

ltx-talk – A class for typesetting presentations*

Joseph Wright[†]

Released 2025-07-12

Contents

I	ltx-talk – Overall set up	1
1	ltx-talk implementation	1
1.1	Set up	1
1.2	Additions for expl3	1
1.3	Extra variants	2
1.4	Scratch space	2
1.5	Messages	2
1.6	Option handling	2
1.7	Setting up	3
1.8	Font selection	4
1.9	Hyperlinks	4
1.10	Tagging	5
II	ltx-talk-color – Color definitions	6
1	ltx-talk-color implementation	6
1.1	Existing definitions	6
1.2	Document commands	6
1.3	Color definition	7
1.4	Semantic colors	7
III	ltx-talk-decode – Decoding overlay specs	8
1	ltx-talk-decode implementation	8
IV	ltx-talk-frame – The structure of frames	15

*This file describes v0.1.0, last revised 2025-07-12.

[†]E-mail: joseph@texdev.net

1	ltx-talk-frame implementation	15
1.1	Slides in frames	15
1.2	Counters	18
1.3	Frame options	19
1.4	Tagging for headers	19
1.5	Wallpaper	20
1.6	The <code>frame</code> environment	24
V	ltx-talk-frame – The structure of frames	27
1	ltx-talk-frame-structure implementation	27
VI	ltx-talk-mode – Modes	30
1	ltx-talk-mode implementation	30
VII	ltx-talk-overlay – Overlays	31
1	ltx-talk-overlay implementation	31
1.1	Utilities	31
1.2	Action commands and environments	31
1.3	Non-action commands and environments	35
1.4	Fixed-size areas	36
1.5	Adding overlays to existing commands	38
VIII	ltx-talk-required – “Required” definitions	41
1	ltx-talk-required implementation	41
1.1	Standard design settings	41
1.2	List support	42
IX	ltx-talk-structure – Structural commands	43
1	ltx-talk-structure implementation	43
1.1	Frame title	43
1.2	Sectioning	44
1.3	Table of contents	46
1.4	Block environments	48
1.5	Lists	49
1.6	Theorems, <i>etc.</i>	52
X	ltx-talk-title – Title pages	53
1	ltx-talk-title implementation	53

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} {2025-05-04} {0.0.0}
4 {A class for typesetting presentations}
    Require the latest LATEX structures.
5 \NeedsDocumentMetadata
```

1.2 Additions for expl3

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

```
6 \cs_gset_protected:Npn \vbox_set_to_wd:Nnn #1#2#3
7 {
8     \tex_setbox:D #1 \tex_vbox:D
9     {
10         \tex_hsize:D \_box_dim_eval:n {#2}
11         \color_group_begin: #3 \par \color_group_end:
12     }
13     \box_dp:N #1 \_box_dim_eval:n {#2}
14 }
15 \cs_gset_protected:Npn \vbox_set_to_wd:Nnw #1#2
16 {
17     \cs_set_protected:Npn \_box_set_to_wd:
18     { \box_wd:N #1 \_box_dim_eval:n {#2} }
19     \tex_setbox:D #1 \tex_vbox:D
20     \c_group_begin_token
21     \tex_hsize:D \_box_dim_eval:n {#2}
22     \group_insert_after:N \_box_set_to_wd:
23     \color_group_begin:
24 }
    Some things from xbox that would be useful.
25 \cs_gset_protected:Npn \rule:nnn #1#2#3
26 {
27     \tex_vrule:D
28     height \dim_eval:n {#2} \exp_stop_f:
29     depth \dim_eval:n {#3} \exp_stop_f:
30     width \dim_eval:n {#1} \exp_stop_f:
31     \scan_stop:
32 }
```

1.3 Extra variants

```

33 \cs_generate_variant:Nn \clist_set:Nn { cv }
34 \cs_generate_variant:Nn \hook_gput_code:nnn { nne }
35 \exp_args_generate:n { nVv }
36 \cs_generate_variant:Nn \color_select:n { V }
37 \cs_generate_variant:Nn \dim_compare:nNnTF { v }
38 \cs_generate_variant:Nn \dim_compare_p:nNn { vNv }
39 \cs_generate_variant:Nn \dim_max:nn { v }
40 \cs_generate_variant:Nn \text_purify:n { v }
41 \cs_generate_variant:Nn \vbox_to_ht:nn { v }

```

1.4 Scratch space

```

\__talk_tmp:w For one-off processing.
42 \cs_new_protected:Npn \__talk_tmp:w { }

```

(End of definition for __talk_tmp:w.)

```

\l__talk_tmp_box
43 \box_new:N \l__talk_tmp_box

```

(End of definition for \l__talk_tmp_box.)

```

\l__talk_tmp_tl
44 \tl_new:N \l__talk_tmp_tl

```

(End of definition for \l__talk_tmp_tl.)

1.5 Messages

Get the right type of message.

```

45 \prop_gput:Nnn \g_msg_module_name_prop { talk } { ltx-talk }
46 \prop_gput:Nnn \g_msg_module_type_prop { talk } { Class }

```

1.6 Option handling

```

\l__talk_aspect_ratio_str
\l__talk_fontsize_dim
\l__talk_frame_title_bool
\l__talk_mode_str
47 \keys_define:nn { talk }
48 {
49   aspect-ratio .str_set:N =
50     \l__talk_aspect_ratio_str ,
51   font-size .dim_set:N =
52     \l__talk_fontsize_dim ,
53   frame-title-arg .bool_set:N =
54     \l__talk_frame_title_bool ,
55   mode .choices:nn =
56     { handout , projector }
57     { \str_set:NV \l__talk_mode_str \l_keys_choice_tl }
58 }

```

(End of definition for \l__talk_aspect_ratio_str and others.)

Scope for options.

```
59 \keys_define:nn { talk }
60 {
61   aspect-ratio .usage:n = load ,
62   font-size    .usage:n = load ,
63   frame-title-arg .usage:n = load ,
64   mode         .usage:n = load
65 }
```

Initial values.

```
66 \keys_set:nn { talk }
67 {
68   aspect-ratio = 16:9 ,
69   font-size    = 11pt ,
70   frame-title-arg = false ,
71   mode         = projector
72 }
```

```
73 \ProcessKeyOptions [ talk ]
```

1.7 Setting up

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

```
74 \file_if_exist_input:nF { size \dim_to_decimal:n \l__talk_fontsize_dim .clo }
75 {
76   \file_input:n { size10.clo }
77   \RequirePackage { relsize }
78   \hook_gput_code:nne { begindocument } { talk }
79   { \exp_not:N \relsize { \fp_eval:n { \l__talk_fontsize_dim / 10pt } } }
80 }
```

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

```
81 \dim_const:Nn \c__talk_paper_height_dim { 100mm }
82 \use:e
83 {
84   \cs_set_protected:Npn \exp_not:N \__talk_tmp:w
85     #1 \tl_to_str:n { : } #2 \tl_to_str:n { : } #3 \exp_not:N \q_stop
86     {
87       \dim_const:Nn \exp_not:N \c__talk_paper_width_dim
88       {
89         \exp_not:N \fp_to_dim:n
90         { (#1 / #2) * \exp_not:N \c__talk_paper_height_dim }
91       }
92     }
93   \exp_not:N \__talk_tmp:w \l__talk_aspect_ratio_str
94   \tl_to_str:n { : } 100 \exp_not:N \q_stop
95 }
96 \use:e
97 {
98   \exp_not:N \RequirePackage
99   [
100     papersize =
```

```

101     {
102         \dim_use:N \c__talk_paper_width_dim ,
103         \dim_use:N \c__talk_paper_height_dim
104     } ,
105     tmargin    = 10mm ,
106     bmargin    = 8mm ,
107     lmargin    = 10mm ,
108     rmargin    = 10mm ,
109     headheight = 10mm ,
110     headsep     = 2mm ,
111     footskip    = 6mm
112 ]
113 { geometry }
114 }

```

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

Turn off justification

```

115 \raggedright

```

1.8 Font selection

The aim here is to minimize change from the standard font setup but at the same time provide a sanserif default. Since `beamer` was released, better sanserif math mode fonts have become available. For OpenType engines, requiring `unicode-math` is the most sensible approach. 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 sanserif is easy. For math mode, the `sansmathfonts` package does a good job: here, using the package rather than adjusting directly is the sensible option.

```

116 \sys_if_engine_opentype:TF
117 {
118     \RequirePackage { unicode-math }
119     \setsansfont { NewCMSans10-Regular.otf }
120     \setmathfont { NewCMSansMath-Regular.otf }
121 }
122 { \RequirePackage { sansmathfonts } }
123 \cs_set_eq:NN \rmdefault \sfdefault

```

1.9 Hyperlinks

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

```

124 \cs_new:Npn \thepage { \@arabic \c@page }

```

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

A requirement.

```

125 \RequirePackage { hyperref }
126 \hypersetup { hidelinks }

```

1.10 Tagging

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

```
127 \tagpdfsetup
128 {
129   role / user-NS = ltx-talk      ,
130   role / new-tag = frame / Sect  ,
131   role / new-tag = frametitle / H4
132 }
133 \</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 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 }
20 \RenewDocumentCommand \mathcolor { D <> { all } o m +m }
21 {
22   \__talk_if_overlay:nT {#1}
```

```

23     {
24         \IfNoValueTF {#2}
25         { \color_math:en {#3} {#4} }
26         { \color_math:nen {#2} {#3} {#4} }
27     }
28 }
29 \RenewDocumentCommand \textcolor { D <> { all } o m +m }
30 {
31     \__talk_if_overlay:nT {#1}
32     {
33         \mode_leave_vertical:
34         \group_begin:
35         \IfNoValueTF {#2}
36         { \color_select:e {#3} }
37         { \color_select:ne {#2} {#3} }
38         #4
39         \group_end:
40     }
41 }

```

(End of definition for `\color`, `\mathcolor`, and `\textcolor`. 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.

```

42 \NewDocumentCommand \DeclareColor { m o m }
43 {
44     \IfNoValueTF {#2}
45     { \colorlet {#1} {#3} }
46     { \definecolor {#1} {#2} {#3} }
47 }

```

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

1.4 Semantic colors

Pick up the standard colors from beamer.

```

48 \DeclareColor { alert } [ RGB ] { 200 , 0 , 0 }
49 \DeclareColor { example } { green!50!black }
50 \DeclareColor { structure } [ rgb ] { 0.2 , 0.2 , 0.7 }
51 </class>

```

Part III

ltx-talk-decode – Decoding overlay specs

1 ltx-talk-decode implementation

Start the DocStrip guards.

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

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

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

(End of definition for \l__talk_decode_overlays_bool.)

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

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

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

`\l__talk_decode_pure_bool` Tracks whether only mode specifications were given.

```
7 \bool_new:N \l__talk_decode_pure_bool
```

(End of definition for \l__talk_decode_pure_bool.)

`\l__talk_decode_step_bool` Tracks whether to step `\g__talk_pauses_int`.

```
8 \bool_new:N \l__talk_decode_step_bool
```

(End of definition for \l__talk_decode_step_bool.)

`\l__talk_decode_arg_str` For error usage.

```
9 \str_new:N \l__talk_decode_arg_str
```

(End of definition for \l__talk_decode_arg_str.)

`\l__talk_decode_overlays_clist` The decoded overlay specification: will have only absolute slide numbers present, potentially along with ranges.

`\l__talk_decode_overlays_str`

```
10 \clist_new:N \l__talk_decode_overlays_clist
11 \str_new:N \l__talk_decode_overlays_str
```

(End of definition for \l__talk_decode_overlays_clist and \l__talk_decode_overlays_str.)

`\l__talk_decode_action_str` The action which is active, if any.

