

1 Package header

We load the required packages needed for L^AT_EX3, and the package header.

```
1 <*package>
2 <@@=yoin>
```

Necessary packages: First, L^AT_EX3 stuff.

```
3 \RequirePackage{expl3,l3keys2e,l3regex,xparse}
```

From zref bundle, for computing the total number of pages.

```
4 \RequirePackage{zref-totpages}
```

We need the absolute paths. This also means we need `-recorder` option to pdf_latex.

```
5 \RequirePackage[abspath]{currfile}
```

Package header.

```
6 \ProvidesExplPackage{yoin}{2016/02/28}{v0.0.1}{Joining articles into issues}
```

2 General macros

Macros not necessarily related to the package; moreorless an addition to L^AT_EX3.

`\yoin_seq_gappend_clist:Nn` Globally append `clist` #2 to `seq` #1.

```
7 \cs_new_protected:Nn \yoin_seq_gappend_clist:Nn {
8   \seq_set_from_clist:Nn \l__yoin_tmpa_seq { #2 }
9   \seq_gconcat:NNN #1 #1 \l__yoin_tmpa_seq
10 }
```

(End definition for \yoin_seq_gappend_clist:Nn. This function is documented on page ??.)

`\yoin_keys_set_from_file:nn` Read a file #2 containing a key–value list and set the keys for #1. No checks are done here, nothing like comments could be used, the keys should be separated by a comma (and spaces of course as needed).

```
11 \tl_new:N \l__yoin_keys_tmpa_tl
12 \cs_new_protected:Nn \yoin_keys_set_from_file:nn {
13   \tl_set_from_file:NNn \l__yoin_keys_tmpa_tl { } { #2 }
14   \keys_set:nV { #1 } \l__yoin_keys_tmpa_tl
15 }
16 \cs_generate_variant:Nn \keys_set:nn { nV }
```

(End definition for \yoin_keys_set_from_file:nn. This function is documented on page ??.)

`\yoin_keyval_parse_from_file:nn` Read a file #2 containing a key-value list and set the keys for #1. No checks are done here, nothing like comments could be used, the keys should be separated by a comma (and spaces of course as needed).

```

17 \cs_new_protected:Nn \yoin_keyval_parse_from_file:NNn {
18   \tl_set_from_file:Nnn \l__yoin_keys_tmpa_tl { } { #3 }
19   \keyval_parse:NNV #1 #2 \l__yoin_keys_tmpa_tl
20 }
21 \cs_generate_variant:Nn \keyval_parse:NNn { NNV }

```

(End definition for \yoin_keyval_parse_from_file:nn. This function is documented on page ??.)

3 Key-value interface for the package setup

First, we define the variables to store the keys.

`\g_yoin_subprocess_bool` Booleans:

```

\g_yoin_article_bool 22 \bool_new:N \g_yoin_subprocess_bool
\g_yoin_dryrun_bool   23 \bool_new:N \g_yoin_article_bool
\g_yoin_onlyflags_bool 24 \bool_new:N \g_yoin_dryrun_bool
\g_yoin_onlytags_bool 25 \bool_new:N \g_yoin_onlyflags_bool
                      26 \bool_new:N \g_yoin_onlytags_bool

```

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

`\g_yoin_flags_seq` Sequences for flags, tags and their filtering:

```

\g_yoin_tags_seq      27 \seq_new:N \g_yoin_flags_seq
\g_yoin_onlyflags_seq 28 \seq_new:N \g_yoin_tags_seq
\g_yoin_onlytags_seq  29 \seq_new:N \g_yoin_onlyflags_seq
                      30 \seq_new:N \g_yoin_onlytags_seq

```

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

`msg: unknown-flag` Two messages, for unknown flags and unknown tags.

`msg: unknown-tag`

```

31 \msg_new:nnnn { yoin } { unknown-flag }
32   { The ~ flag ~ '#1' ~ is ~ unknown ~ to ~ 'yoin'. }
33   { You ~ either ~ misspelled ~ it ~ or ~forgot ~ to ~ declare ~ it. }

34 \msg_new:nnnn { yoin } { unknown-tag }
35   { The ~ tag ~ '#1' ~ is ~ unknown ~ to ~ 'yoin'. }
36   { You ~ either ~ misspelled ~ it ~ or ~forgot ~ to ~ declare ~ it. }

