

1 Package header

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

Necessary packages: First, \LaTeX 3 stuff.

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

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

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

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

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

For including PDF files.

```
6 \RequirePackage{pdfpages}
```

Package header.

```
7 \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 \LaTeX 3.

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

```
8 \cs_new_protected:Nn \yoin_seq_gappend_clist:Nn {
9   \seq_set_from_clist:Nn \l__yoin_tmpa_seq { #2 }
10  \seq_gconcat:NNN #1 #1 \l__yoin_tmpa_seq
11 }
```

(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).

```
12 \tl_new:N \l__yoin_keys_tmpa_tl
13 \cs_new_protected:Nn \yoin_keys_set_from_file:nn {
14   \tl_set_from_file:Nnn \l__yoin_keys_tmpa_tl { } { #2 }
15   \keys_set:nV { #1 } \l__yoin_keys_tmpa_tl
16 }
17 \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).

```
18 \cs_new_protected:Nn \yoin_keyval_parse_from_file:NNn {
19   \tl_set_from_file:Nnn \l__yoin_keys_tmpa_tl { } { #3 }
20   \keyval_parse:NNV #1 #2 \l__yoin_keys_tmpa_tl
21 }
22 \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 23 \bool_new:N \g_yoin_subprocess_bool
\g_yoin_dryrun_bool   24 \bool_new:N \g_yoin_article_bool
\g_yoin_onlyflags_bool 25 \bool_new:N \g_yoin_dryrun_bool
\g_yoin_onlytags_bool 26 \bool_new:N \g_yoin_onlyflags_bool
                      27 \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      28 \seq_new:N \g_yoin_flags_seq
\g_yoin_onlyflags_seq 29 \seq_new:N \g_yoin_tags_seq
\g_yoin_onlytags_seq  30 \seq_new:N \g_yoin_onlyflags_seq
                      31 \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 32 \msg_new:nnnn { yoin } { unknown-flag }
                  33   { The ~ flag ~ '#1' ~ is ~ unknown ~ to ~ 'yoin'. }
                  34   { You ~ either ~ misspelled ~ it ~ or ~forgot ~ to ~ declare ~ it. }

                  35 \msg_new:nnnn { yoin } { unknown-tag }
                  36   { The ~ tag ~ '#1' ~ is ~ unknown ~ to ~ 'yoin'. }
                  37   { 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 38 \prg_new_protected_conditional:Nnn \yoin_if_tag_defined:n { T, F, TF } {
39   \seq_if_in:NnTF \g_yoin_tags_seq { #1 } { \prg_return_true: } { \prg_return_false: }
40 }
41 \prg_new_protected_conditional:Nnn \yoin_if_flag_defined:n { T, F, TF } {
42   \seq_if_in:NnTF \g_yoin_flags_seq { #1 } { \prg_return_true: } { \prg_return_false: }
43 }

```

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

```

\_yoin_error_if_tag_undefined:n Check whether a tag/flag is defined, if not, issue an error.
\_yoin_error_if_flag_undefined:n 44 \cs_new_protected:Nn \_yoin_error_if_tag_undefined:n {
45   \yoin_if_tag_defined:nF { #1 } { \msg_error:nnn { yoin } { unknown-tag } { #1 } }
46 }
47 \cs_new_protected:Nn \_yoin_error_if_flag_undefined:n {
48   \yoin_if_flag_defined:nF { #1 } { \msg_error:nnn { yoin } { unknown-flag } { #1 } }
49 }

```

(End definition for _yoin_error_if_tag_undefined:n and _yoin_error_if_flag_undefined:n. These functions are documented on page ??.)

yoin / general The keys themselves:

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

Booleans:

```

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

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

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

```

Keys whose clist values are appended to a seq:

```

57   defineflags .code:n = \yoin_seq_gappend_clist:Nn \g_yoin_flags_seq { #1 },
58   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.)

```

59   onlyflags .code:n =
60     \seq_gset_from_clist:Nn \g_yoin_onlyflags_seq { #1 }
61     \bool_gset_true:N \g_yoin_onlyflags_bool
62   ,

```

```

63   onlytags .code:n =
64     \seq_gset_from_clist:Nn \g_yoin_onlytags_seq { #1 }
65     \bool_gset_true:N \g_yoin_onlytags_bool
66     ,
67 }

```

(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`.

```

68 \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
69 \cs_new_protected:Nn \yoin_setup:n {
70   \keys_set:nn { yoin / general } { #1 }
71 }

72 \NewDocumentCommand \yoinSetup { R[]{} } {
73   \yoin_setup:n { #1 }
74 }

```

(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.

```

75 \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.

```

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

80 \cs_new:Nn \yoin_yoinadd_prop_item:nn {
81   \prop_item:cn { \yoin_yoinadd_prop:n { #1 } } { #2 }
82 }
83 \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`.

