>	265	R	
<code>overprint (env.)</code>	282	<code>\raggedright</code>	90, 145, 262
P		<code>\refstepcounter</code>	136
<code>\pagecolor</code>	43	<code>\relax</code>	233
<code>\pagestyle</code>	409	<code>\relsize</code>	109
<code>\paperheight</code>	57, 58	<code>\RenewCommandCopy</code>	27
<code>\paperwidth</code>	58, 320, 368, 369	<code>\RenewDocumentCommand</code> 11, 15, 21, 22, 27, 30, 43, 174, 366, 375, 386, 400, 425, 521	
<code>\par</code>	31, 33, 14, 28, 34, 39, 73, 80, 95, 100, 162, 168, 260, 274, 305, 324, 354, 362, 369, 414, 419, 456, 461	<code>\RenewDocumentEnvironment</code>	339, 467, 489, 489, 494, 533
<code>\parsep</code>	53, 61, 62, 68, 69	<code>\RequirePackage</code>	3, 107, 128, 146, 149, 150, 153, 156, 162, 163, 171, 384
<code>\parskip</code>	470	<code>\reuseframe</code>	508
<code>\partopsep</code>	71	<code>\rmdefault</code>	164
<code>\pause</code>	42, 247	<code>\rule</code>	58
		rule commands:	
		<code>\rule:nnn</code>	48, 368

S

scan commands:	
\scan_stop:	54
\scriptsize	84
\section	105, 114
seq commands:	
\seq_gpop_left:NN	186
\seq_gpush:Nn	176
\seq_gput_right:Nn	165
\seq_gremove_all:Nn	108, 109, 110
\seq_gset_from_clist:Nn	138
\seq_map_indexed_inline:Nn	116
\seq_map_inline:Nn	145, 152, 157
\seq_new:N	137, 172
\seq_set_from_clist:Nn	114
\l_tmpa_seq	114, 116
\setcounter	24, 329
\setlength	25, 26, 27, 28, 29, 31, 32, 33, 43, 44, 45, 46, 47, 71, 158, 159
\setmainfont	157
\setmathfont	159
\setsansfont	158
\SetTemplateKeys	121
\sfdefault	164
\SimpleBlockEnv	490
\skip	30
skip commands:	
\skip_horizontal:n	
.....	32, 270, 317, 346, 363, 369
\skip_if_eq_p:n	22
\skip_new:N	17
\skip_set:Nn	258, 268
\skip_set_eq:NN	20
\skip_vertical:n	33, 115, 117, 121, 123, 127, 129, 133, 135, 468, 469
\l_tmpa_skip	465, 466, 468, 470
socket commands:	
\socket_use:n	475
\space	19, 21
\stdcolor	4
\stdemph	350
\stdfootnote	173, 178, 179
\stdincludegraphics	384
\stditem	374, 378, 379
\stdmathcolor	4
\stdnewtheorem	520
stdquotation (env.)	330
stdquote (env.)	330
\stdtextbf	350
\stdtextcolor	4
\stdtextit	350
\stdtextmd	350
\stdtextnormal	350
\stdtextrm	350

\stdtextsc	350
\stdtextsf	350
\stdtextsl	350
\stdtexttt	350
\stdtextup	350
stdverse (env.)	330
\stepcounter	20
str commands:	
\str_clear:N	
.....	21, 148, 470, 479, 492, 501, 528
\str_if_empty:NTF ..	98, 119, 129, 427
\str_if_empty_p:N	50
\str_if_eq:nnTF ..	18, 69, 95, 157, 159, 166
\str_if_eq_p:nn	6, 7, 24, 394
\str_new:N	10, 12, 13, 16, 79, 138
\str_put_right:Nn	157, 193
\str_replace_all:Nnn	21, 23, 59
\str_set:Nn	19, 27, 73, 77, 131
\str_set_eq:NN	178
\string	413
\subsection	105, 114
\subsubsection	105, 114
\subtitle	34
sys commands:	
\c_sys_engine_str	24