```
12 \str_new:N \l__talk_decode_action_str
```

(End of definition for \l__talk_decode_action_str.)

`\l__talk_decode_actions_bool` For the actions versions of overlay tracking.

`\l__talk_decode_actions_clist` 13 `\bool_new:N \l__talk_decode_actions_bool`

`\l__talk_decode_actions_str` 14 `\clist_new:N \l__talk_decode_actions_clist`

15 `\str_new:N \l__talk_decode_actions_str`

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

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

`__talk_decode_parse_aux:n` blocks divided by `|` tokens.

`__talk_decode_parse:w`

```

16 \cs_new_protected:Npn \__talk_decode_parse:n #1
17 {
18   \str_clear:N \l__talk_decode_action_str
19   \bool_lazy_or:nnTF
20     { \tl_if_blank_p:n {#1} }
21     { \str_if_eq_p:nn {#1} { all } }
22     { \bool_set_true:N \l__talk_decode_overlays_bool }
23     {
24       \str_set:Nn \l__talk_decode_arg_str {#1}
25       \bool_set_false:N \l__talk_decode_actions_bool
26       \bool_set_false:N \l__talk_decode_overlays_bool
27       \bool_set_true:N \l__talk_decode_pure_bool
28       \str_clear:N \l__talk_decode_overlays_str
29       \str_clear:N \l__talk_decode_actions_str
30       \exp_args:No \__talk_decode_parse_aux:n { \l__talk_decode_arg_str }
31     }
32 }
33 \cs_new_protected:Npn \__talk_decode_parse_aux:n #1
34 { \__talk_decode_parse:w #1 | \q_recursion_tail | \q_recursion_stop }

```

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

```

35 \cs_new_protected:Npn \__talk_decode_parse:w #1 |
36 {
37   \quark_if_recursion_tail_stop_do:nn {#1}
38   {
39     \bool_lazy_and:nnT
40       { \str_if_empty_p:N \l__talk_decode_overlays_str }
41       { ! \l__talk_decode_pure_bool }
42       { \bool_set_true:N \l__talk_decode_overlays_bool }
43   }
44   \exp_args:Ne \__talk_decode_mode:n
45   { \tl_trim_spaces:n {#1} }
46   \__talk_decode_parse:w
47 }

```

(End of definition for `__talk_decode_parse:n`, `__talk_decode_parse_aux:n`, and `__talk_decode_parse:w`.)

`\c__talk_modes_clist` The possible modes: detokenized as that is applied up-front in decoding.

```

48 \clist_const:Ne \c__talk_modes_clist
49 {
50   \tl_to_str:n { handout } ,
51   \tl_to_str:n { projector }
52 }

```

(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 is filled in with a *.

```

\__talk_decode_mode:w
\__talk_decode_mode_aux:n
53 \cs_new_protected:Npe \__talk_decode_mode:n #1
54 {
55   \clist_if_in:NnTF \exp_not:N \c__talk_modes_clist {#1}
56   {
57     \exp_not:N \str_if_eq:VnT
58     \exp_not:N \l__talk_mode_str {#1}
59     { \bool_set_true:N \exp_not:N \l__talk_decode_overlays_bool }
60   }
61   {
62     \exp_not:N \__talk_decode_mode:w #1 \tl_to_str:n { : : }
63     \exp_not:N \q_stop
64   }
65 }
66 \use:e
67 {
68   \cs_new_protected:Npe \exp_not:N \__talk_decode_mode:w
69   #1 \token_to_str:N :
70   #2 \token_to_str:N :
71   #3 \exp_not:N \q_stop
72 }
73 {
74   \exp_not:N \tl_if_blank:nTF {#2}
75   {
76     \exp_not:N \__talk_decode_mode:nn
77     { \tl_to_str:n { projector } } {#1}
78   }
79   { \exp_not:N \__talk_decode_mode:nn {#1} {#2} }
80 }
81 \cs_new_protected:Npn \__talk_decode_mode:nn #1#2
82 {
83   \str_if_eq:VnTF \l__talk_mode_str {#1}
84   {
85     \__talk_decode_action:n {#2}
86     \str_if_empty:NT \l__talk_decode_overlays_str
87     { \__talk_decode_overlays:nn { overlays } { * } }
88   }
89   {
90     \tl_if_blank:nF {#2}
91     { \bool_set_false:N \l__talk_decode_pure_bool }
92   }
93 }

```

(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 or from the known list. If we don't find one of those outcomes, we can issue an error.

```

\__talk_decode_action:w
94 \cs_new_protected:Npe \__talk_decode_action:n #1
95 {
96   \exp_not:N \__talk_decode_action:w

```

```

97     #1 \tl_to_str:n { @ @ } \exp_not:N \q_stop
98   }
99   \use:e
100   {
101     \cs_new_protected:Npn \exp_not:N \__talk_decode_action:w
102     #1 \tl_to_str:n { @ } #2 \tl_to_str:n { @ } #3 \exp_not:N \q_stop
103   }
104   {
105     \tl_if_blank:nTF {#2}
106     { \__talk_decode_overlays:nn { overlays } {#1} }
107     {
108       \cs_if_exist:cTF { __talk_action_ #1 :N }
109       {
110         \bool_set_false:N \l__talk_decode_pure_bool
111         \str_set:Nn \l__talk_decode_action_str {#1}
112         \tl_if_blank:nF {#2}
113         { \__talk_decode_overlays:nn { actions } {#2} }
114       }
115       {
116         \msg_error:nnV { talk } { bad-action-spec }
117         \l__talk_decode_arg_str
118       }
119     }
120   }

```

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

```

\__talk_decode_overlays:nn
\__talk_decode_overlays:nN
  \@_decode_overlay_+:nw
\__talk_decode_overlay_.:nw
  \__talk_decode_overlay_aux:nNN
  \__talk_decode_overlay_offset:nNnN
  \__talk_decode_overlay_offset:nNn

```

The loop here needs to replace all + and . characters by the current automatic value, allowing for any offsets. This step also needs to track whether to increment the automatic value: true if a + is seen, false otherwise.

```

121 \cs_new_protected:Npn \__talk_decode_overlays:nn #1#2
122 {
123   \bool_set_false:N \l__talk_decode_step_bool
124   \__talk_decode_overlays:nN {#1} #2 \q_recursion_tail \q_recursion_stop
125   \bool_if:NT \l__talk_decode_step_bool
126   { \int_gincr:N \g__talk_pauses_int }
127   \__talk_decode_check:n {#1}
128 }
129 \cs_new_protected:Npn \__talk_decode_overlays:nN #1#2
130 {
131   \quark_if_recursion_tail_stop:N #2
132   \cs_if_exist_use:cF { __talk_decode_overlay_ #2 :nw }
133   {
134     \str_put_right:cn { l__talk_decode_ #1 _str } {#2}
135     \__talk_decode_overlays:nN
136   }
137   {#1}
138 }
139 \cs_new_protected:cpn { __talk_decode_overlay_+:nw } #1
140 {
141   \bool_set_true:N \l__talk_decode_step_bool
142   \__talk_decode_overlay_aux:nNN {#1} 0
143 }
144 \cs_new_protected:cpn { __talk_decode_overlay_.:nw } #1

```

```
145 { \__talk_decode_overlay_aux:nNN {#1} 1 }
```

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 (.

```
146 \cs_new_protected:Npn \__talk_decode_overlay_aux:nNN #1#2#3
147 {
148   \quark_if_recursion_tail_stop_do:Nn #3
149   {
150     \__talk_decode_overlay_offset:nNn {#1} #2 { 0 }
151     \q_recursion_tail \q_recursion_stop
152   }
153   \token_if_eq_meaning:NNTF #3 ( % )
154   { \__talk_decode_overlay_offset:nNn {#1} #2 { } }
155   { \__talk_decode_overlay_offset:nNn {#1} #2 { 0 } #3 }
156 }
```

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.

```
157 \cs_new_protected:Npn \__talk_decode_overlay_offset:nNnN #1#2#3#4
158 {
159   \quark_if_recursion_tail_stop_do:Nn #4
160   {
161     \msg_error:nnV { talk } { bad-action-spec }
162     \l__talk_decode_arg_str
163   } % (
164   \token_if_eq_meaning:NNTF #4 )
165   { \__talk_decode_overlay_offset:nNn {#1} #2 {#3} }
166   { \__talk_decode_overlay_offset:nNnN {#1} #2 {#3#4} }
167 }
```

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

```
168 \cs_new_protected:Npn \__talk_decode_overlay_offset:nNn #1#2#3
169 {
170   \str_put_right:ce { l__talk_decode_ #1 _str }
171   { \int_max:nn { 0 } { #3 + \g__talk_pauses_int - #2 } }
172   \__talk_decode_overlays:nN {#1}
173 }
```

(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.

```
174 \cs_new_protected:Npn \__talk_decode_check:n #1
175 {
176   \clist_set:cv { l__talk_decode_ #1 _clist } { l__talk_decode_ #1 _str }
177   \clist_if_in:cnTF { l__talk_decode_ #1 _clist } { * }
178   { \bool_set_true:c { l__talk_decode_ #1 _bool } }
179   {
```

```

180     \clist_map_inline:cn { l__talk_decode_ #1 _clist }
181     { \__talk_decode_check:nw {#1} 0 ##1 - - \q_stop }
182   }
183 }

```

If #3 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 starting value in all cases, but there may not be an end one.

```

184 \cs_new_protected:Npn \__talk_decode_check:nw #1#2 - #3 - #4 \q_stop
185 {
186   \tl_if_blank:nTF {#4}
187   { \__talk_decode_check_single:nn {#1} {#2} }
188   {
189     \tl_if_blank:nTF {#3}
190     { \__talk_decode_check_range:nnn {#1} {#2} { \c_max_int } }
191     { \__talk_decode_check_range:nnn {#1} {#2} {#3} }
192   }
193 }
194 \cs_set_protected:Npn \__talk_decode_check_single:nn #1#2
195 {
196   \int_compare:nNnTF \g__talk_slide_int = {#2}
197   {
198     \bool_set_true:c { l__talk_decode_ #1 _bool }
199     \clist_map_break:
200   }
201   {
202     \int_compare:nNnT {#2} > \g__talk_slide_int
203     { \bool_gset_true:N \g__talk_slide_continue_bool }
204   }
205 }

```

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.

```

206 \cs_set_protected:Npn \__talk_decode_check_range:nnn #1#2#3
207 {
208   \int_compare:nNnF \g__talk_slide_int > {#3}
209   {
210     \int_compare:nNnTF \g__talk_slide_int < {#2}
211     { \bool_gset_true:N \g__talk_slide_continue_bool }
212     {
213       \bool_set_true:c { l__talk_decode_ #1 _bool }
214       \bool_lazy_and:nnT
215       { \int_compare_p:nNn \g__talk_slide_int < {#3} }
216       { \int_compare_p:nNn {#3} < \c_max_int }
217       { \bool_gset_true:N \g__talk_slide_continue_bool }
218       \clist_map_break:
219     }
220   }
221 }

```

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

```

222 \msg_new:nnnn { talk } { bad-action-spec }
223 { Bad-overlay-specification~"#1". }
224 {
225   The~overlay~specification~given~doesn't~follow~the~pattern~described~in~

```



```
226     the~ltx-talk~manual:~it~has~been~ignored.  
227 }  
228 </class>
```

Part IV

ltx-talk-frame – The structure of frames

1 ltx-talk-frame implementation

Start the DocStrip guards.

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

1.1 Slides in frames

Currently, each slide in a frame will produce a separate page in the output (unless the slide is suppressed entirely). Material is then hidden on some pages by using opacity. An alternative approach would be to use Optional Content Groups to have a similar effect on one page per frame. However, whilst that would be relatively clear for appear/disappear effects, it would be much less straight-forward for partial transparency, *etc.*, plus would depend more heavily on viewer support. At a future stage we may wish to revisit this.

`\g__talk_slide_continue_bool` Tracks whether the frame continues after the current slide.

```
3 \bool_new:N \g__talk_slide_continue_bool
```

(End of definition for `\g__talk_slide_continue_bool`.)

`\l__talk_slide_box`

```
4 \box_new:N \l__talk_slide_box
```

(End of definition for `\l__talk_slide_box`.)

`\g__talk_slide_int`

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

`\c@slide`

We also provide L^AT_EX counter-style access.

`\theslide`

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

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

`__talk_slide: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.

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

```

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

```

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

The very last frame will not be recorded by the above, so we have to add to the hook

at the very end of the run.

```

62 \AddToHook { enddocument / afterlastpage }
63 {
64   \property_record:ee { frame . \int_use:N \g__talk_frame_int }
65   { slides }
66 }

```

`\c__talk_pause_init_int` A simple concept: mainly done for performance reasons.

```

67 \cs_new_eq:NN \c__talk_pause_init_int \c_one_int

```

(End of definition for \c__talk_pause_init_int.)

`\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).

