

declpkg — Including packages with dependencies *

Ting Wei Liu[†]

Released 2020-05-10

Abstract

The declpkg package includes packages with user provided order.

Contents

1	Installation	2
2	Documentation	2
3	Implementation	3
3.1	Lambdas	3
3.2	Internal helpers	4
3.3	Variables and constants	5
3.4	Errors	6
3.5	Internal interfaces	7
3.6	Internals	10
3.7	Public interfaces	13
	Change History	18
	Index	19

*This file describes version v1.1, last revised 2020-05-10.

[†]E-mail: tingwei890@gmail.com

1 Installation

The package is supplied in `dtx` format. To unpack the `dtx`, running `tex declpkg.dtx` will extract the package and `latex declpkg.dtx` will extract it and also typeset the documentation.

The package requires LaTeX3 support.

2 Documentation

<code>\declpkg</code>	<code>\declpkg [<i>options</i>] {<i>package</i>}</code> <code>\declpkg [<i>options</i>] {<i>package</i>} [<i>prereqs</i>] <<<i>antireqs</i>>></code> <code>\declpkg [<i>options</i>] {<i>package</i>} (<<i>load before</i>>) (<<i>load after</i>>)</code> <code>\declpkg <*> [<i>options</i>] {<i>package</i>} [<i>prereqs</i>] [<i>postreqs</i>] [<i>coreqs</i>]</code> <code><<<i>antireqs</i>>> (<<i>load before</i>>) (<<i>load after</i>>)</code>
-----------------------	--

`\declpkg` declares a package that will be included. The `<*>` variant disables the package.

<code>\IncludePackages</code>	Calling <code>\IncludePackages</code> will include all packages. There is no harm in calling this multiple times.
-------------------------------	---

```
\usepackage{declpkg}
% Some declarations ...
```

```
\IncludePackages
\begin{document}
```

<code>\AfterAll</code> <code>\BeforeAll</code>	<code>\AfterAll {<i>packages</i>} {<i>code</i>}</code> <code>\BeforeAll {<i>packages</i>} {<i>code</i>}</code>
---	---

`\AfterAll` and `\BeforeAll` hook some `{code}` to run after/before all `{packages}` have loaded.

<code>\AfterAny</code> <code>\BeforeAny</code>	<code>\AfterAny {<i>packages</i>} {<i>code</i>}</code> <code>\BeforeAny {<i>packages</i>} {<i>code</i>}</code>
---	---

`\AfterAny` and `\BeforeAny` hook some `{code}` to run after/before one of `{packages}` has loaded. This code is only ran once.

<code>\AddOptionsToPackage</code> <code>\RemoveOptionsToPackage</code>	<code>\AddOptionsToPackage {<i>options</i>} {<i>packages</i>}</code> <code>\RemoveOptionsToPackage {<i>options</i>} {<i>packages</i>}</code>
---	---

Add or remove options from a package. Remove takes priority over add.

3 Implementation

```

1 <*package>
2 <@@=declpkg>
   Version data information.
3 \RequirePackage{expl3}
4 \ProvidesExplPackage{declpkg}{2020-05-10}{1.1}{Including packages with dependencies}
   Required packages.
5 \RequirePackage { xparse }

```

3.1 Lambdas

`\g__declpkg_lambda_counter_int` A counter for lambda functions declared.

```

6 \int_new:N \g__declpkg_lambda_counter_int

```

(End definition for \g__declpkg_lambda_counter_int.)

`\g__declpkg_last_lambda_tl` The name of the last lambda function.

```

7 \tl_new:N \g__declpkg_last_lambda_tl

```

(End definition for \g__declpkg_last_lambda_tl.)

`__declpkg_lambda:n` Create a lambda function and store its name in `\g__declpkg_last_lambda_tl`

```

8 \cs_new_protected:Nn \__declpkg_lambda:n
9 {
10   \tl_gset:Nx \g__declpkg_last_lambda_tl
11   {
12     c__declpkg_lambda_
13     \int_use:N \g__declpkg_lambda_counter_int
14     _tl
15   }
16
17   \tl_const:cn { \tl_use:N \g__declpkg_last_lambda_tl } { #1 }
18   \int_gincr:N \g__declpkg_lambda_counter_int
19 }
20 \cs_generate_variant:Nn \__declpkg_lambda:n { x }

```

(End definition for __declpkg_lambda:n.)

`_declpkg_transient_lambda:n` Create a lambda function that can only run once and store its name in `\g__declpkg_last_lambda_tl`

`_declpkg_transient_lambda:nn` The two argument version takes a transiency variable and the code.

```

21 \cs_new:Nn \_declpkg_transient_lambda:Nn
22 {
23   \__declpkg_lambda:n
24   {
25     \bool_if:NF #1
26     {
27       \bool_gset_true:N #1
28       #2
29     }
30   }
31 }
32 \cs_generate_variant:Nn \_declpkg_transient_lambda:Nn { cn, Nx, cx }
33

```

```

34 \cs_new:Nn \__declpkg_transient_lambda:n
35 {
36   \group_begin:
37   \tl_set:Nn \l_tmpa_tl
38   {
39     g__declpkg_lambda_
40     \int_use:N \g__declpkg_lambda_counter_int
41     _bool
42   }
43   \bool_new:c { \tl_use:N \l_tmpa_tl }
44
45   \__declpkg_transient_lambda:cn { \tl_use:N \l_tmpa_tl } { #1 }
46   \group_end:
47 }
48 \cs_generate_variant:Nn \__declpkg_transient_lambda:n { x }

```

(End definition for __declpkg_transient_lambda:n and __declpkg_transient_lambda:nn.)