\sys_if_engine luatex:TF ..	151, 445
\sys_if_engine luatex_p:	19
\sys_if_engine opentype:TF	147
\sys_if_engine pdftex_p:	20

T

\tabbingsep	29
\tabcolsep	26
table (env.)	139
\tableename	150
\tableofcontents	245
tag commands:	
\tag_get:n	450
\tag_mc_begin:n	197, 204, 457
\tag_mc_end:	195, 202, 463
\tag_resume:n	84, 194, 462
\tag_struct_begin:n .	91, 196, 210, 449
\tag_struct_end: ...	92, 203, 217, 453
\tag_suspend:n	72, 205, 458
tag internal commands:	
\g_tag_title_author_tl	19
\g_tag_title_title_tl	31
\tagpdfparaOff	61
\tagpdfsetup	154, 173
talk internal commands:	
__talk_action_alert:N	42, 42, 122, 127
__talk_action_begin:n	13, 91, 141, 143, 144, 150, 152, 341, 353, 405, 535
__talk_action_begin:w .	141, 157, 162

__talk_action_begin_auxi:n [141](#), [160](#), [163](#), [164](#)
 __talk_action_begin_auxii:n ... [141](#), [170](#), [173](#)
 __talk_action_end: [33](#), [29](#), [96](#), [141](#), [146](#), [148](#), [151](#), [195](#), [347](#), [370](#), [401](#), [543](#)
 __talk_action_invisible:N [48](#), [48](#), [193](#)
 __talk_action_invisible_end:N . [48](#), [54](#), [210](#)
 __talk_action_only:N [64](#), [64](#), [135](#)
 __talk_action_only_end:N [64](#), [77](#), [136](#)
 \l__talk_action_spec_str [158](#), [168](#), [383](#), [511](#)
 __talk_action_uncover:N . [97](#), [97](#), [189](#)
 __talk_action_uncover_end:N ... [97](#), [103](#), [191](#)
 __talk_action_visible:N [48](#), [56](#)
 __talk_action_visible_end:N . [48](#), [62](#)
 \l__talk_aspect_ratio_str .. [64](#), [123](#)
 \g__talk_cnt_reset_seq [108](#), [109](#), [110](#), [137](#), [152](#), [157](#), [165](#)
 __talk_cnt_restore: ... [100](#), [150](#), [155](#)
 __talk_cnt_save: [93](#), [150](#), [150](#)
 __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 . [97](#), [106](#), [106](#)
 __talk_decode_action:w [106](#), [108](#), [113](#)
 \l__talk_decode_action_str [13](#), [21](#), [119](#), [129](#), [131](#), [179](#), [187](#)
 \l__talk_decode_actions_bool ... [14](#), [28](#), [181](#), [189](#)
 \l__talk_decode_actions_clist ... [14](#)
 \l__talk_decode_actions_str [14](#)
 \l__talk_decode_arg_str [10](#), [27](#), [32](#), [123](#), [137](#), [142](#), [185](#)
 __talk_decode_check:n . [150](#), [197](#), [197](#)
 __talk_decode_check:nw [197](#), [204](#), [207](#)
 __talk_decode_check_range:nnn . [197](#), [213](#), [214](#), [227](#)
 __talk_decode_check_single:nn . [197](#), [210](#), [217](#)
 __talk_decode_mode:n [54](#), [64](#), [64](#)
 __talk_decode_mode:nn ... [88](#), [91](#), [93](#)
 __talk_decode_mode:w [64](#), [74](#), [80](#)
 __talk_decode_mode_aux:n [64](#)
 \l__talk_decode_modes_bool [8](#), [31](#), [51](#), [68](#), [103](#)