```

68 \int_new:N \g__talk_frame_struct_int

```

(End of definition for \g__talk_frame_struct_int.)

`__talk_slide_begin:`
`__talk_slide_end:`

```

69 \cs_new_protected:Npn \__talk_slide_begin:
70 {
71   \int_gset_eq:NN \g__talk_pauses_int \c__talk_pause_init_int
72   \bool_gset_false:N \g__talk_slide_continue_bool
73   \tl_gclear:N \g__talk_frame_title_tl
74   \tl_gclear:N \g__talk_frame_subtitle_tl
75   \__talk_cnt_save:
76   \vbox_set:Nw \l__talk_slide_box
77   \tl_gclear:N \g__talk_onslide_tl
78 }
79 \cs_new_protected:Npn \__talk_slide_end:
80 {
81   \tl_use:N \g__talk_onslide_tl
82   \vbox_set_end:
83   \bool_if:NT \g__talk_slide_continue_bool
84   { \__talk_cnt_restore: }
85   \vbox_to_ht:nn { \textheight }
86   {
87     \use:c { __talk_slide_align_ \l__talk_frame_alignment_tl :n }
88     { \vbox_unpack_drop:N \l__talk_slide_box }
89   }
90   \clearpage
91 }

```

(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
92 \cs_new_protected:Npn \__talk_slide_align_bottom:n #1
93 {
94   \skip_vertical:n { Opt~plus~1fil }
95   #1
96   \skip_vertical:n { Opt }
97 }
98 \cs_new_protected:Npn \__talk_slide_align_center:n #1
99 {

```

```

100   \skip_vertical:n { Opt~plus~0.5fil }
101   #1
102   \skip_vertical:n { Opt~plus~0.5fil }
103 }
104 \cs_new_protected:Npn \__talk_slide_align_stretch:n #1
105 {
106   \skip_vertical:n { Opt }
107   #1
108   \skip_vertical:n { Opt }
109 }
110 \cs_new_protected:Npn \__talk_slide_align_top:n #1
111 {
112   \skip_vertical:n { Opt }
113   #1
114   \skip_vertical:n { Opt~plus~1fil }
115 }

```

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

1.2 Counters

\l__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.

```

116 \seq_new:N \l__talk_cnt_reset_seq
117 \seq_set_from_clist:Nn \l__talk_cnt_reset_seq
118 {
119   equation      ,
120   footnote      ,
121   mpfootnote    ,
122   parentequation
123 }
124 \seq_map_inline:Nn \l__talk_cnt_reset_seq
125 {
126   \int_new:c { g__talk_saved_ #1 _int }
127   \int_gset_eq:cc { g__talk_saved_ #1 _int } { c@ #1 }
128 }

```

(End of definition for \l__talk_cnt_reset_seq.)

__talk_cnt_save: A simple save-and-restore pair.

__talk_cnt_restore:

```

129 \cs_new_protected:Npn \__talk_cnt_save:
130 {
131   \seq_map_inline:Nn \l__talk_cnt_reset_seq
132   { \int_gset_eq:cc { g__talk_saved_ ##1 _int } { c@ ##1 } }
133 }
134 \cs_new_protected:Npn \__talk_cnt_restore:
135 {
136   \seq_map_inline:Nn \l__talk_cnt_reset_seq
137   { \int_gset_eq:cc { c@ ##1 } { g__talk_saved_ ##1 _int } }
138 }

```

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

```

\@definecounter Track all counters for resetting.
\std@definecounter
139 \cs_new_eq:NN \std@definecounter \@definecounter
140 \cs_gset_protected:Npn \@definecounter #1
141 {
142   \std@definecounter {#1}
143   \int_new:c { g__talk_saved_ #1 _int }
144   \seq_gput_right:Nn \l__talk_cnt_reset_seq {#1}
145 }

```

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

1.3 Frame options

```

\l__talk_frame_alignment_tl
146 \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_tagging_str
147 \keys_define:nn { talk / frame }
148 {
149   action-spec .str_set:N
150     = \l__talk_action_spec_str ,
151   tag-slides .str_set:N
152     = \l__talk_frame_tagging_str ,
153   vertical-alignment .choices:nn =
154     { bottom , center , stretch , top }
155     {
156       \tl_set_eq:NN \l__talk_frame_alignment_tl
157       \l_keys_value_tl
158     }
159 }
160 \keys_set:nn { talk / frame }
161 {
162   action-spec = ,
163   tag-slides = n ,
164   vertical-alignment = center
165 }

(End of definition for \l__talk_action_spec_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
\__talk_header_tag_begin:e of tagging).
\__talk_header_tag_end:
166 \cs_new_protected:Npn \__talk_header_tag_begin:n #1
167 {
168   \tag_resume:n { header }
169   \tag_mc_end:
170   \tag_struct_begin:n {#1}
171   \tag_mc_begin:n { }
172 }
173 \cs_generate_variant:Nn \__talk_header_tag_begin:n { e }

```

```

174 \cs_new_protected:Npn \__talk_header_tag_end:
175 {
176   \tag_mc_end:
177   \tag_struct_end:
178   \tag_mc_begin:n { artifact }
179   \tag_suspend:n { header }
180 }

```

(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
181 \NewTemplateType { footer-element } { 1 }
182 \DeclareTemplateInterface { footer-element } { talk } { 1 }
183 {
184   color      : tokenlist ,
185   font       : tokenlist = ,
186   left-skip  : length = 0em ,
187   right-skip : length = 0em
188 }
189 \DeclareTemplateCode { footer-element } { talk } { 1 }
190 {
191   color      = \l__talk_footelem_color_tl ,
192   font       = \l__talk_footelem_font_tl ,
193   left-skip  = \l__talk_footelem_left_skip ,
194   right-skip = \l__talk_footelem_right_skip
195 }
196 {
197   \tl_if_empty:nF {#1}
198   {
199     \hspace { \l__talk_footelem_left_skip }
200     \group_begin:
201       \tl_if_empty:NF \l__talk_footelem_color_tl
202       { \color_select:V \l__talk_footelem_color_tl }
203       \l__talk_footelem_font_tl
204       #1
205     \group_end:
206     \hspace { \l__talk_footelem_right_skip }
207   }
208 }
209 \DeclareInstance { footer-element } { date } { talk } { }
210 \DeclareInstance { footer-element } { author } { talk } { }
211 \DeclareInstance { footer-element } { title } { talk } { }
212 \DeclareInstance { footer-element } { institute } { talk } { }
213 \DeclareInstance { footer-element } { framenumbers } { talk } { }

```

(End of definition for \l__talk_footelem_left_skip and others.)

```

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

```

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

```

214 \NewTemplateType { header } { 0 }

```

```

215 \DeclareTemplateInterface { header } { talk } { 0 }
216 {
217     background-color : tokenlist,
218     color             : tokenlist = structure ,
219     font              : tokenlist = \normalfont ,
220     height            : length = \Gm@tmargin + \headsep ,
221     left-hspace       : skip = \Gm@lmargin ,
222     print-frame-title : boolean = true ,
223     right-hspace      : skip = \Gm@rmargin
224 }
225 \DeclareTemplateCode { header } { talk } { 0 }
226 {
227     background-color = \l__talk_header_bg_tl ,
228     color            = \l__talk_header_fg_tl ,
229     font             = \l__talk_header_font_tl ,
230     height           = \l__talk_header_ht_dim ,
231     left-hspace      = \l__talk_header_left_skip ,
232     print-frame-title = \l__talk_header_frametitle_bool ,
233     right-hspace     = \l__talk_header_right_skip
234 }
235 {
236     \noindent
237     \__talk_wallpaper_hruler:Nnn
238     \l__talk_header_bg_tl
239     { \l__talk_header_ht_dim - \headsep }
240     { \headsep }
241     \skip_horizontal:n { \l__talk_header_left_skip }
242     \group_begin:
243     \tl_if_empty:NF \l__talk_header_fg_tl
244     { \color_select:V \l__talk_header_fg_tl }
245     \l__talk_header_font_tl
246     \bool_if:NT \l__talk_header_frametitle_bool
247     {
248         \ExpandArgs { nnV }
249         \UseInstance { frametitle } { header }
250         \g__talk_frame_title_tl
251     }
252     \group_end:
253 }
254 \DeclareInstance { header } { std } { talk } { }
255 \AddToHook { begindocument }
256 {
257     \DeclareInstanceCopy { header } { wallpaper } { std }
258     \EditInstance { header } { wallpaper }
259     { print-frame-title = false }
260 }

```

(End of definition for \l__talk_header_bg_tl and others.)

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

```

261 \NewTemplateType { footer } { 0 }
262 \DeclareTemplateInterface { footer } { talk } { 0 }

```

\l__talk_footer_sep_tl
\l__talk_footer_left_skip
\l__talk_footer_right_skip


```

263 {
264     background-color : tokenlist ,
265     color             : tokenlist ,
266     element-order     : commalist ,
267     font              : tokenlist = \tiny ,
268     left-skip         : length = \Gm@lmargin ,
269     right-skip        : length = \Gm@rmargin ,
270     separator         : tokenlist = \hfil
271 }
272 \DeclareTemplateCode { footer } { talk } { 0 }
273 {
274     background-color = \l__talk_footer_bg_tl ,
275     color            = \l__talk_footer_fg_tl ,
276     element-order    = \l__talk_footer_order_clist ,
277     separator        = \l__talk_footer_sep_tl ,
278     font             = \l__talk_footer_font_tl ,
279     left-skip        = \l__talk_footer_left_skip ,
280     right-skip       = \l__talk_footer_right_skip
281 }
282 {
283     \noindent
284     \__talk_wallpaper_hrule:Nnn
285     \l__talk_footer_bg_tl
286     { \footskip }
287     { \Gm@bmargin - \footskip }
288     \skip_horizontal:n { \l__talk_footer_left_skip }
289     \vbox_set_to_wd:Nnn \l__talk_tmp_box
290     {
291         \paperwidth
292         - \l__talk_footer_left_skip
293         - \l__talk_footer_right_skip
294     }
295     {
296         \tl_if_empty:NF \l__talk_footer_fg_tl
297         { \color_select:V \l__talk_footer_fg_tl }
298         \l__talk_footer_font_tl
299         \clist_pop:NNT \l__talk_footer_order_clist \l__talk_tmp_tl
300         {
301             \ExpandArgs { nVv } \UseInstance { footer-element } \l__talk_tmp_tl
302             { @ \l__talk_tmp_tl }
303             \clist_map_inline:Nn \l__talk_footer_order_clist
304             {
305                 \l__talk_footer_sep_tl
306                 \ExpandArgs { nnv }
307                 \UseInstance { footer-element } {##1} { @ ##1 }
308             }
309         }
310         \hfil
311     }
312     \box_use_drop:N \l__talk_tmp_box
313     \skip_horizontal:n { \l__talk_footer_right_skip - \Gm@rmargin }
314 }
315 \DeclareInstance { footer } { std } { talk } { }
316 \AddToHook { begindocument }

```

```

317 {
318   \DeclareInstanceCopy { footer } { wallpaper } { std }
319   \EditInstance { footer } { wallpaper }
320     { element-order = }
321 }

```

(End of definition for \l__talk_footer_bg_tl and others.)

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

```

322 \cs_new_protected:Npn \__talk_wallpaper_hrule:Nnn #1#2#3
323 {
324   \skip_horizontal:n { -\Gm@lmargin }
325   \tl_if_empty:NF #1
326   {
327     \group_begin:
328       \color_select:V #1
329       \rule:nnn { \paperwidth } {#2} {#3}
330       \skip_horizontal:n { -\paperwidth }
331     \group_end:
332   }
333 }

```

(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.

```

334 \cs_set_nopar:Npn \ps@plain
335 {
336   \cs_set_nopar:Npn \@oddhead
337   {
338     \__talk_section_tagged:
339     \hfil
340   }
341   \cs_set_nopar:Npn \@oddfoot { }
342   \cs_set_eq:NN \@evenhead \@oddhead
343   \cs_set_eq:NN \@evenfoot \@oddfoot
344 }
345 \cs_set_nopar:Npn \ps@wallpaper
346 {
347   \cs_set_nopar:Npn \@oddhead
348   {
349     \__talk_section_tagged:
350     \UseInstance { header } { wallpaper }
351     \hfil
352   }
353   \cs_set_nopar:Npn \@oddfoot
354   {
355     \UseInstance { footer } { wallpaper }
356     \hfil
357   }
358   \cs_set_eq:NN \@evenhead \@oddhead
359   \cs_set_eq:NN \@evenfoot \@oddfoot

```

```

360 }
361 \cs_new_nopar:Npn \ps@talk
362 {
363   \cs_set_nopar:Npn \@oddhead
364   {
365     \__talk_section_tagged:
366     \UseInstance { header } { std }
367     \hfil
368   }
369   \cs_set_nopar:Npn \@oddfoot { \UseInstance { footer } { std } }
370   \cs_set_eq:NN \@evenhead \@oddhead
371   \cs_set_eq:NN \@evenfoot \@oddfoot
372 }
373 \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.

```
374 \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).

```
375 \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).

```
376 \bool_new:N \l__talk_frame_verb_bool
```

(End of definition for \l__talk_frame_verb_bool.)