3.2 Internal helpers

__declpkg_new_var:Nn Create a new variable of a certain type.

```

49 \cs_new_protected:Nn \__declpkg_new_var:Nn
50 {
51   \cs:w #2 _if_exist:NF \cs_end: #1
52   {
53     \cs:w #2 _new:N \cs_end: #1
54   }
55 }
56 \cs_generate_variant:Nn \__declpkg_new_var:Nn { cn }

```

(End definition for __declpkg_new_var:Nn.)

__declpkg_seq_ginsert:Nn Insert an element uniquely into a global seq. True if successfully inserted, False otherwise.

__declpkg_seq_ginsert:NnTF

```

57 \prg_new_protected_conditional:Nnn \__declpkg_seq_ginsert:Nn { T, F, TF }
58 {
59   \__declpkg_new_var:Nn #1 { seq }
60   \seq_if_in:NnTF #1 { #2 }
61   {
62     \prg_return_false:
63   } {
64     \seq_gput_right:Nn #1 { #2 }
65     \prg_return_true:
66   }
67 }
68 \prg_generate_conditional_variant:Nnn \__declpkg_seq_ginsert:Nn
69 { NV, Nv, No, Nx, cn, cV, cv, co, cx } { T, F, TF }
70
71 \cs_new:Nn \__declpkg_seq_ginsert:Nn
72 {
73   \__declpkg_seq_ginsert:NnTF #1 { #2 } {} {}
74 }
75 \cs_generate_variant:Nn \__declpkg_seq_ginsert:Nn { NV, Nv, No, Nx, cn, cV, cv, co, cx }

```

(End definition for __declpkg_seq_ginsert:Nn and __declpkg_seq_ginsert:NnTF.)

3.3 Variables and constants

<code>\g__declpkg_tmpa_tl</code>	Temp variables.
<code>\g__declpkg_tmpp_tl</code>	<small>76 \tl_new:N \g__declpkg_tmpa_tl</small>
<code>\l__declpkg_tmpa_tl</code>	<small>77 \tl_new:N \g__declpkg_tmpp_tl</small>
<code>\l__declpkg_tmpa_clist</code>	<small>78 \tl_new:N \l__declpkg_tmpa_tl</small>
	<small>79 \clist_new:N \l__declpkg_tmpa_clist</small>
	<i>(End definition for \g__declpkg_tmpa_tl and others.)</i>
<code>\g__declpkg_packages_seq</code>	A list of packages declared with declpkg.
	<small>80 \seq_new:N \g__declpkg_packages_seq</small>
	<i>(End definition for \g__declpkg_packages_seq.)</i>
<code>\g__declpkg_should_load_queue_seq</code>	Lists of packages that should and should not be loaded.
<code>\g__declpkg_should_load_seq</code>	<small>81 \seq_new:N \g__declpkg_should_load_queue_seq</small>
<code>\g__declpkg_should_not_load_seq</code>	<small>82 \seq_new:N \g__declpkg_should_load_seq</small>
	<small>83 \seq_new:N \g__declpkg_should_not_load_seq</small>
	<i>(End definition for \g__declpkg_should_load_queue_seq, \g__declpkg_should_load_seq, and \g__declpkg_should_not_load_seq.)</i>
<code>\g__declpkg_loaded_seq</code>	Lists of packages that are loaded or queued to be loaded.
<code>\g__declpkg_to_load_seq</code>	<small>84 \seq_new:N \g__declpkg_loaded_seq</small>
	<small>85 \seq_new:N \g__declpkg_to_load_seq</small>
	<i>(End definition for \g__declpkg_loaded_seq and \g__declpkg_to_load_seq.)</i>
<code>\g__declpkg_<pkg>_add_options_seq</code>	List of options that should be added or removed for a package. Removed takes priority
<code>\g__declpkg_<pkg>_remove_options_seq</code>	over added.
	<i>(End definition for \g__declpkg_<pkg>_add_options_seq and \g__declpkg_<pkg>_remove_options_seq.)</i>
<code>\g__declpkg_<pkg>_load_after_seq</code>	List of packages that if loaded, should be done before or after a package.
<code>\g__declpkg_<pkg>_load_before_seq</code>	<i>(End definition for \g__declpkg_<pkg>_load_after_seq and \g__declpkg_<pkg>_load_before_seq.)</i>
<code>\g__declpkg_<pkg>_coreq_seq</code>	List of packages that should be loaded or not with a package.
<code>\g__declpkg_<pkg>_antireq_seq</code>	<i>(End definition for \g__declpkg_<pkg>_coreq_seq and \g__declpkg_<pkg>_antireq_seq.)</i>
<code>\g__declpkg_<pkg>_pre_hook_seq</code>	List of macros that should be run before/after loading a package.
<code>\g__declpkg_<pkg>_post_hook_seq</code>	<i>(End definition for \g__declpkg_<pkg>_pre_hook_seq and \g__declpkg_<pkg>_post_hook_seq.)</i>
<code>\g__declpkg_<pkg>_load_reason_seq</code>	List of reasons as to why a package should be loaded or not.
<code>\g__declpkg_<pkg>_not_load_reason_seq</code>	<i>(End definition for \g__declpkg_<pkg>_load_reason_seq and \g__declpkg_<pkg>_not_load_reason_seq.)</i>
<code>\g__declpkg_<pkg>_preorder_graph_children_seq</code>	List of packages that should be loaded before a package.
	<i>(End definition for \g__declpkg_<pkg>_preorder_graph_children_seq.)</i>
<code>\g__declpkg_<pkg>_to_load_parent_tl</code>	The direct parent that loaded a package.
	<i>(End definition for \g__declpkg_<pkg>_to_load_parent_tl.)</i>

3.4 Errors

`_declpkg_multiple_declare_error:n` If a package is declared more than once.

```

86 \cs_new:Nn \_declpkg_multiple_declare_error:n
87 {
88   \PackageError {declpkg}
89   {
90     Multiple~declare, \MessageBreak
91     packages~should~only~be~declared~once
92   } {
93     Package~ #1 ~is~declared~more~than~once.
94   }
95 }

```

(End definition for _declpkg_multiple_declare_error:n.)