```
84 \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
85 \cs_new_protected:Nn \__yoin_yoinadd_storekey:nn {
86   \prop_gput:cn { \yoin_yoinadd_prop:V \l_yoin_yoinadd_currentarticle_tl } { #1 } { #2 }
87 }
88 \cs_new_protected:Nn \__yoin_yoinadd_storekey:n {
89   \prop_gput:cn { \yoin_yoinadd_prop:V \l_yoin_yoinadd_currentarticle_tl } { #1 } { }
90 }

```

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

`\yoin_yoinadd:nn` 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`).

```

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

```

```

95   } {
96     \seq_gput_right:Nn \g_yoin_yoinadd_seq { #1 }
97     \prop_new:c { \yoin_yoinadd_prop:n { #1 } }
98     \__yoin_yoinadd_storekey:nn { article } { #1 }
99     \keys_set:nn { yoin / yoinadd } { #2 }
100    \file_if_exist:nTF { #1 / #1 .yoin } {
101      \yoin_keyval_parse_from_file:NNn
102      \__yoin_yoinadd_storekey:n
103      \__yoin_yoinadd_storekey:nn
104      { #1 / #1 .yoin }
105    } {
106      \msg_error:nnn { yoin } { yoinadd-dotyoinmissing } { #1 }
107    }
108  }
109 }

110 \NewDocumentCommand \yoinAdd { m O{} } {
111   \yoin_yoinadd:nn { #1 } { #2 }
112 }

```

(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 113 \msg_new:nnn { yoin } { yoinadd-duplicatearticle }
114   { The ~ article ~ "#1" ~ has ~ been ~ already ~ processed ~ by ~ \token_to_str:N \yoinAdd ~.}
115 \msg_new:nnn { yoin } { yoinadd-dotyoinmissing }
116   { 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.

```

117 \clist_map_inline:nn { forceopenany, forceopenright, ignore } {
118   \keys_define:nn { yoin / yoinadd } {
119     #1 .code:n = \__yoin_yoinadd_storekey:nn { #1 } { ##1 },
120   }
121 }

```

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

```

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

```

```

125     \_yoin_yoinadd_storekey:nn { tag } { #1 }
126     ,
127 }

```

(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.

```

128 \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).
129     flag .code:n =
130     \_yoin_error_if_flag_undefined:n { #1 }
131     \bool_if:NT \g_yoin_onlyflags_bool {
132         \seq_if_in:NnF \g_yoin_onlyflags_seq { #1 } {
133             \keys_set:nn { yoin / yoinshell } {
134                 ignore = true
135             }
136         }
137     }
138     ,
    The ignore key sets a boolean
139     ignore .bool_set:N = \l_yoin_yoinshell_ignore_bool,
140     ignore .initial:n = { false },
141 }

```

(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
142 \file_if_exist:nTF { shellesc.sty } {
143     \RequirePackage { shellesc }
144 } {
145     \def \ShellEscape #1 { \immediate \write 18 { #1 } }
146 }

```

```

147 \cs_new_protected:Nn \__yoin_yoinshell_shellescape:n {
148   \ShellEscape { #1 }
149 }

```

(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 \LaTeX 3 counterparts.

```

150 \cs_new_protected:Nn \__yoin_yoinshell_begin:n {
151   \group_begin:
152   \keys_set:nn { yoin / yoinshell } { #1 }
153   \bool_if:NTF \l_yoin_yoinshell_ignore_bool {
154     \DeclareDocumentCommand \RunForEach { 0{} m } { }
155     \DeclareDocumentCommand \Run { 0{} m } { }
156   } {
157     \DeclareDocumentCommand \RunForEach { 0{} m } { \yoin_yoinshell_runforeach:nn { ##1 } { ##2 } }
158     \DeclareDocumentCommand \Run { 0{} m } { \yoin_yoinshell_run:nn { ##1 } { ##2 } }
159   }
160 }

161 \cs_new_protected:Nn \__yoin_yoinshell_end: {
162   \group_end:
163 }

164 \NewDocumentEnvironment { yoinshell } { 0{} } {
165   \__yoin_yoinshell_begin:n { #1 }
166 } {
167   \__yoin_yoinshell_end:
168 }

```

(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 `tl`s 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`.

```

169 \tl_new:N \l_yoin_yoinshell_runforarticle_tag_tl
170 \tl_new:N \l_yoin_yoinshell_runforeach_onlytag_tl
171 \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`.

```

172 \keys_define:nn { yoin / runforeach } {
173   onlytag .code:n =
174     \__yoin_error_if_tag_undefined:n { #1 }
175     \tl_set:Nn \l__yoin_yoinshell_runforeach_onlytag_tl { #1 }
176   ,
177 }

```

(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.

```

178 \tl_new:N \__yoin_yoinshell_runforarticle_tmpa_tl
179 \cs_new_protected:Nn \__yoin_yoinshell_runforarticle_keyfromprop:nnN {
180   \prop_get:cnN { \yoin_yoinadd_prop:n { #1 } } { #2 } \l__yoin_yoinshell_runforarticle_tmpa_tl
181   \quark_if_no_value:NTF \l__yoin_yoinshell_runforarticle_tmpa_tl {
182     \def #3 {}
183   } {
184     \let #3 \l__yoin_yoinshell_runforarticle_tmpa_tl
185   }
186 }

```

(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.