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

```

\c@frame 377 \int_new:N \g__talk_frame_int
\theframe 378 \cs_new_eq:NN \c@frame \g__talk_frame_int
\@framenum 379 \cs_new:Npn \theframe { \@arabic \c@frame }
380 \cs_new:Npn \@framenum { \arabic { frame } }

```

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

The total frames can be handled using the kernel properties.

```

381 \property_new:nnnn { totalframes } { shipout } { -1 }
382 { \int_use:N \g__talk_frame_int }
383 \AddToHook { enddocument / afterlastpage }
384 { \property_record:nn { lastpage } { totalframes } }

```

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

```
385 \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.

```

386 \cs_new_protected:Npn \__talk_frame_process:nn #1#2
387 {
388   \int_gincr:N \g__talk_frame_int
389   \bool_set_true:N \l__talk_frame_bool
390   \__talk_slide:nn {#1} {#2}
391 }

```

(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.

```

392 \cs_new_protected:Npn \__talk_frame_tag:n #1
393 {
394   \tag_struct_begin:n { tag = frame }
395   \int_gset:Nn \g__talk_frame_struct_int { \tag_get:n { struct_num } }
396   \bool_gset_true:N \g__talk_frame_tag_bool
397   #1
398   \tag_struct_end:
399 }

```

(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.

```

400 \cs_new_protected:Npn \__talk_frame_notag:n #1
401 {
402   \tag_mc_begin:n { artifact }
403   \tag_suspend:n { frame }
404   \bool_gset_false:N \g__talk_frame_tag_bool
405   #1
406   \par
407   \tag_resume:n { frame }
408   \tag_mc_end:
409 }

```

(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.

```

410 \bool_if:NTF \l__talk_frame_title_bool
411 {
412   \RenewDocumentEnvironment { frame }
413     { D <> { all } = { action-spec } 0 { } +m +b }
414     {
415       \keys_set:nn { talk / frame } {#2}
416       \bool_set_false:N \l__talk_frame_verb_bool
417       \__talk_frame_process:nn {#1} { \frametitle {#3} #4 }
418     }
419   { }
420   \NewDocumentEnvironment { frame* }
421     { D <> { all } = { action-spec } 0 { } +m c }
422     {
423       \keys_set:nn { talk / frame } {#2}

```

```

424     \bool_set_true:N \l__talk_frame_verb_bool
425     \tl_gset:Nn \g__talk_frame_title_tl {#3}
426     \exp_args:Nne \__talk_frame_process:nn {#1}
427     { \tl_to_str:n { \frametitle } \exp_not:n { {#3} #4 } }
428   }
429   { }
430 }
431 {
432   \RenewDocumentEnvironment { frame }
433   { !D <> { all } = { action-spec } !0 { } +b }
434   {
435     \keys_set:nn { talk / frame } {#2}
436     \bool_set_false:N \l__talk_frame_verb_bool
437     \__talk_frame_process:nn {#1} {#3}
438   }
439   { }
440   \NewDocumentEnvironment { frame* }
441   { !D <> { all } = { action-spec } !0 { } c }
442   {
443     \keys_set:nn { talk / frame } {#2}
444     \bool_set_true:N \l__talk_frame_verb_bool
445     \__talk_frame_process:nn {#1} {#3}
446   }
447   { }
448 }

```

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

449 </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>
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.)

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

```
9 \NewDocumentEnvironment { columns } { D <> { all } 0 { } }
10 {
11   \__talk_action_begin:n {#1}
12   \par
13   \keys_set:nn { talk / columns } {#2}
14   \hbox_set_to_wd:Nnw \l__talk_tmp_box { \l__talk_columns_wd_tl }
15   \dim_set:Nn \textwidth { \l__talk_columns_wd_tl }
16   \dim_set_eq:NN \columnwidth \textwidth
17   \hfil
18   \ignorespaces
19 }
20 {
21   \unskip
22   \hfil
23   \hbox_set_end:
24   \box_use_drop:N \l__talk_tmp_box
25   \par
26   \__talk_action_end:
27 }
```

`\l__talk_column_alignment_tl`

```
28 \keys_define:nn { talk / column }
29 {
30   b .meta:n =
31     { vertical-alignment = bottom } ,
```

```

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

```

(End of definition for `\l__talk_column_alignment_tl`.)

```

\__talk_column_align_bottom:n Based on ideas in the highly experimental xbox.
\__talk_column_align_center:n
\__talk_column_align_top:n
50 \cs_new_protected:Npn \__talk_column_align_bottom:n #1
51   { \vbox:n {#1} }
52 \cs_new_protected:Npn \__talk_column_align_center:n #1
53   {
54     \vbox:n
55     {
56       \hbox:n
57       {
58         \box_move_down:nn
59         {
60           0.5 \box_ht:N \l__talk_tmp_box
61           - \tex_fontdimen:D 22 ~ \tex_textfont:D 2 ~
62         }
63         { \vbox:n {#1} }
64       }
65     }
66   }
67 \cs_new_protected:Npn \__talk_column_align_top:n #1
68   { \vbox_top:n {#1} }

```

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

column (env.) A cut-down version of a minipage: we want to be clear on the semantic meaning.

```

69 \NewDocumentEnvironment { column } { D <> { all } 0 { } m }
70 {
71   \__talk_action_begin:n {#1}
72   \par
73   \keys_set:nn { talk / column } {#2}
74   \vbox_set_to_wd:Nnw \l__talk_tmp_box {#3}
75   \dim_set:Nn \textwidth {#3}
76   \dim_set_eq:NN \columnwidth \textwidth
77   \@parboxrestore

```

```

78     \leavevmode
79     \raggedright
80     \ignorespaces
81 }

```

The `\@ignore` here means that any spaces after `\end{column}` are suppressed by a `\ignorespaces` inserted by the kernel.

```

82 {
83   \vbox_set_end:
84   \use:c { __talk_column_align_ \l__talk_column_alignment_tl :n }
85   { \vbox_unpack_drop:N \l__talk_tmp_box }
86   \hfil
87   \par
88   \__talk_action_end:
89   \@ignoretrue
90 }
91 </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.)

1.2 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`. This is set up such that the inactive versions insert a whatsit equal to that which would be present if they were active: that's needed for spacing.

```
__talk_action_:N The fallback action.
17 \cs_new_protected:Npn __talk_action_:N #1 { }
```

(End of definition for __talk_action_:N.)

```
__talk_action_alert:N At present a color selection.
18 \cs_new_protected:Npn __talk_action_alert:N #1
19 {
20   \bool_if:NTF #1
21   { \color_select:n { alert } }
22   { \color_select:n { . } }
23 }
```

(End of definition for `__talk_action_alert:N`.)

`__talk_action_invisible:N`
`__talk_action_visible:N`

Simply hide unconditionally.

```

24 \cs_new_protected:Npn \__talk_action_invisible:N #1
25 {
26   \bool_if:NTF #1
27     { \opacity_select:n { 0 } }
28     { \opacity_select:n { 1 } }
29 }
30 \cs_new_protected:Npn \__talk_action_visible:N #1
31 {
32   \bool_if:NTF #1
33     { \opacity_select:n { 1 } }
34     { \opacity_select:n { 0 } }
35 }

```

(End of definition for `__talk_action_invisible:N` and `__talk_action_visible:N`.)

`__talk_action_only_begin: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.

```

36 \cs_new_protected:Npn \__talk_action_only:N #1
37 {
38   \bool_if:NF #1
39     { \vbox_set:Nw \l__talk_tmp_box }
40 }
41 \cs_new_protected:Npn \__talk_action_only_end:N #1
42 {
43   \bool_if:NF #1
44     { \vbox_set_end: }
45 }

```

(End of definition for `__talk_action_only_begin:N` and `__talk_action_only_end:N`.)

`\l__talk_uncover_hidden_fp`

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

```

46 \NewTemplateType { hidden } { 0 }
47 \DeclareTemplateInterface { hidden } { talk } { 0 }
48 { opacity : real = 0 }
49 \DeclareTemplateCode { hidden } { talk } { 0 }
50 { opacity = \l__talk_uncover_hidden_fp }
51 { \opacity_select:n { \l__talk_uncover_hidden_fp } }
52 \DeclareInstance { hidden } { std } { talk } { }

```

(End of definition for `\l__talk_uncover_hidden_fp`.)

`__talk_action_uncover:N`

Use the template

```

53 \cs_new_protected:Npn \__talk_action_uncover:N #1
54 {
55   \bool_if:NTF #1
56     { \opacity_select:n { 1 } }
57     { \UseInstance { hidden } { std } }
58 }

```

(End of definition for `__talk_action_uncover:N`.)

`\only` Commands and environments where the payload applies when the material is not active on the slide.

```
\invisible
\uncover
59 \clist_map_inline:nn { only , invisible , uncover }
60 {
61   \ExpandArgs { cne } \NewDocumentCommand {#1}
62     { > { \__talk_overlay_arg:n } D <> { all } +m }
63     {
64       \group_begin:
65       \exp_not:c { __talk_action_ #1 :N } ##1
66       ##2
67       \cs_if_exist:cT { __talk_action_ #1 _end:N }
68       { \exp_not:c { __talk_action_ #1 _end:N } ##1 }
69       \group_end:
70     }
```

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

```
onlyenv (env.)
invisibleenv (env.)
uncoverenv (env.)
71 \ExpandArgs { nnee } \NewDocumentEnvironment { #1 env }
72 { > { \__talk_overlay_arg:n } D <> { all } }
73 { \exp_not:c { __talk_action_ #1 :N } ##1 }
74 {
75   \cs_if_exist:cT { __talk_action_ #1 _end:N }
76   { \exp_not:c { __talk_action_ #1 _end:N } ##1 }
77 }
78 }
```

`\alert` And those where the action applies when we are on the slide.

```
\visible
79 \clist_map_inline:nn { alert , visible }
80 {
81   \ExpandArgs { cne } \NewDocumentCommand {#1}
82     { > { \__talk_overlay_arg:n } D <> { all } +m }
83     {
84       \group_begin:
85       \exp_not:c { __talk_action_ #1 :N } ##1
86       ##2
87       \group_end:
88     }
```

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

```
alertenv (env.)
visibleenv (env.)
89 \ExpandArgs { nnee } \NewDocumentEnvironment { #1 env }
90 { > { \__talk_overlay_arg:n } D <> { all } }
91 { \exp_not:c { __talk_action_ #1 :N } ##1 }
92 { }
93 }
```

`\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).

```
94 \RenewDocumentCommand \only { D <> { all } +m }
95 {
```

```

96     \_talk_if_overlay:nT {#1}
97     {#2}
98 }

```

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

```

\l__talk_saved_overlays_bool
\l__talk_saved_action_str
\l__talk_saved_actions_bool
99 \bool_new:N \l__talk_saved_overlays_bool
100 \str_new:N \l__talk_saved_action_str
101 \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.)

actionenv

As we need data on not just overlays but also actions at the end of the environment, this has to be done manually. To allow working with environments but also items, the code needs to save data for the end function. The group is needed for cases where we are not in a L^AT_EX environment group.

```

\__talk_action_begin:n
\__talk_action_end:
102 \NewDocumentCommand \action { D <> { all } +m }
103 {
104     \group_begin:
105     \__talk_action_begin:n {#1}
106     #2
107     \__talk_action_end:
108     \group_end:
109 }
110 \NewDocumentEnvironment { actionenv } { D <> { all } }
111 { \__talk_action_begin:n {#1} }
112 { \__talk_action_end: }
113 \cs_new_protected:Npn \__talk_action_begin:n #1
114 {
115     \group_begin:
116     \__talk_decode_parse:n {#1}
117     \bool_set_eq:NN \l__talk_saved_overlays_bool
118     \l__talk_decode_overlays_bool
119     \str_set_eq:NN \l__talk_saved_action_str
120     \l__talk_decode_action_str
121     \bool_set_eq:NN \l__talk_saved_actions_bool
122     \l__talk_decode_actions_bool
123     \bool_if:NTF \l__talk_decode_overlays_bool
124     {
125         \use:c { __talk_action_ \l__talk_decode_action_str :N }
126         \l__talk_decode_actions_bool
127     }
128     { \UseInstance { hidden } { std } }
129 }
130 \cs_new_protected:Npn \__talk_action_end:
131 {
132     \bool_if:NT \l__talk_saved_overlays_bool
133     {
134         \cs_if_exist_use:cF
135         { __talk_action_ \l__talk_saved_action_str _end:N }
136         { \use_none:n }
137         \l__talk_saved_actions_bool

```

```

138     }
139     \group_end:
140 }

```

(End of definition for \action, _talk_action_begin:n, and _talk_action_end:. This function is documented on page ??.)

1.3 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.