 __talk_decode_overlay_:nw [146](#)
 __talk_decode_overlay_aux:nNN . [146](#), [165](#), [168](#), [169](#)
 __talk_decode_overlay_offset:nNn [146](#), [173](#), [178](#), [188](#), [191](#)
 __talk_decode_overlay_offset:nNnN [146](#), [177](#), [180](#), [189](#)
 __talk_decode_overlays:nN [146](#), [149](#), [152](#), [158](#), [195](#)
 __talk_decode_overlays:nn [99](#), [120](#), [133](#), [146](#), [146](#)
 \l__talk_decode_overlays_bool . [3](#), [6](#), [25](#), [29](#), [52](#), [71](#), [177](#), [184](#), [189](#), [191](#)
 \l__talk_decode_overlays_clist .. [11](#)
 \l__talk_decode_overlays_str ... [11](#), [50](#), [98](#)
 __talk_decode_parse:n [5](#), [17](#), [17](#), [175](#), [188](#)
 __talk_decode_parse:w . [17](#), [38](#), [45](#), [56](#)
 __talk_decode_parse_auxi:n [17](#), [18](#), [19](#)
 __talk_decode_parse_auxii:n ... [17](#), [32](#), [35](#)
 \l__talk_decode_step_bool [9](#), [37](#), [39](#), [164](#)
 \l__talk_decode_trailing_bool .. [4](#), [30](#), [224](#), [230](#), [242](#)
 \l__talk_float_alignment_tl [103](#), [115](#), [116](#), [117](#), [126](#), [136](#)
 \l__talk_fontsize_dim ... [64](#), [104](#), [109](#)
 \l__talk_footelem_color_tl [207](#)
 \l__talk_footelem_font_tl [207](#)
 \l__talk_footelem_left_skip [207](#)
 \l__talk_footelem_right_skip ... [207](#)
 \l__talk_footer_bg_tl [290](#)
 \l__talk_footer_fg_tl [290](#)
 \l__talk_footer_font_tl [290](#)
 \l__talk_footer_left_skip [290](#)
 \l__talk_footer_order_clist [290](#)
 \l__talk_footer_right_skip [290](#)
 \l__talk_footer_sep_tl [290](#)
 \g__talk_footnote_box [92](#), [105](#), [108](#), [171](#), [183](#), [185](#)
 \g__talk_footnote_overlay_seq .. [172](#), [176](#), [186](#)
 \l__talk_frame_alignment_tl [94](#), [103](#), [167](#), [181](#)
 \l__talk_frame_bool [128](#), [141](#), [410](#), [426](#)
 \g__talk_frame_int [15](#), [30](#), [83](#), [279](#), [413](#), [418](#), [425](#)
 \l__talk_frame_name_str [168](#), [427](#), [429](#), [432](#), [434](#), [435](#), [470](#), [479](#), [492](#), [501](#), [528](#)
 __talk_frame_notag:n ... [55](#), [455](#), [455](#)

_talk_frame_overprint:	_talk_if_overlay:nTF
. 277 , 277 , 289 , 292 , 296 ,	12 , 13 , 13 , 23 , 34 , 38 , 45 , 46 , 130 ,
309 , 311 , 312 , 315 , 317 , 324 , 326 , 331	131 , 216 , 229 , 239 , 369 , 388 , 403 , 427
_talk_frame_process:nn	_talk_item_parse_spec:n
. 423 , 423 , 473 , 483 , 495 , 504 , 522 372 , 385 , 389 , 390
_talk_frame_reuse:nn . 508 , 508 , 529	_talk_item_parse_spec:w
_talk_frame_reuse_aux:nn 372 , 382 , 388
. 508 , 512 , 517	_talk_label:n 400 , 404 , 407
_talk_frame_reuse_aux:nnn	_talk_latex_frame:n . . 27 , 422 , 422