```

(End definition for msg: unknown-flag and msg: unknown-tag. These functions are documented on page ??.)

```

\yoin_if_tag_defined:n Conditionals for checking whether a tag/flag was defined.
\yoin_if_flag_defined:n 37 \prg_new_protected_conditional:Nnn \yoin_if_tag_defined:n { T, F, TF } {
38   \seq_if_in:NnTF \g_yoin_tags_seq { #1 } { \prg_return_true: } { \prg_return_false: }
39 }
40 \prg_new_protected_conditional:Nnn \yoin_if_flag_defined:n { T, F, TF } {
41   \seq_if_in:NnTF \g_yoin_flags_seq { #1 } { \prg_return_true: } { \prg_return_false: }
42 }
43 \cs_new_protected:Nn \__yoin_error_if_tag_undefined:n {
44   \yoin_if_tag_defined:nF { #1 } { \msg_error:nnn { yoin } { unknown-tag } { #1 } }
45 }
46 \cs_new_protected:Nn \__yoin_error_if_flag_undefined:n {
47   \yoin_if_flag_defined:nF { #1 } { \msg_error:nnn { yoin } { unknown-flag } { #1 } }
48 }

```

(End definition for \yoin_if_tag_defined:n and \yoin_if_flag_defined:n. These functions are documented on page ??.)

yoin / general The keys themselves:

```
49 \keys_define:nn { yoin / general } {
```

Booleans:

```

50   dryrun .bool_gset:N = \g_yoin_dryrun_bool,
51   dryrun .initial:n = { false },

52   article .bool_gset:N = \g_yoin_article_bool,
53   article .initial:n = { false },

54   subprocess .bool_gset:N = \g_yoin_subprocess_bool,
55   subprocess .initial:n = { false },

```

Keys whose clist values are appended to a seq:

```

56   defineflags .code:n = \yoin_seq_gappend_clist:Nn \g_yoin_flags_seq { #1 },
57   definetags .code:n = \yoin_seq_gappend_clist:Nn \g_yoin_tags_seq { #1 },

```

A clist key is stored in a seq, also, a corresponding bool is set true. (The point is, if onlyflags/onlytags is not ever set up, we want to know it since we treat it as if we use all flags/tags.)

```

58   onlyflags .code:n =
59     \seq_gset_from_clist:Nn \g_yoin_onlyflags_seq { #1 }
60     \bool_gset_true:N \g_yoin_onlyflags_bool
61     ,

```

```

62   onlytags .code:n =
63     \seq_gset_from_clist:Nn \g_yoin_onlytags_seq { #1 }
64     \bool_gset_true:N \g_yoin_onlytags_bool
65     ,
66 }

```

(End definition for `yoin / general`. This function is documented on page ??.)

`\ProcessKeysPackageOptions` Process key options given to the package. We do not want to process any options given to the class. Whence `\ProcessKeysPackageOptions` and not `\ProcessKeysOptions`.

```

67 \ProcessKeysPackageOptions { yoin / general }

```

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

`\yoin_setup:n` Allow keys to be set later. We define both a \LaTeX 3 interface and an xparse UI wrapper.

```

\yoinSetup
68 \cs_new_protected:Nn \yoin_setup:n {
69   \keys_set:nn { yoin / general } { #1 }
70 }

71 \NewDocumentCommand \yoinSetup { R[]{} } {
72   \yoin_setup:n { #1 }
73 }

```

(End definition for `\yoin_setup:n` and `\yoinSetup`. These functions are documented on page ??.)

4 `\yoinAdd` macro — adding articles to the issue

The key–value interface. In this case, we basically only store the keys for each article in a prop. First, an interface for setting the keys for the articles. `\yoin_yoinadd_prop:n` returns the name of the prop for the given article; *no check for existence is done at this place*.

`\g_yoin_yoinadd_seq` A sequence for storing the list of the existing articles.

```

74 \seq_new:N \g_yoin_yoinadd_seq

```

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

`\yoin_yoinadd_prop:n` `\yoin_yoinadd_prop:V`
`\yoin_yoinadd_prop_item:nn`
`\yoin_yoinadd_prop_item:Vn`

`\yoin_yoinadd_prop:n` returns the name of the prop for the given article; *no check for existence is done at this place*. `\yoin_yoinadd_prop:nn` returns property #2 of article #1, or `\q_no_value` if the property is not set.

```

75 \cs_new:Nn \yoin_yoinadd_prop:n {
76   g__yoin_article_#1_prop
77 }
78 \cs_generate_variant:Nn \yoin_yoinadd_prop:n { V }

79 \cs_new:Nn \yoin_yoinadd_prop_item:nn {
80   \prop_item:cn { \yoin_yoinadd_prop:n { #1 } } { #2 }
81 }
82 \cs_generate_variant:Nn \yoin_yoinadd_prop_item:nn { V }

```

(End definition for `\yoin_yoinadd_prop:n` and others. These functions are documented on page ??.)