```

141 \NewDocumentCommand \alt { D <> { all } +m +m }
142 {
143   \_talk\_if\_overlay:nTF {#1}
144     {#2}
145     {#3}
146 }

```

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

\onslide Simply make transparent: we will likely need to save the original opacity level. To allow us to apply independent of group level, a little work is needed.

```

\__talk\_onslide:n
\__talk\_onslide\_reset:
147 \NewDocumentCommand \onslide { D <> { all } }
148 { \__talk\_onslide:n {#1} }
149 \cs_new_protected:Npn \__talk\_onslide:n #1
150 {
151   \tl\_use:N \g\_talk\_onslide\_tl
152   \_talk\_if\_overlay:nTF {#1}
153     { \__talk\_onslide\_reset: }
154     {
155       \opacity\_select:n { 0 }
156       \tl\_gset:Nn \g\_talk\_onslide\_escape\_tl
157         {
158           \opacity\_select:n { 0 }
159           \group\_insert\_after:N \g\_talk\_onslide\_escape\_tl
160         }
161       \group\_insert\_after:N \g\_talk\_onslide\_escape\_tl
162       \tl\_gset:Nn \g\_talk\_onslide\_tl
163         {
164           \tl\_gclear:N \g\_talk\_onslide\_tl
165           \tl\_gclear:N \g\_talk\_onslide\_escape\_tl
166           \__talk\_onslide\_reset:
167         }
168     }
169 }
170 \cs_new_protected:Npn \__talk\_onslide\_reset: { \opacity\_select:n { 1 } }

```

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

```

\g\_talk\_onslide\_tl
\g\_talk\_onslide\_escape\_tl
171 \tl\_new:N \g\_talk\_onslide\_tl
172 \tl\_new:N \g\_talk\_onslide\_escape\_tl

```

(End of definition for `\g__talk_onslide_tl` and `\g__talk_onslide_escape_tl`.)

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

```

173 \NewDocumentCommand \temporal { D <> { all } +m +m +m }
174 {
175   \__talk_if_overlay:nTF {#1}
176     {#3}
177     {
178       \bool_if:NTF \g__talk_slide_continue_bool
179         {#4}
180         {#2}
181     }
182 }

```

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

`\pause` A thin wrapper.

```

183 \NewDocumentCommand \pause { o }
184 {
185   \IfNoValueTF {#1}
186     { \int_gincr:N \g__talk_pauses_int }
187     { \int_gset:Nn \g__talk_pauses_int {#1} }
188   \exp_args:Ne \__talk_onslide:n { \int_use:N \g__talk_pauses_int - }
189 }

```

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

1.4 Fixed-size areas

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

```

190 \cs_new_protected:Npn \__talk_overprint_begin:n #1
191 {
192   \par
193   \vbox_set_to_wd:Nnw \l__talk_tmp_box {#1}
194   \raggedright
195   \ignorespaces
196 }

```

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

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

```

197 \NewDocumentEnvironment { overlayarea } { m m }
198 { \__talk_overprint_begin:n {#1} }
199 {
200   \vbox_set_end:
201   \vbox_to_ht:nn {#2}
202   {
203     \box_use_drop:N \l__talk_tmp_box
204     \vfil
205   }
206   \par
207 }

```

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

```
208 \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.

```
209 \cs_new:Npn \__talk_frame_overprint:
```

```
210 {
```

```
211   \int_to_Roman:n \g__talk_frame_int
```

```
212   \int_to_roman:n \l__talk_overprint_int
```

```
213 }
```

(End of definition for `__talk_frame_overprint:`.)

`__talk_overprint_int` 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.

```
214 \NewDocumentEnvironment { overprint } { 0 { \textwidth } }
```

```
215 { \__talk_overprint_begin:n {#1} }
```

```
216 {
```

```
217   \vbox_set_end:
```

```
218   \int_incr:N \l__talk_overprint_int
```

```
219   \__talk_overprint_save_ht:
```

```
220   \cs_if_exist:cTF
```

```
221     { overprint@ \__talk_frame_overprint: } 
```

```
222     {
```

```
223       \dim_compare:vNnTF
```

```
224         { overprint@ \__talk_frame_overprint: } 
```

```
225         > { \box_ht:N \l__talk_tmp_box } 
```

```
226         {
```

```
227           \vbox_to_ht:vn
```

```
228             { overprint@ \__talk_frame_overprint: } 
```

```
229             {
```

```
230               \box_use_drop:N \l__talk_tmp_box
```

```
231               \vfil
```

```
232             }
```

```
233           }
```

```
234           { \box_use_drop:N \l__talk_tmp_box } 
```

```
235         }
```

```
236         { \box_use_drop:N \l__talk_tmp_box } 
```

```
237     \par
```

```
238 }
```

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.

```
239 \cs_new_protected:Npn \__talk_overprint_save_ht:
```

```
240 {
```

```
241   \tl_if_exist:cF { g__talk_overprint_ \__talk_frame_overprint: _tl } 
```

```
242   {
```

```
243     \tl_new:c { g__talk_overprint_ \__talk_frame_overprint: _tl } 
```

```
244     \tl_gset:cn { g__talk_overprint_ \__talk_frame_overprint: _tl } 
```

```
245     { Opt } 
```



```

246     }
247     \tl_gset:ce { g__talk_overprint_ \__talk_frame_overprint: _tl }
248     {
249         \dim_max:vn { g__talk_overprint_ \__talk_frame_overprint: _tl }
250         { \box_ht:N \l__talk_tmp_box }
251     }
252     \legacy_if:nT { @files }
253     {
254         \iow_now:Ne \@auxout
255         {
256             \gdef \exp_not:c { overprint@ \__talk_frame_overprint: }
257             {
258                 \exp_not:v { g__talk_overprint_ \__talk_frame_overprint: _tl }
259             }
260         }
261     }
262     \hook_gput_code:nne { enddocument / afterlastpage } { talk }
263     { \__talk_overprint_check_ht:n { \__talk_frame_overprint: } }
264 }
265 \cs_new_protected:Npn \__talk_overprint_check_ht:n #1
266 {
267     \bool_lazy_and:nnF
268     { \exp_not:N \cs_if_exist_p:c { overprint@ #1 } }
269     {
270         \dim_compare_p:vNv { overprint@ #1 } = { g__talk_overprint_ #1 _tl }
271     }
272     {
273         \msg_warning:nn { talk } { overprint-ht }
274         \cs_gset_protected:Npn \__talk_overprint_check_ht:n ##1 { }
275     }
276 }
277 \msg_new:nnn { talk } { overprint-ht }
278 {
279     Overprint~area~height~has~changed:\\
280     rerun~LaTeX.
281 }

```

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

1.5 Adding overlays to existing commands

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 282 \tl_map_inline:nn
\textrm      283 {
\textsc      284     \textbf
\textsf      285     \textit
\textsl      286     \textmd
\texttt      287     \textnormal
\textup      288     \textrm
\emph        289     \textsc
\stdtextbf   290     \textsf
\stdtextit   291     \textsl
\stdtextmd
\stdtextnormal
\stdtextrm
\stdtextsc
\stdtextsf
\stdtextsl
\stdtexttt
\stdtextup
\stdemph
\__talk_textcmd equiv:n

```

```

292 \texttt
293 \textup
294 \emph
295 }
296 {
297 \ExpandArgs { c } \NewCommandCopy { std \cs_to_str:N #1 } #1
298 \ExpandArgs { Nne } \RenewDocumentCommand #1
299 { D <> { all } +m }
300 {
301 \exp_not:N \__talk_if_overlay:nTF {##1}
302 { \exp_not:c { std \cs_to_str:N #1 } }
303 { \exp_not:N \__talk_textcmd_equiv:n }
304 {##2}
305 }
306 }
307 \cs_new_protected:Npn \__talk_textcmd_equiv:n #1
308 {
309 \mode_if_math:TF
310 { { \mbox {#1} } }
311 {
312 \mode_leave_vertical:
313 {#1}
314 }
315 }

```

(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`

```

316 \RequirePackage { graphicx }
317 \NewCommandCopy \stdincludegraphics \includegraphics
318 \RenewDocumentCommand \includegraphics { s D <> { all } o o m }
319 {
320 \__talk_if_overlay:nT {#2}
321 {
322 \use:e
323 {
324 \exp_not:N \stdincludegraphics
325 \IfBooleanT #1 { * }
326 \IfNoValueF {#3} { [ \exp_not:n { {#3} } ] }
327 \IfNoValueF {#4} { [ \exp_not:n { {#4} } ] }
328 }
329 {#5}
330 }
331 }

```

(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`

```

332 \RenewDocumentCommand \label { D <> { 1 } m }
333 {

```

```

334 \@@bsphack
335 \__talk_if_overlay:nT {#1}
336 { \__talk_label:n {#2} }
337 \@@esphack
338 }
339 \cs_new_protected:Npn \__talk_label:n #1
340 {
341   \begingroup
342     \UseHookWithArguments { label } { 1 } {#1}
343     \protected@write \@auxout { }
344     {
345       \string \newlabel {#1}
346       {
347         { \@currentlabel }
348         { \thepage }
349         { \@currentlabelname }
350         { \@currentHref }
351         { \@kernel@reserved@label@data }
352       }
353     }
354   \endgroup
355 }

```

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

```

356 </class>

```

Part VIII

ltx-talk-required – “Required” definitions

1 ltx-talk-required implementation

Start the DocStrip guards.

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

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

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

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

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

1.1 Standard design settings

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

```

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

```

1.2 List support

```

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

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

```

Part IX

ltx-talk-structure – Structural commands

1 ltx-talk-structure implementation

Start the DocStrip guards.

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

1.1 Frame title

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

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

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

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

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

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

```

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

```

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

1.2 Sectioning

```

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

<pre> \section \subsection \subsubsection \thesection \thesubsection \thesubsubsection </pre>	<p>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</p>
---	---

as in article). As (subsub)section numbers never increment inside frames, we remove these counters from the general tracker.

```

72 \newcounter { section }
73 \newcounter { subsection } [ section ]
74 \newcounter { subsubsection } [ subsection ]
75 \seq_gremove_all:Nn \l__talk_cnt_reset_seq { section }
76 \seq_gremove_all:Nn \l__talk_cnt_reset_seq { subsection }
77 \seq_gremove_all:Nn \l__talk_cnt_reset_seq { subsubsection }
78 \cs_gset:Npn \thesection { \@arabic \c@section }
79 \cs_gset:Npn \thesubsection { \thesection . \@arabic \c@subsection }
80 \cs_gset:Npn \thesubsubsection { \thesubsection . \@arabic \c@subsubsection }

```

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

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

```

81 \seq_set_from_clist:Nn \l_tmpa_seq
82   { section , subsection , subsubsection }
83 \seq_map_indexed_inline:Nn \l_tmpa_seq
84   {
85     \use:e
86     {
87       \NewDocumentCommand \exp_not:c { insert #2 } { { }
88       {
89         \exp_not:N \tl_use:N
90         \exp_not:c { l__talk_ #2 _tl }
91       }
92       \NewDocumentCommand \exp_not:c {#2}
93       { s D <> { all } 0 {##4} m }
94       {
95         \exp_not:N \refstepcounter {#2}
96         \tag_tool:n { sec-start = #2 , restore-para }
97         \tl_set:Nn \exp_not:c { l__talk_ #2 _tl } {##4}
98         \tl_gset_eq:NN \exp_not:c { g__talk_ #2 _tl }
99         \exp_not:c { l__talk_ #2 _tl }
100        \str_if_eq:nnT {#2} { section }
101          { \tl_clear:N \exp_not:N \l__talk_subsubsection_tl }
102        \str_if_eq:nnF {#2} { subsubsection }
103          { \tl_clear:N \exp_not:N \l__talk_subsubsection_tl }
104        \exp_not:N \addcontentsline { toc } {#2}
105        {
106          \exp_not:N \int_compare:nNnF {#1} >
107            { \exp_not:N \value { secnumdepth } }
108            {
109              \exp_not:N \protect \exp_not:N \numberline
110                { \exp_not:c { the #2 } }
111            }
112          ##4

```



```

113         }
114         \hook_use:n { #2 / begin }
115     }
116     \hook_new:n { #2 / begin }
117 }
118 }