. 508 , 518 , 519	\l_talk_list_end_tl
\g_talk_frame_struct_int 56 , 86 , 450 396 , 402 , 408 , 421 , 459 , 481
\g_talk_frame_subtitle_tl 3 , 13 , 91	_talk_metadata_name:n
_talk_frame_tag:n 50 , 447 , 447 331 , 334 , 339 , 355 , 355
\g_talk_frame_tag_bool	_talk_mode:n 3
. 46 , 68 , 82 , 224 , 411 , 451 , 459	_talk_mode:nTF 3 , 12
\l_talk_frame_tagging_str	\l_talk_mode_str 7 , 64 , 70 , 95
. 18 , 19 , 21 , 23 , 46 , 168	\c_talk_modes_clist 58 , 66
_talk_frame_title:n 15 , 38 , 44	_talk_onslide:n . 220 , 222 , 225 , 254
\l_talk_frame_title_bool . . 64 , 465	\g_talk_onslide_tl
_talk_frame_title_tagged:n 43 , 66 , 182 , 199 , 227 , 228 , 232 , 236
. 15 , 47 , 51	_talk_opacity_begin:n
\g_talk_frame_title_tl 38 , 38 , 51 , 52 , 59 , 60 , 95 , 100 , 231
. 3 , 8 , 58 , 90 , 279 , 482	_talk_opacity_end:
\l_talk_frame_verb_bool 38 , 40 , 55 , 63 , 104 , 208 , 233
. 60 , 412 , 438 , 472 , 481 , 494 , 503 , 521	_talk_opacity_group_begin:
\l_talk_frametitle_after_skip 431 , 431 , 447
. 25 , 41	_talk_opacity_group_end:
\l_talk_frametitle_before_skip 431 , 433 , 448
. 26 , 32	\g_talk_opacity_group_int
\l_talk_frametitle_color_tl 430 , 437 , 439 , 443
. 27 , 34 , 35	\l_talk_overlay_all_bool
\l_talk_frametitle_font_tl . . 28 , 36 140 , 167 , 169 , 197
_talk_head_marks:nnnnn	_talk_overlay_arg:n
. 161 , 174 , 174 , 192 3 , 11 , 108 , 115 , 119 , 126 , 134
_talk_head_marks_aux:Nnn 174	_talk_overprint_begin:n
_talk_head_marks_aux:Nnnn 258 , 258 , 266 , 283
. 181 , 188 , 193	_talk_overprint_check_ht:n
_talk_head_section:Nnn 114 282 , 331 , 333 , 342
_talk_head_subsection:Nnn 114	\l_talk_overprint_int . 276 , 280 , 286
_talk_head_subsubsection:Nnn . 114	_talk_overprint_save_ht:
_talk_head_tag:nn 202 , 205 , 208 282 , 287 , 307
\l_talk_header_bg_tl 243	_talk_pagecolor:n 43 , 48 , 49 , 52
\l_talk_header_fg_tl 243	\c_talk_paper_height_dim 111
\l_talk_header_font_tl 243	\c_talk_paper_width_dim 111
\l_talk_header_frametitle_bool 243	\g_talk_pauses_int
\l_talk_header_ht_dim 243 10 , 5 , 42 , 89 , 194 , 252 , 253 , 255
\l_talk_header_left_skip 243	\l_talk_saved_action_str
\l_talk_header_right_skip 243 137 , 178 , 204
_talk_header_tag_begin:n	\l_talk_saved_actions_bool
. 53 , 192 , 192 , 199 137 , 180 , 206
_talk_header_tag_end: . 64 , 192 , 200	\l_talk_saved_overlays_bool
_talk_if_overlay:n 3 , 10 137 , 176 , 201
	\l_talk_sec_bookmark_tl 72 , 151 , 163
	\l_talk_sec_label_tl 72

<code>\l__talk_sec_nameref_tl</code>	72 , 164	<code>\l__talk_uncover_hidden_fp</code>	90
<code>\l__talk_sec_numbered_bool</code> .	72 , 154	<code>\l__talk_vcenter_offset_tl</code> 64 , 69 , 75	
<code>\l__talk_sec_quote_tl</code>	72	<code>__talk_wallpaper_hruler:Nnn</code>	