For processing `\yoinAdd`, we first set up a `tl` to contain the name of the article, then create the prop, and finally use `l3keys` to fill in the prop. Note that if an article is added twice, an error is issued, if the error is ignored, the article is not added but the properties are set.

`\l_yoin_yoinadd_currentarticle_tl` A `tl` that stores the name of the article that is being processed by `\yoinAdd`.

```
83 \tl_new:N \l_yoin_yoinadd_currentarticle_tl
```

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

`__yoin_yoinadd_storekey:nn` Internal macro for storing a key in the prop. The one-parameter variant sets the value of the key empty.

```

\__yoin_yoinadd_storekey:n
84 \cs_new_protected:Nn \__yoin_yoinadd_storekey:nn {
85   \prop_gput:cn { \yoin_yoinadd_prop:V \l_yoin_yoinadd_currentarticle_tl } { #1 } { #2 }
86 }
87 \cs_new_protected:Nn \__yoin_yoinadd_storekey:n {
88   \prop_gput:cn { \yoin_yoinadd_prop:V \l_yoin_yoinadd_currentarticle_tl } { #1 } { }
89 }

```

(End definition for `__yoin_yoinadd_storekey:nn` and `__yoin_yoinadd_storekey:n`. These functions are documented on page ??.)

`\yoin_yoinadd:nn`
`\yoinAdd`

The macro `\yoinAdd` itself. We first set `\l_@@_yoinadd_currentarticle_tl`, then check whether the same article has not been processed before (issuing an error in that case and finishing). Then, the article is added in `\g_yoin_yoinadd_seq`, the prop created, the article's name added in the prop with key `article` and the keys are set. If the article has a `.yoin` file in its sub-directory, the key-values in it is added to the prop. If the file does not exist, it means things are wrong (the article should first be set up, before being added to its issue by `\yoinAdd`).

```

90 \cs_new_protected:Nn \yoin_yoinadd:nn {
91   \tl_set:Nn \l_yoin_yoinadd_currentarticle_tl { #1 }
92   \seq_if_in:NnTF \g_yoin_yoinadd_seq { #1 } {
93     \msg_error:nnn { yoin } { yoinadd-duplicatearticle } { #1 }

```

```

94   } {
95     \seq_gput_right:Nn \g_yoin_yoinadd_seq { #1 }
96     \prop_new:c { \yoin_yoinadd_prop:n { #1 } }
97     \__yoin_yoinadd_storekey:nn { article } { #1 }
98     \keys_set:nn { yoin / yoinadd } { #2 }
99     \file_if_exist:nTF { #1 / #1 .yoin } {
100       \yoin_keyval_parse_from_file:NNn
101       \__yoin_yoinadd_storekey:n
102       \__yoin_yoinadd_storekey:nn
103       { #1 / #1 .yoin }
104     } {
105       \msg_error:nnn { yoin } { yoinadd-dotyoinmissing } { #1 }
106     }
107   }
108 }

109 \NewDocumentCommand \yoinAdd { m O{} } {
110   \yoin_yoinadd:nn { #1 } { #2 }
111 }

```

(End definition for \yoin_yoinadd:nn and \yoinAdd. These functions are documented on page ??.)

`msg: yoinadd-duplicatearticle` The error messages: for adding a duplicate article and for adding an article with no #1/#1.yoin file.

`msg: yoinadd-dotyoinmissing`

```

112 \msg_new:nnn { yoin } { yoinadd-duplicatearticle }
113   { The ~ article ~ "#1" ~ has ~ been ~ already ~ processed ~ by ~ \token_to_str:N \yoinAdd ~.}
114 \msg_new:nnn { yoin } { yoinadd-dotyoinmissing }
115   { The ~ article ~ "#1" ~ has ~ no ~ file ~ "#1/#1.yoin" ~ and ~ was ~ not ~ properly ~ set ~ up.}

```

(End definition for msg: yoinadd-duplicatearticle and msg: yoinadd-dotyoinmissing. These functions are documented on page ??.)