```

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

`__talk_section_tagged:`

```

119 \cs_new_protected:Npn \__talk_section_tagged:
120 {
121     \clist_map_inline:nn { section , subsection , subsubsection }
122     {
123         \tl_if_empty:cF { g__talk_ ##1 _ tl }
124         {
125             \__talk_header_tag_begin:e
126             {
127                 tag = ##1 ,
128                 title = { \text_purify:v { g__talk_ ##1 _ tl } } ,
129             }
130             \__talk_header_tag_end:
131             \tl_gclear:c { g__talk_ ##1 _ tl }
132         }
133     }
134 }

```

(End of definition for `__talk_section_tagged:`)

1.3 Table of contents

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

```

135 \cs_gset_protected:Npn \@starttoc #1
136 {
137     \begingroup
138     \makeatletter
139     \UseTaggingSocket { toc / starttoc / before } {#1}
140     \@input { \jobname .#1 }
141     \UseTaggingSocket { toc / starttoc / after } {#1}
142     \legacy_if:nT { @files }
143     {
144         \AddToHook { enddocument / afterlastpage }
145         {
146             \expandafter \newwrite \csname tf@ #1 \endcsname
147             \immediate \openout \csname tf@ #1 \endcsname \jobname .#1 \relax
148         }
149     }
150     \nobreakfalse
151     \endgroup
152 }

```

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

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

```

153 \NewDocumentCommand \tableofcontents { 0 { } }
154 {
155   \group_begin:
156   \@starttoc { toc }
157   \group_end:
158 }

```

(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`

```

159 \cs_new_protected:Npn \l@section #1#2
160 { \__talk_toc_aux:nnnn { 1 } { \bfseries \color { structure } } {#1} {#2} }
161 \cs_new_protected:Npn \l@subsection #1#2
162 {
163   \__talk_toc_aux:nnnn
164   { 2 }
165   {
166     \skip_set:Nn \leftskip { 2em }
167     \color { . }
168   }
169   {#1} {#2}
170 }
171 \cs_new_protected:Npn \l@subsubsection #1#2
172 {
173   \__talk_toc_aux:nnnn
174   { 3 }
175   {
176     \skip_set:Nn \leftskip { 4em }
177     \color { . }
178     \footnotesize
179   }
180   {#1} {#2}
181 }
182 \cs_new_protected:Npn \__talk_toc_aux:nnnn #1#2#3#4
183 {
184   \int_compare:nNnTF { \value { section } } < 1
185   { \use:n }
186   { \__talk_toc_dest:n }
187   { \__talk_toc_level:nnnn {#1} {#2} {#3} {#4} }
188 }

```

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.

```

189 \cs_new_protected:Npn \__talk_toc_dest:n
190 {
191   \exp_after:wN \__talk_toc_dest:w \@contentsline@destination
192   . 0 . 0 . 0 . \q_stop
193 }
194 \cs_new_protected:Npn \__talk_toc_dest:w #1 . #2 . #3 . #4 . #5 \q_stop #6

```

```

195 {
196   \int_compare:nNnTF { \value { section } } = {#2}
197     {#6}
198     {
199       \group_begin:
200         \opacity_select:n { 0.2 }
201         #6
202       \group_end:
203     }
204   }
205   \cs_new_protected:Npn \__talk_toc_level:nnnn #1#2#3#4
206   {
207     \int_compare:nNnF {#1} > { \value { tocdepth } }
208     {
209       \group_begin:
210         \noindent
211         #2
212         \UseHookWithArguments { contentsline / text / before } { 4 }
213         {#1} {#3} {#4} { \@contentsline@destination }
214         #3
215         \UseHookWithArguments { contentsline / text / after } { 4 }
216         {#1} {#3} {#4} { \@contentsline@destination }
217         \UseHookWithArguments { contentsline / page / before } { 4 }
218         {#1} {#3} {#4}
219         { \@contentsline@destination }
220         \UseHookWithArguments { contentsline / page / after } { 4 }
221         {#1} {#3} {#4}
222         { \@contentsline@destination }
223         \par
224       \group_end:
225       \vfil
226     }
227   }

```

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

```

228 \setcounter { tocdepth } { 2 }

```

1.4 Block environments

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

```

quote (env.) 229 \NewDocumentEnvironment { description } { } { } { } { }
quotation (env.) 230 \NewDocumentEnvironment { quote } { } { } { } { }
verse (env.) 231 \NewDocumentEnvironment { quotation } { } { } { } { }
stdquote (env.) 232 \NewDocumentEnvironment { verse } { } { } { } { }
stdquotation (env.) 233 \AddToHook { begindocument / before }
stdverse (env.) 234 {
235   \clist_map_inline:nn { quote , quotation , verse }
236   {
237     \NewEnvironmentCopy { std #1 } {#1}
238     \RenewDocumentEnvironment {#1} { D <> { all } !0 { } }
239     {
240       \__talk_action_begin:n {##1}
241       \begin { std #1 } [ {##2} ]
242       \ignorespaces

```

```

243     }
244     {
245         \end { std #1 }
246         \__talk_action_end:
247     }
248 }
249 }

```

block (env.)

```

250 \NewDocumentEnvironment { block } { D <> { all } m }
251 {
252     \__talk_action_begin:n {#1}
253     \par
254     \vbox_set:Nw \l__talk_tmp_box
255     \group_begin:
256         \medskip
257         \leavevmode
258         \normalfont \large \bfseries
259         \color { structure }
260         #2
261         \par
262         \medskip
263     \group_end:
264 }
265 {
266     \vbox_set_end:
267     \box_use:N \l__talk_tmp_box
268     \par
269     \__talk_action_end:
270 }

```

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!

```

271 \AddToHook { begindocument / before }
272 {
273     \NewCommandCopy \stditem \item
274     \RenewDocumentCommand \item { d <> = { label } o }
275     {
276         \IfNoValueTF {#2}
277         { \stditem }
278         { \stditem [ {#2} ] }
279     \IfNoValueTF {#1}
280     {
281         \exp_after:wN \__talk_item_parse_spec:w
282         \l__talk_action_spec_str < all > \q_stop
283     }
284     { \__talk_item_parse_spec:n {#1} }
285 }
286 }

```

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.

```

287 \cs_new_protected:Npn \__talk_item_parse_spec:w #1 < #2 > #3 \q_stop
288 { \__talk_item_parse_spec:n {#2} }
289 \cs_new_protected:Npn \__talk_item_parse_spec:n #1
290 {
291   \tl_if_blank:nF {#1}
292   {
293     \tl_set:Nx \l__talk_list_end_tl
294     {
295       \exp_not:N \int_compare:nNnT \tex_currentgrouplevel:D =
296       { \int_use:N \tex_currentgrouplevel:D + 1 }
297       {
298         \__talk_action_end:
299         \tl_clear:N \exp_not:N \l__talk_list_end_tl
300       }
301     }
302     \__talk_action_begin:n {#1}
303   }
304 }

```

(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`

```

305 \tl_new:N \l__talk_list_end_tl

```

(End of definition for `\l__talk_list_end_tl`.)

`__block_inter_item:`

`\endblockenv`

There are no currently no hooks for insertion at the end of list items, so we have to do it manually. We cannot target `__block_list_item_end:/__block_list_end:` as these change definition if tagging is suspended.

```

306 \cs_gset_protected:Npn \__block_inter_item:
307 {
308   \legacy_if:nT { @inlabel }
309   { \indent \par }
310   \mode_if_horizontal:T
311   {
312     \__block_skip_remove_last:
313     \__block_skip_remove_last:
314     \par
315   }
316   \l__talk_list_end_tl
317   \__kernel_list_item_end:
318   \__kernel_list_item_begin:
319   \addpenalty \@itempenalty
320   \addvspace \itemsep
321 }
322 \cs_gset:Npn \endblockenv
323 {
324   \__block_debug_typeout:n { blockenv-common~ending \on@line }
325   \bool_if:NT \l__block_level_incr_bool

```

```

326     { \int_gdecr:N \g_block_nesting_depth_int }
327 \legacy_if:nT { @inlabel }
328 {
329     \mode_leave_vertical:
330     \legacy_if_gset_false:n { @inlabel }
331 }
332 \__block_if_list:T
333 { \legacy_if:nT { @newlist } { \@noitemerr } }
334 \mode_if_horizontal:TF
335 {
336     \__block_skip_remove_last:
337     \__block_skip_remove_last:
338     \par
339 }
340 { \@inmatherr { \end { \@currenvir } } }
341 \l__talk_list_end_tl
342 \__kernel_displayblock_end:
343 \__block_if_list:T { \legacy_if_gset_false:n { @newlist } }
344 \legacy_if:nF { @noparlist }
345 {
346     \__block_skip_set_to_last:N \l_tmpa_skip
347     \dim_compare:nNnT \l_tmpa_skip > \c_zero_dim
348     {
349         \skip_vertical:n { - \l_tmpa_skip }
350         \skip_vertical:n { \l_tmpa_skip + \parskip - \@outerparskip }
351     }
352     \addpenalty \@endparpenalty
353     \addvspace \l__block_topsepadd_skip
354 }
355 \socket_use:n { block / endpe }
356 }

```

(End of definition for `__block_inter_item:` and `\endblockenv`. This function is documented on page ??.)

```

itemize (env.) Allow for the classical beamer syntax.
enumerate (env.) 357 \AddToHook { begindocument / before }
description (env.) 358 {
359     \clist_map_inline:nn { itemize , enumerate , description }
360     {
361         \RenewDocumentEnvironment {#1} { = { action-spec } !o }
362         {
363             \IfNoValueTF {##1}
364             { \UseInstance { blockenv } {#1} { } }
365             { \UseInstance { blockenv } {#1} {##1} }
366         }
367         { \endblockenv }
368     }
369 }

```

And add the structural color to item labels.

```

370 \AddToHook { begindocument / before }
371 {
372     \EditInstance { item } { basic }
373     { label-format = \color { structure } #1 }

```

```

374 \EditInstance { item } { description }
375     { label-format = \normalfont \bfseries \color { structure } #1 }
376 }

```

`\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.

```

377 \keys_define:nn { template / block / display }
378   { action-spec .str_set:N = \l__talk_action_spec_str }

```

(End of definition for `\l__talk_action_spec_str`.)

1.6 Theorems, *etc.*

`\newtheorem` We need to extend the creation of theorems in two ways: add the overlay argument, and
`\stdnewtheorem` add the counter to the list of those reset during overlay creation.

```

379 \NewCommandCopy \stdnewtheorem \newtheorem
380 \RenewDocumentCommand \newtheorem { m O {#1} m o }
381 {
382   \IfNoValueTF {#4}
383     { \stdnewtheorem {#1} [ {#2} ] {#3} }
384     { \stdnewtheorem {#1} [ {#2} ] {#3} [ {#4} ] }
385   \NewEnvironmentCopy { std #1 } {#1}
386   \RenewDocumentEnvironment {#1} { D <> { all } }
387   {
388     \__talk_action_begin:n {##1}
389     \begin { std #1 }
390     \ignorespaces
391   }
392   {
393     \end { std #1 }
394     \__talk_action_end:
395   }
396 }

```

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

```

397 </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>
```

```
\institute Simple storage at present: we use names similar to the kernel ones for author, etc., as
\subtitle this makes data management easier.
\@institute 3 \cs_new_nopar:Npn \@institute { }
\@subtitle 4 \cs_new_nopar:Npn \@subtitle { }
5 \NewDocumentCommand \institute { = { short-institute } 0 {#2} m }
6 { \cs_gset_nopar:Npn \@institute {#2} }
7 \NewDocumentCommand \subtitle { = { short-subtitle } 0 {#2} m }
8 { \cs_gset_nopar:Npn \@subtitle {#2} }
```

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

```
\l__talk_titlelem_after_skip As the various elements of the titlepage share certain characteristics, we use a single
\l__talk_titlelem_before_skip template and split them as instances.
\l__talk_titlelem_color_tl
\l__talk_titlelem_font_tl
\l__talk_titlelem_tag_begin_tl
\l__talk_titlelem_tag_end_tl
9 \NewTemplateType { titlepage-element } { 1 }
10 \DeclareTemplateInterface { titlepage-element } { talk } { 1 }
11 {
12   after-skip : length = 0em ,
13   before-skip : length = 0em ,
14   color : tokenlist = . ,
15   font : tokenlist = \normalfont ,
16   tag-begin : tokenlist = ,
17   tag-end : tokenlist =
18 }
19 \DeclareTemplateCode { titlepage-element } { talk } { 1 }
20 {
21   after-skip = \l__talk_titlelem_after_skip ,
22   before-skip = \l__talk_titlelem_before_skip ,
23   color = \l__talk_titlelem_color_tl ,
24   font = \l__talk_titlelem_font_tl ,
25   tag-begin = \l__talk_titlelem_tag_begin_tl ,
26   tag-end = \l__talk_titlelem_tag_end_tl
27 }
28 {
29   \tl_if_empty:nF {#1}
30   {
31     \vspace { \l__talk_titlelem_before_skip }
32     \group_begin:
33       \tl_if_empty:nF \l__talk_titlelem_color_tl
34       { \color_select:V \l__talk_titlelem_color_tl }
35       \l__talk_titlelem_font_tl
36       \l__talk_titlelem_tag_begin_tl
```



```

37         #1
38         \par
39         \l__talk_titlelem_tag_end_tl
40     \group_end:
41     \vspace { \l__talk_titlelem_after_skip }
42 }
43 }