`_declpkg_should_and_should_not_error:n` If a package, both, should be and should not be included.

```

96 \cs_new_protected:Nn \_declpkg_should_and_should_not_error:n
97 {
98   \PackageError {declpkg}
99   {
100     Package~conflict, \MessageBreak
101     package~ #1 ~is~set~to~be~included~and~not~included
102   } {
103     Package~ #1 ~should~be~included
104     \MessageBreak \space \space
105     \seq_use:cnnn { g\_declpkg_ #1 _load_reason_seq }
106     { ~and \MessageBreak \space \space }
107     { , \MessageBreak \space \space }
108     { ,~and \MessageBreak \space \space }
109     ,\MessageBreak
110     but~should~not~be~included
111     \MessageBreak \space \space
112     \seq_use:cnnn { g\_declpkg_ #1 _not_load_reason_seq }
113     { ~and \MessageBreak \space \space }
114     { , \MessageBreak \space \space }
115     { ,~and \MessageBreak \space \space }
116     .
117   }
118 }

```

(End definition for _declpkg_should_and_should_not_error:n.)

`_declpkg_late_pre_hook_error:nn`

```

119 \cs_new:Nn \_declpkg_late_pre_hook_error:nn
120 {
121   \cs_show:c { #2 }
122   \str_set:Nx \l_tmpa_str { \cs_to_str:N \cs:w #2 \cs_end: }
123   \str_show:N \l_tmpa_str
124   \PackageError { declpkg }
125   {
126     Pre-hook~defined~too~late, \MessageBreak
127     pre-hooks~must~be~defined~before~packages~have~loaded
128   } {
129     Pre-hook~ #2 ~ is~defined~after~package~ #1 ~has~loaded.

```

```

130   }
131 }

(End definition for \_declpkg_late_pre_hook_error:nn.)

```

```

\_declpkg_dag_error:n If the dependency tree is not a DAG.

132 \cs_new_protected:Nn \_declpkg_dag_error:n
133 {
134   \tl_gset:Nn \g_tmpa_tl { #1 }
135   \tl_gset:Nn \g_tmpb_tl { #1 }
136   \seq_gclear:N \g_tmpa_seq
137
138   \bool_do_until:nn { \tl_if_eq_p:NN \g_tmpa_tl \g_tmpb_tl }
139   {
140     \seq_gput_left:NV \g_tmpa_seq \g_tmpb_tl
141
142     \group_begin:
143     \tl_set_eq:Nc \l_tmpa_tl
144     {
145       g\_declpkg\_
146       \tl_use:N \g_tmpb_tl
147       _to_load_parent_tl
148     }
149     \tl_gset_eq:NN \g_tmpb_tl \l_tmpa_tl
150     \group_end:
151   }
152
153   \seq_gput_left:NV \g_tmpa_seq \g_tmpa_tl
154
155   \PackageError { declpkg }
156   {
157     The~dependency~tree~is~not~a~DAG, \MessageBreak
158     including~package~ #1 ~created~a~cycle
159   } {
160     The~cycle~is
161     \MessageBreak \space \space
162     \seq_use:Nn \g_tmpa_seq { ~->~ }
163   }
164 }

(End definition for \_declpkg_dag_error:n.)

```

3.5 Internal interfaces

```

\_declpkg_add_option:nn Add or remove an option from a package.
\_declpkg_remove_option:nn
165 \cs_new:Nn \_declpkg_add_option:nn
166 {
167   \_declpkg_seq_ginsert:cn { g\_declpkg_ #1 _add_options_seq } { #2 }
168 }
169
170 \cs_new:Nn \_declpkg_remove_option:nn
171 {
172   \_declpkg_seq_ginsert:cn { g\_declpkg_ #1 _remove_options_seq } { #2 }
173 }

```

(End definition for `_declpkg_add_option:nn` and `_declpkg_remove_option:nn`.)

`_declpkg_load_after:nn` Introduce a precedence or posteriority relation between two packages. $\langle package2 \rangle$ loads
`_declpkg_load_before:nn` after/before $\langle package1 \rangle$

```

174 \cs_new:Nn \_declpkg\_load\_before:nn
175 {
176   \_declpkg\_seq\_ginsert:cn { g\_declpkg\_ #1\_load\_before\_seq } { #2 }
177 }
178
179 \cs_new:Nn \_declpkg\_load\_after:nn
180 {
181   \_declpkg\_seq\_ginsert:cn { g\_declpkg\_ #1\_load\_after\_seq } { #2 }
182 }

```

(End definition for `_declpkg_load_after:nn` and `_declpkg_load_before:nn`.)

`_declpkg_add_coreq:nn` Introduce a dependency or a conflict relation between two packages.

```

183 \cs_new:Nn \_declpkg\_add\_coreq:nn
184 {
185   \_declpkg\_seq\_ginsert:cn { g\_declpkg\_ #1\_coreq\_seq } { #2 }
186 }
187
188 \cs_new_protected:Nn \_declpkg\_add\_prereq:nn
189 {
190   \_declpkg\_add\_coreq:nn { #1 } { #2 }
191   \_declpkg\_load\_before:nn { #1 } { #2 }
192 }
193
194 \cs_new_protected:Nn \_declpkg\_add\_postreq:nn
195 {
196   \_declpkg\_add\_coreq:nn { #1 } { #2 }
197   \_declpkg\_load\_after:nn { #1 } { #2 }
198 }
199
200 \cs_new:Nn \_declpkg\_add\_antireq:nn
201 {
202   \_declpkg\_seq\_ginsert:cn { g\_declpkg\_ #1\_antireq\_seq } { #2 }
203 }

```

(End definition for `_declpkg_add_coreq:nn` and others.)