<code>\l__talk_sec_running_tl</code>	72 , 152		266 , 313 , 361 , 361
<code>\l__talk_sec_subtitle_tl</code>	72	<code>talk/sec/title</code>	202
<code>\l__talk_sec_toc_tl</code>	72 , 153 , 162	<code>\temporal</code>	10 , 237
<code>\g__talk_section_tl</code>	66	TeX and L ^A T _E X 2 _ε commands:	
<code>\l__talk_section_tl</code>	66	<code>\@arabic</code>	7 , 8 , 111 , 112 , 113 , 170 , 415
<code>\l__talk_shuffle_skip</code> . .	17 , 20 , 22 , 34	<code>\@author</code>	3 , 18 , 19
<code>__talk_shuffle_skip:n</code> .	18 , 18 , 39 , 41	<code>\@auxout</code>	322 , 411
<code>__talk_slide:nn</code>	10 , 10 , 445	<code>\@bsphack</code>	402
<code>__talk_slide_aux:nn</code>	10	<code>\@caption</code>	160
<code>__talk_slide_align_bottom:n</code> 113 , 113		<code>\@capytype</code>	130
<code>__talk_slide_align_center:n</code> 113 , 119		<code>\@contentsline@destination</code>	
<code>__talk_slide_align_stretch:n</code> . .			59 , 283 , 314 , 317 , 320 , 323
	113 , 125	<code>\@currentHref</code>	418
<code>__talk_slide_align_top:n</code> .	113 , 131	<code>\@currentlabel</code>	415
<code>__talk_slide_aux:n</code>	10 , 61 , 71	<code>\@currentlabelname</code>	176 , 417
<code>__talk_slide_aux:nn</code>	29 , 34	<code>\@currenvir</code>	458
<code>__talk_slide_begin:</code>	40 , 87 , 87	<code>\@date</code>	3 , 25
<code>\l__talk_slide_box</code>	5 , 94 , 104	<code>\@definecounter</code>	160
<code>\g__talk_slide_continue_bool</code> . .	3 ,	<code>\@endparpenalty</code>	472
	28 , 37 , 45 , 48 , 53 , 99 , 223 , 233 , 239	<code>\@esphack</code>	405
<code>\l__talk_slide_continue_bool</code> . . .		<code>\@evenfoot</code>	381 , 396 , 407
	3 , 44 , 49 , 54	<code>\@evenhead</code>	380 , 395 , 406
<code>__talk_slide_end:</code>	68 , 87 , 96	<code>\@framenummer</code>	413
<code>\g__talk_slide_int</code>		<code>\@ignore</code>	33
	6 , 9 , 26 , 36 , 219 , 222 , 229 , 232 , 237	<code>\@ignoretrue</code>	101
<code>\g__talk_subsection_tl</code>	66	<code>\@inmatherr</code>	458
<code>\l__talk_subsection_tl</code>	66 , 158	<code>\@input</code>	227
<code>\g__talk_subsubsection_tl</code>	66	<code>\@institute</code>	3 , 37
<code>\l__talk_subsubsection_tl</code> . .	66 , 160	<code>\@itempenalty</code>	424
<code>__talk_textcmd_equiv:n</code> .	350 , 371 , 375	<code>\@kernel@reserved@label@data</code> . . .	419
<code>\l__talk_titlelem_after_skip</code>	44	<code>\@listI</code>	56
<code>\l__talk_titlelem_before_skip</code> . .	44	<code>\@listi</code>	49 , 56
<code>\l__talk_titlelem_color_tl</code>	44	<code>\@listii</code>	57
<code>\l__talk_titlelem_font_tl</code>	44	<code>\@listiii</code>	64
<code>\l__talk_titlelem_tag_begin_tl</code> . .	44	<code>\@makecaption</code>	167