```

187 \cs_new_protected:Nn \yoin_yoinshell_runforeach:nn {
188   \group_begin:
189   \keys_set:nn { yoin / runforeach } { #1 }
190   \seq_map_inline:Nn \g_yoin_yoinadd_seq { \__yoin_yoinshell_runforarticle:nn { ##1 } { #2 } }
191   \group_end:
192 }

```

(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.

```

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

```

```

196     \quark_if_no_value_p:N \l__yoin_yoinshell_runforarticle_tag_tl
197     ||
198     \quark_if_no_value_p:N \l__yoin_yoinshell_runforeach_onlytag_tl
199     ||
200     \tl_if_eq_p:NN \l__yoin_yoinshell_runforeach_onlytag_tl \l__yoin_yoinshell_runforarticle_tag_tl
201   }{
202     \group_begin:
203     \__yoin_yoinshell_runforarticle_keyfromprop:nnN { #1 } { article } \Article
204     \__yoin_yoinshell_runforarticle_keyfromprop:nnN { #1 } { jobname } \Jobname
205     \__yoin_yoinshell_shellescape:n { #2 }
206     \group_end:
207   }
208 }

```

(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.

```

209 \cs_new_protected:Nn \__yoin_article_write:n {
210   \immediate \write \@auxout { \token_to_str:N \@writefile { yoin } { #1 } }
211 }
212
213 \cs_new_protected:Nn \__yoin_article_write_keyval:nn {
214   \__yoin_article_write:n { #1 ~ = ~ #2 , }
215 }
216 \cs_generate_variant:Nn \__yoin_article_write_keyval:nn { nx }
217
218 \cs_new_protected:Nn \yoin_article_write_meta:nn {
219   \__yoin_article_write_keyval:nn { meta-#1 } { #2 }
220 }
221
222 \cs_new_protected:Nn \yoin_article_writekeys: {
223   \__yoin_article_write_keyval:nx { jobname } { \jobname }
224   \__yoin_article_write_keyval:nx { totpages } { \ztotpages }
225   \__yoin_article_write_keyval:nx { currrdir } { \l_yoin_article_currrdir_tl }
226   \__yoin_article_write_keyval:nx { firstpage } { \int_use:N \l_yoin_article_firstpage_int }
227 }
228

```

```

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

```

```

269 }
270
271 \AtBeginDocument{ \yoin_atbegindocument: }
272
273 \cs_new_protected:Nn \yoin_atbegindocument: {
274   \expandafter \newwrite \csname tf@yoin\endcsname
275   \bool_if:NTF \g_yoin_article_bool {
276     \yoin_article_getcurrdir:N \l_yoin_article_currdir_tl
277     \immediate \openout \csname tf@yoin\endcsname \l_yoin_article_currdir_tl .yoin\relax
278     \yoin_article_readkeys:
279     \setcounter { page } { \l_yoin_article_firstpage_int }
280     \yoin_article_writekeys:
281   } {
282     \immediate \openout \csname tf@yoin\endcsname \jobname .yoin\relax
283   }
284 }

```

7 yoinProcess (undocumented)

```

285 \int_new:N \g_yoin_page_int
286 \iow_new:N \g__yoin_yoinprocess_stream
287 \bool_new:N \g__yoin_yoinprocess_output_bool
288 \bool_set_true:N \g__yoin_yoinprocess_output_bool
289 \cs_new_protected:Nn \yoin_yoinprocess:n {
290   \keys_set:nn { yoin / yoinprocess } { #1 }
291   \clearpage
292   \int_gset:Nn \g_yoin_page_int { \value { page } }
293   \seq_map_inline:Nn \g_yoin_yoinadd_seq {
294     \__yoin_yoinprocess_cleardoublepage:
295     \bool_if:NT \g__yoin_yoinprocess_output_bool {
296       \includepdf [ pages = - ] { ##1 / \yoin_yoinadd_prop_item:nn { ##1 } { jobname } .pdf }
297     }
298     \iow_open:Nn \g__yoin_yoinprocess_stream { ##1 .yoin }
299     \iow_now:Nx \g__yoin_yoinprocess_stream { firstpage ~ = ~ \int_use:N \g_yoin_page_int , }
300     \iow_now:Nx \g__yoin_yoinprocess_stream { parent ~ = ~ \jobname , }
301     \iow_close:N \g__yoin_yoinprocess_stream
302     \int_gadd:Nn \g_yoin_page_int { \yoin_yoinadd_prop_item:nn { ##1 } { totpages } }
303   }
304 }

```

```

305 \DeclareDocumentCommand \yoinProcess { 0{ } } { \yoin_yoinprocess:n { #1 } }
306
307 \cs_new_protected:Nn \__yoin_yoinprocess_cleardoublepage: {
308   \bool_if:NT \g__yoin_yoinprocess_output_bool { \cleardoublepage }
309   \int_if_even:nT { \g_yoin_page_int } { \int_gincr:N \g_yoin_page_int }
310 }
311

```

8 Experimental

\bla

```

312 \cs_new:Nn \yoin_blabla: {
313   Blabla
314 }
315

```

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

```

316 \</package>

```