`_declpkg_should_load:nn` Indicate if a package should be loaded or not with a reason.

```

204 \cs_new_protected:Nn \_declpkg\_should\_load:nn
205 {
206   \group_begin:
207   \tl_set:Nn \l_tmpa_tl { #2 }
208   \tl_if_blank:VF \l_tmpa_tl
209   {
210     \_declpkg\_seq\_ginsert:cV { g\_declpkg\_ #1\_load\_reason\_seq } \l_tmpa_tl
211   }
212   \group_end:
213
214   % error on conflict
215   \seq_if_in:NnT \g\_declpkg\_should\_not\_load\_seq { #1 }

```



```

216 {
217   \__declpkg_should_and_should_not_error:n { #1 }
218 }
219
220 \seq_if_in:NnF \g__declpkg_should_load_seq { #1 }
221 {
222   \__declpkg_seq_ginsert:Nn \g__declpkg_should_load_queue_seq { #1 }
223 }
224 }
225
226 \cs_new_protected:Nn \__declpkg_should_not_load:nn
227 {
228   \group_begin:
229   \tl_set:Nn \l_tmpa_tl { #2 }
230   \tl_if_blank:VF \l_tmpa_tl
231   {
232     \__declpkg_seq_ginsert:cV { g__declpkg_ #1 _not_load_reason_seq } \l_tmpa_tl
233   }
234   \group_end:
235
236   % error on conflict
237   \group_begin:
238   \seq_concat:NNN \l_tmpa_seq \g__declpkg_should_load_seq \g__declpkg_should_load_queue_seq
239   \seq_if_in:NnT \l_tmpa_seq { #1 }
240   {
241     \__declpkg_should_and_should_not_error:n { #1 }
242   }
243   \group_end:
244
245   \__declpkg_seq_ginsert:Nn \g__declpkg_should_not_load_seq { #1 }
246 }

```

(End definition for __declpkg_should_load:nn and __declpkg_should_not_load:nn.)

```

\__declpkg_add_pre_hook:nn Add hooks to a package that will be ran before/after a package is loaded.
\__declpkg_add_post_hook:nn
247 \cs_new_protected:Nn \__declpkg_add_pre_hook:nn
248 {
249   \seq_if_in:NnTF \g__declpkg_loaded_seq { #1 }
250   {
251     \__declpkg_late_pre_hook_error:nn { #1 } { #2 }
252   } {
253     \seq_if_in:NnTF \g__declpkg_should_load_seq { #1 }
254     {
255       \tl_use:c { #2 }
256     } {
257       \__declpkg_seq_ginsert:cn { g__declpkg_ #1 _pre_hook_seq } { #2 }
258     }
259   }
260 }
261 \cs_generate_variant:Nn \__declpkg_add_pre_hook:nn { nV, nv, nx, Vn, VV, Vv, Vx, vn, vV, vv,
262
263 \cs_new_protected:Nn \__declpkg_add_post_hook:nn
264 {
265   \seq_if_in:NnTF \g__declpkg_loaded_seq { #1 }

```

```

266 {
267   \tl_use:c { #2 }
268 } {
269   \__declpkg_seq_ginsert:cn { g__declpkg_ #1 _post_hook_seq } { #2 }
270 }
271 }
272 \cs_generate_variant:Nn \__declpkg_add_post_hook:nn { nV, nv, nx, Vn, VV, Vv, Vx, vn, vV, vv,
(End definition for \__declpkg_add_pre_hook:nn and \__declpkg_add_post_hook:nn.)

```

3.6 Internals

`__declpkg_prep_load:n` Move a package to `should_load_seq`, run its pre hooks, and add the necessary `coreq` and `antireq`.

```

273 \cs_new_protected:Nn \__declpkg_prep_load:n
274 {
275   % add to should load
276   \__declpkg_seq_ginsert:Nn \g__declpkg_should_load_seq { #1 }
277
278   % run pre hooks
279   \tl_gset:Nn \g__declpkg_tmpa_tl { g__declpkg_ #1 _pre_hook_seq }
280   \seq_if_exist:cT { \tl_use:N \g__declpkg_tmpa_tl }
281   {
282     \seq_map_inline:cn { \tl_use:N \g__declpkg_tmpa_tl }
283     {
284       \tl_use:c { ##1 }
285     }
286   }
287
288   % setup preorder graph
289   \group_begin:
290   \tl_set:Nn \l_tmpa_tl { g__declpkg_ #1 _load_before_seq }
291   \seq_if_exist:cT { \tl_use:N \l_tmpa_tl }
292   {
293     \seq_map_inline:cn { \tl_use:N \l_tmpa_tl }
294     {
295       \__declpkg_seq_ginsert:cn { g__declpkg_ #1 _preorder_graph_children_seq } { ##1 }
296     }
297   }
298   \group_end:
299
300   \group_begin:
301   \tl_set:Nn \l_tmpa_tl { g__declpkg_ #1 _load_after_seq }
302   \seq_if_exist:cT { \tl_use:N \l_tmpa_tl }
303   {
304     \seq_map_inline:cn { \tl_use:N \l_tmpa_tl }
305     {
306       \__declpkg_seq_ginsert:cn { g__declpkg_ ##1 _preorder_graph_children_seq } { #1 }
307     }
308   }
309   \group_end:
310
311   % load coreqs
312   \group_begin:

```

```

313 \tl_set:Nn \l_tmpa_tl { g__declpkg_ #1 _coreq_seq }
314 \seq_if_exist:cT { \tl_use:N \l_tmpa_tl }
315 {
316   \seq_map_inline:cn { \tl_use:N \l_tmpa_tl }
317   {
318     \__declpkg_should_load:nn { ##1 } { per~request-of~#1 }
319   }
320 }
321 \group_end:
322
323 % not load antireqs
324 \group_begin:
325 \tl_set:Nn \l_tmpa_tl { g__declpkg_ #1 _anticoreq_seq }
326 \seq_if_exist:cT { \tl_use:N \l_tmpa_tl }
327 {
328   \seq_map_inline:cn { \tl_use:N \l_tmpa_tl }
329   {
330     \__declpkg_should_not_load:nn { ##1 } { per~request-of~#1 }
331   }
332 }
333 \group_end:
334 }
335 \cs_generate_variant:Nn \__declpkg_prep_load:n { V, v, x }

```

(End definition for __declpkg_prep_load:n.)