```

Standard settings are taken from beamer with minor adjustments.

```

44 \DeclareInstance { titlepage-element } { author } { talk }
45 { before-skip = 1em }
46 \DeclareInstance { titlepage-element } { date } { talk }
47 { after-skip = 0.5em }
48 \DeclareInstance { titlepage-element } { institute } { talk }
49 { font = \scriptsize }
50 \DeclareInstance { titlepage-element } { subtitle } { talk }
51 { before-skip = 0.25em , color = structure }
52 \DeclareInstance { titlepage-element } { title } { talk }
53 {
54     color = structure ,
55     font = \Large ,
56     tag-begin = \tag_struct_begin:n { tag = Title } ,
57     tag-end = \tag_struct_end:
58 }

```

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

```

\l__talk_titlepage_order_clist
\l__talk_titlepage_alignment_tl
\l__talk_titlepage_framestyle_tl
\l__talk_frame_alignment_tl

```

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.

```

59 \NewTemplateType { titlepage } { 0 }
60 \DeclareTemplateInterface { titlepage } { talk } { 0 }
61 {
62     element-order : commalist =
63     {
64         title ,
65         subtitle ,
66         author ,
67         institute ,
68         date
69     } ,
70     framestyle : tokenlist = talk ,
71     horizontal-alignment : choice { left , center , right } = center ,
72     vertical-alignment : choice { bottom , center , stretch , top } = center
73 }
74 \DeclareTemplateCode { titlepage } { talk } { 0 }
75 {
76     element-order = \l__talk_titlepage_order_clist ,
77     framestyle = \l__talk_titlepage_framestyle_tl ,
78     horizontal-alignment =
79     {
80         left = \tl_set:Nn \l__talk_titlepage_alignment_tl { flushleft } ,
81         center = \tl_set:Nn \l__talk_titlepage_alignment_tl { center } ,
82         right = \tl_set:Nn \l__talk_titlepage_alignment_tl { flushright }
83     } ,

```

```

84     vertical-alignment =
85     {
86         bottom = \tl_set:Nn \l__talk_frame_alignment_tl { bottom } ,
87         center = \tl_set:Nn \l__talk_frame_alignment_tl { center } ,
88         stretch = \tl_set:Nn \l__talk_frame_alignment_tl { stretch } ,
89         top = \tl_set:Nn \l__talk_frame_alignment_tl { top }
90     }
91 }
92 {
93     \tl_if_empty:NF \l__talk_titlepage_framestyle_tl
94     { \exp_args:NV \thispagestyle \l__talk_titlepage_framestyle_tl }
95     \begin { \l__talk_titlepage_alignment_tl }
96         \cs_set_protected:Npn \and { \quad }
97         \clist_map_inline:Nn \l__talk_titlepage_order_clist
98         {
99             \ExpandArgs { nnv } \UseInstance { titlepage-element }
100             {##1} { @ ##1 }
101         }
102     \end { \l__talk_titlepage_alignment_tl }
103 }

```

(End of definition for \l__talk_titlepage_order_clist and others.)

\maketitle A very simple setup.

```

104 \NewDocumentCommand \maketitle { 0 {} }
105 {
106     \bool_if:NTF \l__talk_frame_bool
107     { \UseTemplate { titlepage } { talk } {#1} }
108     {
109         \begin { frame }
110             \UseTemplate { titlepage } { talk } {#1}
111         \end { frame }
112     }
113 }