`yoin / yoinadd` The keys here are pretty simple; each defined key just stores its value in the prop. We recall that #1 is the key and ##1 is the value.

```

116 \clist_map_inline:nn { forceopenany, forceopenright, ignore } {
117   \keys_define:nn { yoin / yoinadd } {
118     #1 .code:n = \__yoin_yoinadd_storekey:nn { #1 } { ##1 },
119   }
120 }

```

However, for the tag key, we additionally check that the tag exists.

```

121 \keys_define:nn { yoin / yoinadd } {
122   tag .code:n =
123     \__yoin_error_if_tag_undefined:n { #1 }

```

```

124     \_yoin_yoinadd_storekey:nn { tag } { #1 }
125     ,
126 }

```

(End definition for yoin / yoinadd. This function is documented on page ??.)

5 Environment yoinshell

`\l_yoin_yoinshell_ignore_bool` A boolean for storing the ignore key's value.

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

`yoin / yoinshell` Key-value interface to yoinshell.

```

127 \keys_define:nn { yoin / yoinshell } {
    If flag is set and onlyflags is set but the flag is not amongst them, the whole yoinshell is ignored (by setting the ignore key).
128     flag .code:n =
129     \_yoin_error_if_flag_undefined:n { #1 }
130     \bool_if:NT \g_yoin_onlyflags_bool {
131         \seq_if_in:NnF \g_yoin_onlyflags_seq { #1 } {
132             \keys_set:nn { yoin / yoinshell } {
133                 ignore = true
134             }
135         }
136     }
137     ,
    The ignore key sets a boolean
138     ignore .bool_set:N = \l_yoin_yoinshell_ignore_bool,
139     ignore .initial:n = { false },
140 }

```

(End definition for yoin / yoinshell. This function is documented on page ??.)

`shellesc.sty` A reasonable shell escape that should work in both pdf_latex and lua_latex in T_EX Live 2016.

```

\ShellEscape
\__yoin_yoinshell_shellescape:n
141 \file_if_exist:nTF { shellesc.sty } {
142     \RequirePackage { shellesc }
143 } {
144     \def \ShellEscape #1 { \immediate \write 18 { #1 } }
145 }

```

```

146 \cs_new_protected:Nn \__yoin_yoinshell_shellescape:n {
147   \ShellEscape { #1 }
148 }

```

(End definition for shellesc.sty, \ShellEscape, and __yoin_yoinshell_shellescape:n. These functions are documented on page ??.)

`__yoin_yoinshell_begin:n` Environment yoinshell (one key-value argument). We perform some local definitions that should stay local, so we put everything in a group. The keys are set. Then we define the macros — “shell commands”. If ignore is set, these macros are declared to do nothing, otherwise they are simply wrappers to the L^AT_EX3 counterparts.

```

149 \cs_new_protected:Nn \__yoin_yoinshell_begin:n {
150   \group_begin:
151   \keys_set:nn { yoin / yoinshell } { #1 }
152   \bool_if:NTF \l_yoin_yoinshell_ignore_bool {
153     \DeclareDocumentCommand \RunForEach { 0{} m } { }
154     \DeclareDocumentCommand \Run { 0{} m } { }
155   } {
156     \DeclareDocumentCommand \RunForEach { 0{} m } { \yoin_yoinshell_runforeach:nn { ##1 } { ##2 } }
157     \DeclareDocumentCommand \Run { 0{} m } { \yoin_yoinshell_run:nn { ##1 } { ##2 } }
158   }
159 }

160 \cs_new_protected:Nn \__yoin_yoinshell_end: {
161   \group_end:
162 }

163 \NewDocumentEnvironment { yoinshell } { 0{} } {
164   \__yoin_yoinshell_begin:n { #1 }
165 } {
166   \__yoin_yoinshell_end:
167 }

```

(End definition for __yoin_yoinshell_begin:n, __yoin_yoinshell_end:, and {yoinshell}. These functions are documented on page ??.)

The yoinshell command \RunForEach.

`\l_yoin_yoinshell_runforarticle_tag_tl` First, two t_ls that will store tags: One for the tag of the article, one that could be passed to \RunForEach that is initially set to `\q_no_value`.

```

168 \tl_new:N \l_yoin_yoinshell_runforarticle_tag_tl
169 \tl_new:N \l_yoin_yoinshell_runforeach_onlytag_tl
170 \tl_set:Nn \l_yoin_yoinshell_runforeach_onlytag_tl { \q_no_value }

```

(End definition for \l_yoin_yoinshell_runforarticle_tag_tl and \l_yoin_yoinshell_runforeach_onlytag_tl. These functions are documented on page ??.)