__declpkg_load:n Include a package and run its post hooks.

```

336 \cs_new_protected:Nn \__declpkg_load:n
337 {
338   % require package with options
339   \group_begin:
340   \seq_clear:N \l_tmpa_seq
341   \tl_set:Nn \l_tmpa_tl { g__declpkg_ #1 _add_options_seq }
342   \seq_if_exist:cT { \tl_use:N \l_tmpa_tl }
343   {
344     \seq_set_eq:Nc \l_tmpa_seq { \tl_use:N \l_tmpa_tl }
345     \seq_remove_duplicates:N \l_tmpa_seq
346
347     \tl_set:Nn \l_tmpa_tl { g__declpkg_ #1 _remove_options_seq }
348     \seq_if_exist:cT { \tl_use:N \l_tmpa_tl }
349     {
350       \seq_map_inline:cn { \tl_use:N \l_tmpa_tl }
351       {
352         \seq_remove_all:Nn \l_tmpa_seq { ##1 }
353       }
354     }
355   }
356
357   \seq_if_empty:NF \l_tmpa_seq
358   {
359     \PassOptionsToPackage { \seq_use:Nn \l_tmpa_seq { , } } { #1 }
360   }
361   \group_end:
362   \RequirePackage { #1 }

```

```

363
364 % add to loaded
365 \__declpkg_seq_ginsert:Nn \g__declpkg_loaded_seq { #1 }
366
367 % run post hooks
368 \tl_gset:Nn \g__declpkg_tmpa_tl { g__declpkg_ #1 _post_hook_seq }
369 \seq_if_exist:cT { \tl_use:N \g__declpkg_tmpa_tl }
370 {
371   \seq_map_inline:cn { \tl_use:N \g__declpkg_tmpa_tl }
372   {
373     \tl_use:c { ##1 }
374   }
375 }
376 }
377 \cs_generate_variant:Nn \__declpkg_load:n { V }

```

(End definition for __declpkg_load:n.)

__declpkg_to_load:n Loads a package while making sure all dependencies are loaded first. The first argument is $\langle package \rangle$ and $\langle parent \rangle$ is an optional extra parameter.

```

\__declpkg_to_load:nn
378 \cs_new_protected:Nn \__declpkg_to_load:n
379 {
380   \__declpkg_seq_ginsert:NnF \g__declpkg_to_load_seq { #1 }
381   {
382     \__declpkg_dag_error:n { #1 }
383   }
384
385   \tl_gset:Nn \g__declpkg_tmpa_tl { g__declpkg_ #1 _preorder_graph_children_seq }
386   \seq_if_exist:cT { \tl_use:N \g__declpkg_tmpa_tl }
387   {
388     \seq_map_inline:cn { \tl_use:N \g__declpkg_tmpa_tl }
389     {
390       \seq_if_in:NnF \g__declpkg_loaded_seq { ##1 }
391       {
392         \__declpkg_to_load:nn { ##1 } { #1 }
393       }
394     }
395   }
396
397   \__declpkg_load:n { #1 }
398 }
399
400 \cs_new_protected:Nn \__declpkg_to_load:nn
401 {
402   \group_begin:
403   \tl_set:Nn \l_tmpa_tl { g__declpkg_ #1 _to_load_parent_tl }
404   \__declpkg_new_var:cn { \tl_use:N \l_tmpa_tl } { tl }
405   \tl_gset:cn { \tl_use:N \l_tmpa_tl } { #2 }
406   \group_end:
407
408   \__declpkg_to_load:n { #1 }
409 }

```

(End definition for __declpkg_to_load:n and __declpkg_to_load:nn.)

`__declpkg_include_packages:`

```

410 \cs_new:Nn \__declpkg_include_packages:
411 {
412   \bool_until_do:nn { \seq_if_empty_p:N \g__declpkg_should_load_queue_seq }
413   {
414     \group_begin:
415     \seq_gpop_left:NN \g__declpkg_should_load_queue_seq \l__declpkg_tmpa_tl
416     \tl_gset_eq:NN \g__declpkg_tmpa_tl \l__declpkg_tmpa_tl
417     \group_end:
418
419     \seq_if_in:NVF \g__declpkg_should_load_seq \g__declpkg_tmpa_tl
420     {
421       \__declpkg_prep_load:V \g__declpkg_tmpa_tl
422     }
423   }
424
425   \seq_map_inline:Nn \g__declpkg_should_load_seq
426   {
427     \seq_if_in:NnF \g__declpkg_to_load_seq { ##1 }
428     {
429       \__declpkg_to_load:n { ##1 }
430     }
431   }
432 }

```

(End definition for `__declpkg_include_packages:`.)