<code>\l__talk_titlelem_tag_end_tl</code>	44	<code>\@makefnstext</code>	195
<code>\l__talk_titlepage_alignment_tl</code> .	94	<code>\@mpfootins</code>	30
<code>\l__talk_titlepage_framestyle_tl</code> .	94	<code>\@nbreakfalse</code>	236
<code>\l__talk_titlepage_order_clist</code> . .	94	<code>\@noitemerr</code>	451
<code>__talk_tmp:w</code>	61 , 61 , 114 , 123	<code>\@oddfont</code>	379 , 381 , 390 , 396 , 405 , 407
<code>\l__talk_tmp_box</code>	18 , 26 ,	<code>\@oddhead</code>	375 , 380 , 385 , 395 , 400 , 406
	59 , 60 , 62 , 62 , 63 , 66 , 68 , 70 , 71 , 71 ,	<code>\@outerparskip</code>	470
	74 , 85 , 99 , 261 , 271 , 293 , 298 , 302 ,	<code>\@parboxrestore</code>	88 , 164
	304 , 318 , 318 , 345 , 355 , 368 , 432 , 441	<code>\@setminipage</code>	165
<code>\l__talk_tmp_tl</code>	13 , 19 ,	<code>\@shortauthor</code>	3
	22 , 24 , 63 , 112 , 187 , 188 , 328 , 330 , 331	<code>\@shortdate</code>	3
<code>__talk_toc_aux:nnnn</code>		<code>\@shortinstitute</code>	3
	251 , 252 , 255 , 265 , 274	<code>\@shortsubtitle</code>	3
<code>__talk_toc_dest:n</code>	251 , 278 , 281	<code>\@shorttitle</code>	3
<code>__talk_toc_dest:w</code>	251 , 283 , 286	<code>\@skiphyperreftrue</code>	34 , 131
<code>__talk_toc_level:nnnn</code> .	251 , 279 , 306	<code>\@starttoc</code>	220 , 248

\@starttoc@aux@nn	225, 239	\text_titlecase_first:n	152
\@subtitle	3, 42	\textasteriskcentered	40
\@title	3, 30, 31	\textbf	350
\@totalframes	417	\textbullet	38
\c@figure	150	\textcolor	6, 11
\c@frame	413	\textendash	39
\c@page	170	\textheight	101
\c@pauses	5	\textit	350
\c@section	111	\textmd	350
\c@slide	6	\textnormal	350
\c@subsection	112	\textperiodcentered	41
\c@subsubsection	113	\textrm	350
\c@table	150	\textsc	350
\check@mathfonts	58	\textsf	350
\currentgrouplevel	62	\textsl	350
\fnum@figure	150	\texttt	350
\fnum@table	150	\textup	350
\Gm@bmargin	316	\textwidth	31, 8, 19, 20, 86, 87, 282
\Gm@lmargin	250, 297, 363	\theenumi	34
\Gm@rmargin	252, 298, 346	\theenumii	35
\Gm@tmargin	249	\theenumiii	36
\if@minipage	165	\theenumiv	37
\ifmeasuring@	11	\thefigure	150
\ignorespaces	33	\theframe	413
\l@section	251	\thepage	6, 170, 416
\l@subsection	251	\thepauses	5
\l@subsubsection	251	\thesection	105
\label@in@display	49, 424, 425	\theslide	6
\on@line	432	\thesubsection	105
\protected@write	411	\thesubsubsection	105
\ps@plain	373	\thetable	150
\ps@talk	373	\thispagestyle	129
\ps@wallpaper	373	\tiny	296
\reset@color	62, 63	\title	15
\set@color	61, 63	tl commands:	
\std@definecounter	160	\tl_clear:N	151, 152, 153, 158, 160, 402
\std@label@in@display	424	\tl_clear_new:N	434
\stdreset@color	61	\tl_gclear:N	12, 13, 43, 90, 91, 228
\stdset@color	61	\tl_gput_right:Nn	232