`yoin / runforeach` So far, the only key-val passable to `\RunForEach` is `onlytag`, which tests for the tag to be declared and passes it to `\l_@@_yoinshell_-runforeach_onlytag_tl`.

```

171 \keys_define:nn { yoin / runforeach } {
172   onlytag .code:n =
173     \__yoin_error_if_tag_undefined:n { #1 }
174     \tl_set:Nn \l__yoin_yoinshell_runforeach_onlytag_tl { #1 }
175   ,
176 }

```

(End definition for yoin / runforeach. This function is documented on page ??.)

`__yoin_yoinshell_runforarticle_keyfromprop:nnN` This macro lets #3 to the value of property #2 of article #1. It makes it an empty definition if the property is unset.

```

177 \tl_new:N \__yoin_yoinshell_runforarticle_tmpa_tl
178 \cs_new_protected:Nn \__yoin_yoinshell_runforarticle_keyfromprop:nnN {
179   \prop_get:cnN { \yoin_yoinadd_prop:n { #1 } } { #2 } \l__yoin_yoinshell_runforarticle_tmpa_tl
180   \quark_if_no_value:NTF \l__yoin_yoinshell_runforarticle_tmpa_tl {
181     \def #3 {}
182   } {
183     \let #3 \l__yoin_yoinshell_runforarticle_tmpa_tl
184   }
185 }

```

(End definition for __yoin_yoinshell_runforarticle_keyfromprop:nnN. This function is documented on page ??.)

`__yoin_yoinshell_runforeach:nn` `\RunForEach` itself just sets the keys (in a group to make things local) and then calls `\@@_yoinshell_runforarticle:nn` on each article.

```

186 \cs_new_protected:Nn \yoin_yoinshell_runforeach:nn {
187   \group_begin:
188   \keys_set:nn { yoin / runforeach } { #1 }
189   \seq_map_inline:Nn \g_yoin_yoinadd_seq { \__yoin_yoinshell_runforarticle:nn { ##1 } { #2 } }
190   \group_end:
191 }

```

(End definition for __yoin_yoinshell_runforeach:nn. This function is documented on page ??.)

`__yoin_yoinshell_runforarticle:nn` If the tag passed to `onlytag` of `\RunForEach` is identical to the tag of the article or if any of them is not set, we do what should be done, otherwise nothing is done (the tags do not match). We only extract the prop to publically available macros like `\Article`, `\Jobname` etc. (in a group to make this local), and then run the command in shell escape.

```

192 \cs_new_protected:Nn \__yoin_yoinshell_runforarticle:nn {
193   \prop_get:cnN { \yoin_yoinadd_prop:n { #1 } } { tag } \l__yoin_yoinshell_runforarticle_tag_tl
194   \bool_if:nT {

```

```

195     \quark_if_no_value_p:N \l__yoin_yoinshell_runforarticle_tag_tl
196     ||
197     \quark_if_no_value_p:N \l__yoin_yoinshell_runforeach_onlytag_tl
198     ||
199     \tl_if_eq_p:NN \l__yoin_yoinshell_runforeach_onlytag_tl \l__yoin_yoinshell_runforarticle_tag_tl
200   }{
201     \group_begin:
202     \__yoin_yoinshell_runforarticle_keyfromprop:nNn { #1 } { article } \Article
203     \__yoin_yoinshell_runforarticle_keyfromprop:nNn { #1 } { jobname } \Jobname
204     \__yoin_yoinshell_shellescape:n { #2 }
205     \group_end:
206   }
207 }

```

(End definition for __yoin_yoinshell_runforarticle:nn. This function is documented on page ??.)

6 Article setting stuff (undocumented)

Information to be stored in an auxiliary file.

```

208 \cs_new_protected:Nn \__yoin_article_write:n {
209   \immediate \write \@auxout { \token_to_str:N \@writefile { yoin } { #1 } }
210 }
211
212 \cs_new_protected:Nn \__yoin_article_write_keyval:nn {
213   \__yoin_article_write:n { #1 ~ = ~ #2 , }
214 }
215 \cs_generate_variant:Nn \__yoin_article_write_keyval:nn { nx }
216
217 \cs_new_protected:Nn \yoin_article_write_meta:nn {
218   \__yoin_article_write_keyval:nn { meta-#1 } { #2 }
219 }
220
221 \cs_new_protected:Nn \yoin_article_writekeys: {
222   \__yoin_article_write_keyval:nx { jobname } { \jobname }
223   \__yoin_article_write_keyval:nx { totpages } { \ztotpages }
224   \__yoin_article_write_keyval:nx { currdir } { \l_yoin_article_currdir_tl }
225   \__yoin_article_write_keyval:nx { firstpage } { \int_use:N \l_yoin_article_firstpage_int }
226 }
227