3.7 Public interfaces

`\declpkg` The main command to declare a package.

```

433 \NewDocumentCommand \declpkg { s o m d[] d[] d[] d<> d() d() }
434 {
435   \__declpkg_seq_ginsert:NnF \g__declpkg_package_seq { #3 }
436   {
437     \__declpkg_multiple_declare_error:n { #3 }
438   }
439
440   \IfBooleanTF { #1 }
441   {
442     \__declpkg_should_not_load:nn { #3 } { per~user~request }
443   } {
444     \__declpkg_should_load:nn { #3 } { per~user~request }
445   }
446
447   \IfValueT { #2 }
448   {
449     \group_begin:
450     \clist_set:Nn \l_tmpa_clist { #2 }
451     \clist_map_inline:Nn \l_tmpa_clist
452     {
453       \__declpkg_add_option:nn { #3 } { ##1 }
454     }
455     \group_end:
456   }

```

```

457
458 \IfValueT { #4 }
459 {
460   \group_begin:
461   \clist_set:Nn \l_tmpa_clist { #4 }
462   \clist_map_inline:Nn \l_tmpa_clist
463   {
464     \__declpkg_add_prereq:nn { #3 } { ##1 }
465   }
466   \group_end:
467 }
468
469 \IfValueT { #5 }
470 {
471   \group_begin:
472   \clist_set:Nn \l_tmpa_clist { #5 }
473   \clist_map_inline:Nn \l_tmpa_clist
474   {
475     \__declpkg_add_postreq:nn { #3 } { ##1 }
476   }
477   \group_end:
478 }
479
480 \IfValueT { #6 }
481 {
482   \group_begin:
483   \clist_set:Nn \l_tmpa_clist { #6 }
484   \clist_map_inline:Nn \l_tmpa_clist
485   {
486     \__declpkg_add_coreq:nn { #3 } { ##1 }
487   }
488   \group_end:
489 }
490
491
492 \IfValueT { #7 }
493 {
494   \group_begin:
495   \clist_set:Nn \l_tmpa_clist { #7 }
496   \clist_map_inline:Nn \l_tmpa_clist
497   {
498     \__declpkg_add_antireq:nn { #3 } { ##1 }
499   }
500   \group_end:
501 }
502
503 \IfValueT { #8 }
504 {
505   \group_begin:
506   \clist_set:Nn \l_tmpa_clist { #8 }
507   \clist_map_inline:Nn \l_tmpa_clist
508   {
509     \__declpkg_load_before:nn { #3 } { ##1 }
510   }

```

```

511   \group_end:
512 }
513
514 \IfValueT { #9 }
515 {
516   \group_begin:
517   \clist_set:Nn \l_tmpa_clist { #9 }
518   \clist_map_inline:Nn \l_tmpa_clist
519   {
520     \__declpkg_load_after:nn { #3 } { ##1 }
521   }
522   \group_end:
523 }
524 }

```

(End definition for `\declpkg`. This function is documented on page 2.)

`\IncludePackages` Public interface of `__declpkg_include_packages:`.

```

525 \NewDocumentCommand \IncludePackages {} { \__declpkg_include_packages: }

```

(End definition for `\IncludePackages`. This function is documented on page 2.)

`\AddOptionsToPackage` Add or remove options to a package.

`\RemoveOptionsToPackage`

```

526 \NewDocumentCommand \AddOptionsToPackage { m m }
527 {
528   \group_begin:
529   \clist_set:Nn \l_tmpa_clist { #1 }
530   \clist_map_inline:Nn \l_tmpa_clist
531   {
532     \__declpkg_add_option:nn { #2 } { #1 }
533   }
534   \group_end:
535 }
536
537 \NewDocumentCommand \RemoveOptionsToPackage { m m }
538 {
539   \group_begin:
540   \clist_set:Nn \l_tmpa_clist { #1 }
541   \clist_map_inline:Nn \l_tmpa_clist
542   {
543     \__declpkg_remove_option:nn { #2 } { #1 }
544   }
545   \group_end:
546 }

```

(End definition for `\AddOptionsToPackage` and `\RemoveOptionsToPackage`. These functions are documented on page 2.)

`\AfterAll` Hook some code after/before loading all packages in a list.

`\BeforeAll`

```

547 \NewDocumentCommand \AfterAll { m +m }
548 {
549   \group_begin:
550   \clist_set:Nn \l_tmpa_clist { #1 }
551   \__declpkg_lambda:n { #2 }
552 }

```

```

553 \bool_until_do:nn { \clist_if_empty_p:N \l_tmpa_clist }
554 {
555   \clist_pop:NN \l_tmpa_clist \l_tmpb_tl
556   \__declpkg_lambda:x { \__declpkg_add_post_hook:nn { \tl_use:N \l_tmpb_tl } { \tl_use:N \g
557 }
558 \group_end:
559
560 \tl_use:c { \tl_use:N \g__declpkg_last_lambda_tl }
561 }
562
563 \NewDocumentCommand \BeforeAll { m +m }
564 {
565   \group_begin:
566   \clist_set:Nn \l_tmpa_clist { #1 }
567   \__declpkg_lambda:n { #2 }
568
569   \bool_until_do:nn { \clist_if_empty_p:N \l_tmpa_clist }
570   {
571     \clist_pop:NN \l_tmpa_clist \l_tmpb_tl
572     \__declpkg_lambda:x { \__declpkg_add_pre_hook:nn { \tl_use:N \l_tmpb_tl } { \tl_use:N \g
573   }
574   \group_end:
575
576   \tl_use:c { \tl_use:N \g__declpkg_last_lambda_tl }
577 }

```

(End definition for `\AfterAll` and `\BeforeAll`. These functions are documented on page 2.)

\AfterAny Hook some code after/before loading any one package in a list.