\textsubscript@offset	166	\tl_gset:Nn	8, 13, 18, 25, 30, 37, 42, 312, 315, 435, 482
\textsubscript@space	166	\tl_gset_eq:NN	14, 19, 31
\textsuperscript@offset	166	\tl_if_blank:nTF	37, 86, 102, 117, 132, 177, 179, 186, 212
\textsuperscript@space	166	\tl_if_blank_p:n	23, 393
tex commands:		\tl_if_empty:NTF	34, 68, 128, 182, 199, 228, 272, 325, 334, 364
\tex_currentgrouplevel:D	398, 399	\tl_if_empty:nTF	64, 209, 223
\tex_hsize:D	33, 44	\tl_if_exist:NTF	309, 357, 429, 510
\tex_lastskip:D	20	\tl_if_novalue:nTF	155
\tex_setbox:D	31, 42	\tl_map_inline:nn	350
\tex_unskip:D	29	\tl_new:N	3, 4, 49, 63, 66, 67, 68, 69, 70, 71, 75, 103, 104, 151, 153, 167, 236, 311, 408
\tex_vbox:D	31, 42		
\tex_vrule:D	50		
\texorpdfstring	173, 189		
text commands:			
\text_purify:n	58, 206		

<code>\tl_put_right:N</code>	77	<code>\UseInstance</code> .	101, 134, 144, 191, 278, 330, 338, 387, 392, 402, 405, 497, 498
<code>\tl_retokenize:n</code>	78	<code>\UseStructureName</code>	128, 143, 213
<code>\tl_set:Nn</code>	13, 76, 115, 115, 116, 116, 117, 117, 121, 122, 123, 124, 147, 152, 176, 396	<code>\UseTaggingSocket</code> .	137, 138, 156, 241, 243
<code>\tl_set_eq:NN</code>	45, 154, 181	<code>\UseTemplate</code>	142, 145
<code>\tl_to_str:n</code>	60, 61, 62, 74, 89, 109, 114, 115, 124, 484	V	
<code>\tl_trim_spaces:n</code>	55	<code>\value</code> ...	197, 276, 291, 292, 296, 297, 308
<code>\tl_use:N</code>	66, 122, 227	vbox commands:	
<code>\today</code>	3	<code>\vbox:n</code>	55, 58
token commands:		<code>\vbox_gset:Nn</code>	183
<code>\token_if_eq_meaning:NNTF</code> .	176, 187	<code>\vbox_set:Nn</code>	59
<code>\token_to_str:N</code>	81, 82	<code>\vbox_set:Nw</code>	70, 74, 94, 355
<code>\topsep</code>	52, 60, 67	<code>\vbox_set_end:</code>	
U		81, 86, 97, 98, 268, 285, 367
<code>\uncover</code>	105	<code>\vbox_set_to_wd:Nnn</code>	29, 318
<code>\uncoverenv (env.)</code>	114	<code>\vbox_set_to_wd:Nnw</code>	38, 85, 261
<code>\unskip</code>	24	<code>\vbox_to_ht:nn</code>	60, 101, 269, 295
use commands:		<code>\vbox_top:n</code>	74
<code>\use:N</code>		<code>\vbox_unpack:N</code>	185
. 98, 103, 137, 140, 144, 198, 213, 521		<code>\vbox_unpack_drop:N</code>	99, 104, 108
<code>\use:n</code>	48, 62, 75, 78, 111, 112, 118, 123, 126, 277, 390, 523	vcoffin commands:	
<code>\use_ii:nn</code>	226	<code>\vcoffin_set:Nnn</code>	2
<code>\use_none:n</code>	188, 205	<code>verse (env.)</code>	330
<code>\usebox</code>	441	<code>\vfil</code>	272, 299, 326
<code>\UseHookWithArguments</code>		<code>\visible</code>	105
..... 313, 316, 318, 321, 410		<code>visibleenv (env.)</code>	114
		<code>\vspace</code>	32, 41, 66, 76
		Y	
		<code>\year</code>	22