```

```

228 \prop_new:N \l__yoin_article_readkeys_prop
229
230 \cs_new_protected:Nn \yoin_article_set_readkey:nn {
231   \prop_put:Nnn \l__yoin_article_readkeys_prop { #1 } { #2 }
232 }
233
234 \int_new:N \l_yoin_article_firstpage_int
235 \int_set:Nn \l_yoin_article_firstpage_int { 1 }
236
237 \keys_define:nn { yoin / toarticle } {
238   firstpage .code:n =
239     \int_set:Nn \l_yoin_article_firstpage_int { #1 }
240     \yoin_article_set_readkey:nn { firstpage } { #1 }
241   ,
242
243   unknown .code:n =
244     \yoin_article_set_readkey:nn { \l_keys_key_tl } { #1 }
245   ,
246 }
247
248 \bool_new:N \g__yoin_article_readkeys_bool
249 \bool_gset_true:N \g__yoin_article_readkeys_bool
250
251 \cs_new_protected:Nn \yoin_article_readkeys: {
252   \bool_if:NT \g__yoin_article_readkeys_bool {
253     \file_if_exist:nT { ../ \l_yoin_article_currdir_tl .yoin } {
254       \yoin_keys_set_from_file:nn { yoin / toarticle } { ../ \l_yoin_article_currdir_tl .yoin }
255     }
256   }
257   \bool_gset_false:N \g__yoin_article_readkeys_bool
258 }
259
260 \tl_new:N \l__yoin_article_tmpa_tl
261 \seq_new:N \l__yoin_article_tmpa_seq
262 \tl_new:N \l_yoin_article_currdir_tl
263 \cs_new_protected:Nn \yoin_article_getcurrdir:N {
264   \tl_set:Nx \l__yoin_article_tmpa_tl { \currfileabsdir }
265   \cs_generate_variant:Nn \regex_extract_once:nnNF { nV }
266   \regex_extract_once:nVNF { /([~/]+)/\Z } \l__yoin_article_tmpa_tl \l__yoin_article_tmpa_seq { \error }
267   \seq_get_right:NN \l__yoin_article_tmpa_seq #1

```

```

268 }
269
270 \AtBeginDocument{ \yoin_atbegindocument: }
271
272 \cs_new_protected:Nn \yoin_atbegindocument: {
273   \expandafter \newwrite \csname tf@yoin\endcsname
274   \bool_if:NTF \g_yoin_article_bool {
275     \yoin_article_getcurrdir:N \l_yoin_article_currdir_tl
276     \immediate \openout \csname tf@yoin\endcsname \l_yoin_article_currdir_tl .yoin\relax
277     \yoin_article_readkeys:
278     \setcounter { page } { \l_yoin_article_firstpage_int }
279     \yoin_article_writekeys:
280   } {
281     \immediate \openout \csname tf@yoin\endcsname \jobname .yoin\relax
282   }
283 }

```

7 yoinProcess (undocumented)

```

284 \int_new:N \g_yoin_page_int
285 \iow_new:N \g__yoin_yoinprocess_stream
286 \cs_new_protected:Nn \yoin_yoinprocess:n {
287   \keys_set:nn { yoin / yoinprocess } { #1 }
288   \seq_map_inline:Nn \g_yoin_yoinadd_seq {
289     \cleardoublepage
290     \int_gset:Nn \g_yoin_page_int { \value { page } }
291     \includepdf [ pages = - ] { ##1 / \yoin_yoinadd_prop_item:nn { ##1 } { jobname } .pdf }
292     \iow_open:Nn \g__yoin_yoinprocess_stream { ##1 .yoin }
293     \iow_now:Nx \g__yoin_yoinprocess_stream { firstpage ~ = ~ \int_use:N \g_yoin_page_int , }
294     \iow_close:N \g__yoin_yoinprocess_stream
295     \int_gadd:Nn \g_yoin_page_int { \yoin_yoinadd_prop_item:nn { ##1 } { totpages } }
296   }
297 }
298 \DeclareDocumentCommand \yoinProcess { O{} } { \yoin_yoinprocess:n { #1 } }

```

8 Experimental

\bla

```

299 \cs_new:Nn \yoin_bla: {

```

```
300     Blabla
301 }
302
    (End definition for \bla. This function is documented on page ??.)
303  $\langle$ /package $\rangle$ 
```