\BeforeAny

```

578 \NewDocumentCommand \AfterAny { m +m }
579 {
580   \group_begin:
581   \clist_set:Nn \l__declpkg_tmpa_clist { #1 }
582   \__declpkg_transient_lambda:n { #2 }
583   \tl_set_eq:NN \l__declpkg_tmpa_tl \g__declpkg_last_lambda_tl
584
585   \bool_until_do:nn { \clist_if_empty_p:N \l__declpkg_tmpa_clist }
586   {
587     \clist_pop:NN \l__declpkg_tmpa_clist \l_tmpa_tl
588     \tl_gset_eq:NN \g__declpkg_tmpa_tl \l_tmpa_tl
589     \tl_gset_eq:NN \g__declpkg_tmpb_tl \l__declpkg_tmpa_tl
590     \group_end:
591     \__declpkg_add_post_hook:VV \g__declpkg_tmpa_tl \g__declpkg_tmpb_tl
592     \group_begin:
593   }
594   \group_end:
595 }
596
597 \NewDocumentCommand \BeforeAny { m +m }
598 {
599   \group_begin:
600   \clist_set:Nn \l__declpkg_tmpa_clist { #1 }
601   \__declpkg_transient_lambda:n { #2 }
602   \tl_set_eq:NN \l__declpkg_tmpa_tl \g__declpkg_last_lambda_tl

```



```

603
604 \bool_until_do:nn { \clist_if_empty_p:N \l__declpkg_tmpa_clist }
605 {
606   \clist_pop:NN \l__declpkg_tmpa_clist \l_tmpa_tl
607   \tl_gset_eq:NN \g__declpkg_tmpa_tl \l_tmpa_tl
608   \tl_gset_eq:NN \g__declpkg_tmpb_tl \l__declpkg_tmpa_tl
609   \group_end:
610   \__declpkg_add_pre_hook:VV \g__declpkg_tmpa_tl \g__declpkg_tmpb_tl
611   \group_begin:
612 }
613 \group_end:
614 }

```

(End definition for \AfterAny and \BeforeAny. These functions are documented on page 2.)

```

615 \endpackage

```

Change History

v1.0		v1.1
General: First public release	1	General: Add pre-hooks 1

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.

A	
<code>\AddOptionsToPackage</code>	<u>2</u> , <u>526</u>
<code>\AfterAll</code>	<u>2</u> , <u>547</u>
<code>\AfterAny</code>	<u>2</u> , <u>578</u>
B	
<code>\BeforeAll</code>	<u>2</u> , <u>547</u>
<code>\BeforeAny</code>	<u>2</u> , <u>578</u>
bool commands:	
<code>\bool_do_until:nn</code>	<u>138</u>
<code>\bool_gset_true:N</code>	<u>27</u>
<code>\bool_if:NTF</code>	<u>25</u>
<code>\bool_new:N</code>	<u>43</u>
<code>\bool_until_do:nn</code>	<u>412</u> , <u>553</u> , <u>569</u> , <u>585</u> , <u>604</u>
C	
clist commands:	
<code>\clist_if_empty_p:N</code>	<u>553</u> , <u>569</u> , <u>585</u> , <u>604</u>
<code>\clist_map_inline:Nn</code>	<u>451</u> , <u>462</u> , <u>473</u> , <u>484</u> , <u>496</u> , <u>507</u> , <u>518</u> , <u>530</u> , <u>541</u>
<code>\clist_new:N</code>	<u>79</u>
<code>\clist_pop:NN</code>	<u>555</u> , <u>571</u> , <u>587</u> , <u>606</u>
<code>\clist_set:Nn</code>	<u>450</u> , <u>461</u> , <u>472</u> , <u>483</u> , <u>495</u> , <u>506</u> , <u>517</u> , <u>529</u> , <u>540</u> , <u>550</u> , <u>566</u> , <u>581</u> , <u>600</u>
<code>\l_tmpa_clist</code>	<u>450</u> , <u>451</u> , <u>461</u> , <u>462</u> , <u>472</u> , <u>473</u> , <u>483</u> , <u>484</u> , <u>495</u> , <u>496</u> , <u>506</u> , <u>507</u> , <u>517</u> , <u>518</u> , <u>529</u> , <u>530</u> , <u>540</u> , <u>541</u> , <u>550</u> , <u>553</u> , <u>555</u> , <u>566</u> , <u>569</u> , <u>571</u>
cs commands:	
<code>\cs:w</code>	<u>51</u> , <u>53</u> , <u>122</u>
<code>\cs_end:</code>	<u>51</u> , <u>53</u> , <u>122</u>
<code>\cs_generate_variant:Nn</code>	<u>20</u> , <u>32</u> , <u>48</u> , <u>56</u> , <u>75</u> , <u>261</u> , <u>272</u> , <u>335</u> , <u>377</u>
<code>\cs_new:Nn</code>	<u>21</u> , <u>34</u> , <u>71</u> , <u>86</u> , <u>119</u> , <u>165</u> , <u>170</u> , <u>174</u> , <u>179</u> , <u>183</u> , <u>200</u> , <u>410</u>
<code>\cs_new_protected:Nn</code>	<u>8</u> , <u>49</u> , <u>96</u> , <u>132</u> , <u>188</u> , <u>194</u> , <u>204</u> , <u>226</u> , <u>247</u> , <u>263</u> , <u>273</u> , <u>336</u> , <u>378</u> , <u>400</u>
<code>\cs_show:N</code>	<u>121</u>
<code>\cs_to_str:N</code>	<u>122</u>
D	
<code>\declpkg</code>	<u>2</u> , <u>433</u>
declpkg internal commands:	
<code>\g__declpkg_<pkg>_add_options_</code> <code>seq</code>	<u>86</u>
<code>\g__declpkg_<pkg>_antireq_seq</code> ...	<u>86</u>
<code>\g__declpkg_<pkg>_coreq_seq</code>	<u>86</u>
<code>\g__declpkg_<pkg>_load_after_seq</code>	<u>86</u>
<code>\g__declpkg_<pkg>_load_before_</code> <code>seq</code>	<u>86</u>
<code>\g__declpkg_<pkg>_load_reason_</code> <code>seq</code>	<u>86</u>
<code>\g__declpkg_<pkg>_not_load_</code> <code>reason_seq</code>	<u>86</u>
<code>\g__declpkg_<pkg>_post_hook_seq</code> .	<u>86</u>
<code>\g__declpkg_<pkg>_pre_hook_seq</code> ..	<u>86</u>
<code>\g__declpkg_<pkg>_preorder_</code> <code>graph_children_seq</code>	<u>86</u>
<code>\g__declpkg_<pkg>_remove_</code> <code>options_seq</code>	<u>86</u>
<code>\g__declpkg_<pkg>_to_load_</code> <code>parent_tl</code>	<u>86</u>
<code>__declpkg_add_antireq:nn</code> ..	<u>183</u> , <u>498</u>
<code>__declpkg_add_coreq:nn</code>	<u>183</u> , <u>486</u>
<code>__declpkg_add_option:nn</code> ..	<u>165</u> , <u>453</u> , <u>532</u>
<code>__declpkg_add_post_hook:nn</code>	<u>247</u> , <u>556</u> , <u>591</u>
<code>__declpkg_add_postreq:nn</code> ..	<u>183</u> , <u>475</u>
<code>__declpkg_add_pre_hook:nn</code>	<u>247</u> , <u>572</u> , <u>610</u>
<code>__declpkg_add_prereq:nn</code> ...	<u>183</u> , <u>464</u>
<code>__declpkg_dag_error:n</code>	<u>132</u> , <u>382</u>
<code>__declpkg_include_packages:</code> ...	<u>15</u> , <u>410</u> , <u>525</u>
<code>__declpkg_lambda:n</code>	<u>8</u> , <u>23</u> , <u>551</u> , <u>556</u> , <u>567</u> , <u>572</u>
<code>\g__declpkg_lambda_counter_int</code> ..	<u>6</u> , <u>13</u> , <u>18</u> , <u>40</u>
<code>\g__declpkg_last_lambda_tl</code> .	<u>3</u> , <u>3</u> , <u>7</u> , <u>10</u> , <u>17</u> , <u>556</u> , <u>560</u> , <u>572</u> , <u>576</u> , <u>583</u> , <u>602</u>
<code>__declpkg_late_pre_hook_</code> <code>error:nn</code>	<u>119</u> , <u>251</u>
<code>__declpkg_load:n</code>	<u>336</u> , <u>397</u>
<code>__declpkg_load_after:nn</code> ..	<u>174</u> , <u>197</u> , <u>520</u>
<code>__declpkg_load_before:nn</code>	<u>174</u> , <u>191</u> , <u>509</u>
<code>\g__declpkg_loaded_seq</code>	<u>84</u> , <u>249</u> , <u>265</u> , <u>365</u> , <u>390</u>
<code>__declpkg_multiple_declare_</code> <code>error:n</code>	<u>86</u> , <u>437</u>
<code>__declpkg_new_var:Nn</code> ...	<u>49</u> , <u>59</u> , <u>404</u>
<code>\g__declpkg_package_seq</code>	<u>435</u>
<code>\g__declpkg_packages_seq</code>	<u>80</u>
<code>__declpkg_prep_load:n</code>	<u>273</u> , <u>421</u>
<code>__declpkg_remove_option:nn</code> ..	<u>165</u> , <u>543</u>