```

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

```

114 \</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

@ commands:

\@_decode_overlay_+ :nw 121
 \\ 279

A

\action 102
 actionenv (env.) 102
 \addcontentsline 104
 \addpenalty 319, 352
 \AddToHook 62,
 144, 233, 255, 271, 316, 357, 370, 383
 \addtolength 48
 \addvspace 320, 353
 \alert 79
 alertenv (env.) 89
 \alt 141
 \and 96
 \arabic 380
 \arraycolsep 25
 \arrayrulewidth 27

B

\begin 95, 109, 241, 389
 \begingroup 137, 341
 \bfseries 21, 39, 160, 258, 375
 \bigskipamount 18
 block (env.) 250
 block commands:
 \g_block_nesting_depth_int 326
 block internal commands:
 __block_debug_typeout:n 324
 __block_if_list:TF 332, 343
 __block_inter_item: 306, 306
 \l_block_level_incr_bool 325
 __block_list_end: 50
 __block_list_item_end: 50
 __block_skip_remove_last:
 312, 313, 336, 337
 __block_skip_set_to_last:N 346
 \l_block_topsepadd_skip 353

bool commands:

\bool_do_while:Nn 27
 \bool_gset_false:N 35, 72, 404
 \bool_gset_true:N . 203, 211, 217, 396
 \bool_if:NTF 6,
 20, 26, 32, 38, 40, 43, 46, 55, 83,
 106, 123, 125, 132, 178, 246, 325, 410

\bool_lazy_and:nnTF 39, 214, 267
 \bool_lazy_or:nnTF 5, 19
 \bool_new:N 3,
 3, 7, 8, 13, 99, 101, 374, 375, 376
 \bool_set_eq:NN 117, 121
 \bool_set_false:N
 25, 26, 91, 110, 123, 416, 436
 \bool_set_true:N 22, 27,
 42, 59, 141, 178, 198, 213, 389, 424, 444
 \c_false_bool 15
 \c_true_bool 14

box commands:

\box_dp:N 13
 \box_ht:N 60, 225, 250
 \box_move_down:nn 58
 \box_new:N 4, 43
 \box_use:N 267
 \box_use_drop:N
 24, 203, 230, 234, 236, 312
 \box_wd:N 18

box internal commands:

__box_dim_eval:n 10, 13, 18, 21
 __box_set_to_wd: 17, 22

C

\clearpage 90

clist commands:

\clist_const:Nn 48
 \clist_if_in:NnTF 55, 177
 \clist_map_break: 199, 218
 \clist_map_inline:Nn 97, 180, 303
 \clist_map_inline:nn
 59, 79, 121, 235, 359
 \clist_new:N 10, 14
 \clist_pop:NNTF 299
 \clist_set:Nn 33, 176
 \color .. 4, 11, 160, 167, 177, 259, 373, 375

color commands:

\color_group_begin: 11, 23
 \color_group_end: 11
 \color_math:nn 9, 25
 \color_math:nnn 10, 26
 \color_select:n 7, 16,
 21, 22, 34, 35, 36, 36, 202, 244, 297, 328
 \color_select:nn 8, 17, 37

\colorlet 45

column (env.) 69

columns (env.) 9

\ExpandArgs	61, 71, 81, 89, 99, 248, 297, 298, 301, 306	\ignorespaces	18, 80, 195, 242, 390
		\immediate	147
		\includegraphics	316
		\indent	309
		\insertsection	81
		\insertsubsection	81
		\insertsubsubsection	81
		\institute	3
		int commands:	
		\int_compare:nNnTF	106, 184, 196, 196, 202, 207, 208, 210, 295
		\int_compare_p:nNn	215, 216
		\int_gdecr:N	326
		\int_gincr:N	29, 126, 186, 388
		\int_gset:Nn	187, 395
		\int_gset_eq:NN	71, 127, 132, 137
		\int_gzero:N	25
		\int_incr:N	218
		\int_max:nn	171
		\int_new:N	4, 5, 68, 126, 143, 208, 377
		\int_to_Roman:n	211
		\int_to_roman:n	212
		\int_use:N	8, 14, 48, 56, 64, 188, 296, 382
		\c_max_int	190, 216
		\c_one_int	67
		\invisible	59
		invisibleenv (env.)	71
		iow commands:	
		\iow_now:Nn	254
		\item	52, 271
		itemize (env.)	357
		\itemsep	54, 62, 69, 320
		J	
		\jobname	140, 147
		K	
		kernel internal commands:	
		__kernel_displayblock_end:	342
		__kernel_list_item_begin:	318
		__kernel_list_item_end:	317
		keys commands:	
		\l_keys_choice_tl	57
		\keys_define:nn	3, 5, 28, 47, 59, 147, 377
		\keys_set:nn	7, 13, 46, 66, 73, 160, 415, 423, 435, 443
		\l_keys_value_tl	43, 157
		L	
		\label	332
		\labelenumi	34
		\labelenumii	35
		\labelenumiii	36
		\labelenumiv	37
		I	
\IfBooleanT	325		
\ifcase	5		
\IfNoValueF	326, 327		
\IfNoValueTF	15, 24, 35, 44, 185, 276, 279, 363, 382		
		H	
hbox commands:			
\hbox:n	56		
\hbox_set_end:	23		
\hbox_set_to_wd:Nnw	14		
\headsep	220, 239, 240		
\hfil	17, 22, 86, 270, 310, 339, 351, 356, 367		
hook commands:			
\hook_gput_code:nnn	34, 78, 262		
\hook_new:n	116		
\hook_use:n	114		
\hspace	199, 206		
\hypersetup	126		
		F	
\fboxrule	32		
\fboxsep	31		
\fi	18		
file commands:			
\file_if_exist_input:nTF	74		
\file_input:n	76		
\footins	30		
\footnotesize	178		
\footskip	286, 287		
fp commands:			
\fp_eval:n	79		
\fp_to_dim:n	89		
\frame	24, 26, 385		
frame	410		
frame*	410		
\framesubtitle	10		
\frametitle	5, 417, 427		
		G	
\gdef	256		
\geometry	3		
group commands:			
\group_begin:	11, 32, 33, 34, 54, 60, 64, 84, 104, 115, 155, 199, 200, 209, 242, 255, 327		
\c_group_begin_token	20		
\group_end:	39, 40, 40, 50, 58, 63, 69, 87, 108, 139, 157, 202, 205, 224, 252, 263, 331		
\group_insert_after:N	22, 159, 161		
		E	
		\e	140, 147

talk internal commands:

__talk_action:N [17](#), [17](#)
 __talk_action_alert:N [18](#), [18](#)
 __talk_action_begin:n ... [11](#), [71](#),
 [102](#), [105](#), [111](#), [113](#), [240](#), [252](#), [302](#), [388](#)
 __talk_action_end: [26](#), [88](#),
 [102](#), [107](#), [112](#), [130](#), [246](#), [269](#), [298](#), [394](#)
 __talk_action_invisible:N .. [24](#), [24](#)
 __talk_action_only:N [36](#)
 __talk_action_only_begin:N [36](#)
 __talk_action_only_end:N ... [36](#), [41](#)
 \l__talk_action_spec_str [147](#), [282](#), [377](#)
 __talk_action_uncover:N [53](#), [53](#)
 __talk_action_visible:N [24](#), [30](#)
 \l__talk_aspect_ratio_str ... [47](#), [93](#)
 \l__talk_cnt_reset_seq
 [75](#), [76](#), [77](#), [116](#), [131](#), [136](#), [144](#)
 __talk_cnt_restore: [84](#), [129](#), [134](#)
 __talk_cnt_save: [75](#), [129](#), [129](#)
 __talk_column_align_bottom:n [50](#), [50](#)
 __talk_column_align_center:n [50](#), [52](#)
 __talk_column_align_top:n .. [50](#), [67](#)
 \l__talk_column_alignment_tl . [28](#), [84](#)
 \l__talk_columns_wd_tl [5](#), [14](#), [15](#)
 __talk_decode_action:n .. [85](#), [94](#), [94](#)
 __talk_decode_action:w .. [94](#), [96](#), [101](#)
 \l__talk_decode_action_str
 [12](#), [18](#), [111](#), [120](#), [125](#)
 \l__talk_decode_actions_bool ...
 [13](#), [25](#), [122](#), [126](#)
 \l__talk_decode_actions_clist ... [13](#)
 \l__talk_decode_actions_str .. [13](#), [29](#)
 \l__talk_decode_arg_str
 [9](#), [24](#), [30](#), [117](#), [162](#)
 __talk_decode_check:n . [127](#), [174](#), [174](#)
 __talk_decode_check:nw [174](#), [181](#), [184](#)
 __talk_decode_check_range:nnn .
 [174](#), [190](#), [191](#), [206](#)
 __talk_decode_check_single:nn .
 [174](#), [187](#), [194](#)
 __talk_decode_mode:n [44](#), [53](#), [53](#)
 __talk_decode_mode:nn ... [76](#), [79](#), [81](#)
 __talk_decode_mode:w [53](#), [62](#), [68](#)
 __talk_decode_mode_aux:n [53](#)
 __talk_decode_overlay.:nw [121](#)
 __talk_decode_overlay_aux:nNN .
 [121](#), [142](#), [145](#), [146](#)
 __talk_decode_overlay_offset:nNn
 [121](#), [150](#), [155](#), [165](#), [168](#)
 __talk_decode_overlay_offset:nNnN
 [121](#), [154](#), [157](#), [166](#)
 __talk_decode_overlays:nN
 [121](#), [124](#), [129](#), [135](#), [172](#)

__talk_decode_overlays:nn
 [87](#), [106](#), [113](#), [121](#), [121](#)
 \l__talk_decode_overlays_bool ..
 [3](#), [6](#), [22](#), [26](#), [42](#), [59](#), [118](#), [123](#)
 \l__talk_decode_overlays_clist .. [10](#)
 \l__talk_decode_overlays_str ...
 [10](#), [28](#), [40](#), [86](#)
 __talk_decode_parse:n . [5](#), [16](#), [16](#), [116](#)
 __talk_decode_parse:w . [16](#), [34](#), [35](#), [46](#)
 __talk_decode_parse_aux:n [16](#), [30](#), [33](#)
 \l__talk_decode_pure_bool
 [7](#), [27](#), [41](#), [91](#), [110](#)
 \l__talk_decode_step_bool
 [8](#), [123](#), [125](#), [141](#)
 \l__talk_fontsize_dim [47](#), [74](#), [79](#)
 \l__talk_footelem_color_tl [181](#)
 \l__talk_footelem_font_tl [181](#)
 \l__talk_footelem_left_skip [181](#)
 \l__talk_footelem_right_skip ... [181](#)
 \l__talk_footer_bg_tl [261](#)
 \l__talk_footer_fg_tl [261](#)
 \l__talk_footer_font_tl [261](#)
 \l__talk_footer_left_skip [261](#)
 \l__talk_footer_order_clist [261](#)
 \l__talk_footer_right_skip [261](#)
 \l__talk_footer_sep_tl [261](#)
 \l__talk_frame_alignment_tl
 [59](#), [87](#), [146](#), [156](#)
 \l__talk_frame_bool [106](#), [374](#), [389](#)
 \g__talk_frame_int
 [14](#), [48](#), [64](#), [211](#), [377](#), [382](#), [388](#)
 __talk_frame_notag:n ... [38](#), [400](#), [400](#)
 __talk_frame_overprint:
 [209](#), [209](#), [221](#), [224](#), [228](#),
 [241](#), [243](#), [244](#), [247](#), [249](#), [256](#), [258](#), [263](#)
 __talk_frame_process:nn
 [386](#), [386](#), [417](#), [426](#), [437](#), [445](#)
 \g__talk_frame_struct_int [56](#), [68](#), [395](#)
 \g__talk_frame_subtitle_tl [3](#), [13](#), [74](#)
 __talk_frame_tag:n [36](#), [392](#), [392](#)
 \g__talk_frame_tag_bool
 [46](#), [375](#), [396](#), [404](#)
 \l__talk_frame_tagging_str
 [17](#), [18](#), [20](#), [22](#), [33](#), [147](#)
 __talk_frame_title:n [15](#), [38](#), [44](#)
 \l__talk_frame_title_bool .. [47](#), [410](#)
 __talk_frame_title_tagged:n ...
 [15](#), [47](#), [51](#)
 \g__talk_frame_title_tl
 [3](#), [8](#), [58](#), [73](#), [250](#), [425](#)
 \l__talk_frame_verb_bool
 [40](#), [376](#), [416](#), [424](#), [436](#), [444](#)
 \l__talk_frametitle_after_skip .
 [25](#), [41](#)

<code>\l__talk_frametitle_before_skip</code>	<code>\l__talk_saved_actions_bool</code>
. 26, 32 99, 121, 137
<code>\l__talk_frametitle_color_tl</code>	<code>\l__talk_saved_overlays_bool</code>
. 27, 34, 35 99, 117, 132
<code>\l__talk_frametitle_font_tl</code> . . 28, 36	<code>__talk_section_tagged:</code>
<code>\l__talk_header_bg_tl</code> 214 119, 119, 338, 349, 365
<code>\l__talk_header_fg_tl</code> 214	<code>\g__talk_section_tl</code> 66
<code>\l__talk_header_font_tl</code> 214	<code>\l__talk_section_tl</code> 66
<code>\l__talk_header_frametitle_bool</code> 214	<code>__talk_slide:nn</code> 9, 9, 390
<code>\l__talk_header_ht_dim</code> 214	<code>__talk_slide_align_bottom:n</code> . 92, 92
<code>\l__talk_header_left_skip</code> 214	<code>__talk_slide_align_center:n</code> . 92, 98
<code>\l__talk_header_right_skip</code> 214	<code>__talk_slide_align_stretch:n</code> 92, 104
<code>__talk_header_tag_begin:n</code>	<code>__talk_slide_align_top:n</code> . . 92, 110
. 53, 125, 166, 166, 173	<code>__talk_slide_aux:n</code> 9, 41, 52
<code>__talk_header_tag_end:</code>	<code>__talk_slide_begin:</code> 32, 69, 69
. 64, 130, 166, 174	<code>\l__talk_slide_box</code> 4, 76, 88
<code>__talk_if_overlay:n</code> 3, 10	<code>\g__talk_slide_continue_bool</code>
<code>__talk_if_overlay:nTF</code> 3, 27, 35, 72, 83, 178, 203, 211, 217
. 3, 7, 12, 13, 13, 22, 30,	<code>__talk_slide_end:</code> 45, 69, 79
31, 33, 96, 143, 152, 175, 301, 320, 335	<code>\g__talk_slide_int</code>
<code>__talk_item_parse_spec:n</code> 5, 8, 25, 29, 196, 202, 208, 210, 215
. 271, 284, 288, 289	<code>\g__talk_subsection_tl</code> 66
<code>__talk_item_parse_spec:w</code>	<code>\l__talk_subsection_tl</code> 66, 101
. 271, 281, 287	<code>\g__talk_subsubsection_tl</code> 66
<code>__talk_label:n</code> 332, 336, 339	<code>\l__talk_subsubsection_tl</code> . . 66, 103
<code>__talk_latex_frame:n</code> . . 26, 385, 385	<code>__talk_textcmd_equiv:n</code> . 282, 303, 307
<code>\l__talk_list_end_tl</code>	<code>\l__talk_titlelem_after_skip</code> 9
. 293, 299, 305, 316, 341	<code>\l__talk_titlelem_before_skip</code> 9
<code>__talk_mode:n</code> 3	<code>\l__talk_titlelem_color_tl</code> 9
<code>__talk_mode:nTF</code> 3, 12	<code>\l__talk_titlelem_font_tl</code> 9
<code>\l__talk_mode_str</code> 7, 47, 58, 83	<code>\l__talk_titlelem_tag_begin_tl</code> . . . 9
<code>\c__talk_modes_clist</code> 48, 55	<code>\l__talk_titlelem_tag_end_tl</code> 9
<code>__talk_onslide:n</code> . 147, 148, 149, 188	<code>\l__talk_titlepage_alignment_tl</code> . 59
<code>\g__talk_onslide_escape_tl</code>	<code>\l__talk_titlepage_framestyle_tl</code> 59
. 156, 159, 161, 165, 171	<code>\l__talk_titlepage_order_clist</code> . . 59
<code>__talk_onslide_reset:</code>	<code>__talk_tmp:w</code> 42, 42, 84, 93
. 147, 153, 166, 170	<code>\l__talk_tmp_box</code> 14,
<code>\g__talk_onslide_tl</code>	24, 39, 43, 60, 74, 85, 193, 203, 225,
. 77, 81, 151, 162, 164, 171	230, 234, 236, 250, 254, 267, 289, 312
<code>__talk_overlay_arg:n</code>	<code>\l__talk_tmp_tl</code>
. 3, 11, 62, 72, 82, 90 12, 18, 21, 23, 44, 299, 301, 302
<code>__talk_overprint_begin:n</code>	<code>__talk_toc_aux:nnnn</code>
. 190, 190, 198, 215 159, 160, 163, 173, 182
<code>__talk_overprint_check_ht:n</code>	<code>__talk_toc_dest:n</code> 159, 186, 189
. 214, 263, 265, 274	<code>__talk_toc_dest:w</code> 159, 191, 194
<code>\l__talk_overprint_int</code> . 208, 212, 218	<code>__talk_toc_level:nnnn</code> . 159, 187, 205
<code>__talk_overprint_save_ht:</code>	<code>\l__talk_uncover_hidden_fp</code> 46
. 214, 219, 239	<code>__talk_wallpaper_hruler:Nnn</code>
<code>\c__talk_paper_height_dim</code> 81 237, 284, 322, 322
<code>\c__talk_paper_width_dim</code> 81	<code>\temporal</code> 173
<code>\c__talk_pause_init_int</code> 67, 71	TeX and L ^A T _E X 2 _ε commands:
<code>\g__talk_pauses_int</code>	<code>\@arabic</code> 6, 7, 78, 79, 80, 124, 379
. . . . 8, 4, 71, 126, 171, 186, 187, 188	<code>\@auxout</code> 254, 343
<code>\l__talk_saved_action_str</code> 99, 119, 135	<code>\@sphack</code> 334

\@contentsline@destination	47, 191, 213, 216, 219, 222
\@currentHref	350
\@currentlabel	347
\@currentlabelname	349
\@currenvir	340
\@definecounter	139
\@endparpenalty	352
\@esphack	337
\@evenfoot	343, 359, 371
\@evenhead	342, 358, 370
\@framenumbers	377
\@ignore	29
\@ignoretrue	89
\@inmatherr	340
\@input	140
\@institute	3
\@itempenalty	319
\@kernel@reserved@label@data	351
\@listI	56
\@listi	49, 56
\@listii	57
\@listiii	64
\@mpfootins	30
\@nobreakfalse	150
\@noitemerr	333
\@oddfoot	341, 343, 353, 359, 369, 371
\@oddhead	336, 342, 347, 358, 363, 370
\@outerparskip	350
\@parboxrestore	77
\@starttoc	135, 156
\@subtitle	3
\c@frame	377
\c@page	124
\c@pauses	4
\c@section	78
\c@slide	5
\c@subsection	79
\c@subsubsection	80
\currentgrouplevel	50
\Gm@bmargin	287
\Gm@lmargin	221, 268, 324
\Gm@rmargin	223, 269, 313
\Gm@tmargin	220
\ignorespaces	29
\l@section	159
\l@subsection	159
\l@subsubsection	159
\on@line	324
\protected@write	343
\ps@plain	334
\ps@talk	334
\ps@wallpaper	334
\std@definecounter	139
tex commands:	
\tex_currentgrouplevel:D	295, 296
\tex_fontdimen:D	61
\tex_hsize:D	10, 21
\tex_setbox:D	8, 19
\tex_textfont:D	61
\tex_vbox:D	8, 19
\tex_vrule:D	27
text commands:	
\text_purify:n	40, 58, 128
\textasteriskcentered	40
\textbf	282
\textbullet	38
\textcolor	6, 11
\textendash	39
\textheight	85
\textit	282
\textmd	282
\textnormal	282
\textperiodcentered	41
\textrm	282
\textsc	282
\textsf	282
\textsl	282
\texttt	282
\textup	282
\textwidth	27, 8, 15, 16, 75, 76, 214
\theenumi	34
\theenumii	35
\theenumiii	36
\theenumiv	37
\theframe	377
\thepage	4, 124, 348
\thepauses	4
\thesection	72
\theslide	5
\thesubsection	72
\thesubsubsection	72
\thispagestyle	94
\tiny	267
tl commands:	
\tl_clear:N	101, 103, 299
\tl_gclear:N	73, 74, 77, 131, 164, 165
\tl_gset:Nn	8, 13, 156, 162, 244, 247, 425
\tl_gset_eq:NN	98
\tl_if_blank:nTF	37, 74, 90, 105, 112, 186, 189, 291
\tl_if_blank_p:n	20
\tl_if_empty:NTF	33, 34, 93, 123, 201, 243, 296, 325
\tl_if_empty:nTF	29, 197
\tl_if_exist:NTF	241
\tl_map_inline:nn	282