_declpkg_seq_ginsert:Nn	57, 167, 172, 176, 181, 185, 202, 210, 222, 232, 245, 257, 269, 276, 295, 306, 365
_declpkg_seq_ginsert:NnTF	57, 380, 435
_declpkg_should_and_should_not_error:n	96, 217, 241
_declpkg_should_load:nn	204, 318, 444
\g_declpkg_should_load_queue_seq	81, 222, 238, 412, 415
\g_declpkg_should_load_seq	81, 220, 238, 253, 276, 419, 425
_declpkg_should_not_load:nn	204, 330, 442
\g_declpkg_should_not_load_seq	81, 215, 245
\l_declpkg_tmpa_clist	76, 581, 585, 587, 600, 604, 606
\g_declpkg_tmpa_tl	76, 279, 280, 282, 368, 369, 371, 385, 386, 388, 416, 419, 421, 588, 591, 607, 610
\l_declpkg_tmpa_tl	76, 415, 416, 583, 589, 602, 608
\g_declpkg_tmpb_tl	76, 589, 591, 608, 610
_declpkg_to_load:n	378, 429
_declpkg_to_load:nn	378
\g_declpkg_to_load_seq	84, 380, 427
_declpkg_transient_lambda:n	21, 582, 601
_declpkg_transient_lambda:Nn	21, 32, 45
_declpkg_transient_lambda:nn	21
G	
group commands:	
\group_begin:	36, 142, 206, 228, 237, 289, 300, 312, 324, 339, 402, 414, 449, 460, 471, 482, 494, 505, 516, 528, 539, 549, 565, 580, 592, 599, 611
\group_end:	46, 150, 212, 234, 243, 298, 309, 321, 333, 361, 406, 417, 455, 466, 477, 488, 500, 511, 522, 534, 545, 558, 574, 590, 594, 609, 613
I	
\IfBooleanTF	440
\IfValueT	447, 458, 469, 480, 492, 503, 514
\IncludePackages	2, 525
int commands:	
\int_gincr:N	18
\int_new:N	6
\int_use:N	13, 40
M	
\MessageBreak	90, 100, 104, 106, 107, 108, 109, 111, 113, 114, 115, 126, 157, 161
N	
\NewDocumentCommand	433, 525, 526, 537, 547, 563, 578, 597
P	
\PackageError	88, 98, 124, 155
\PassOptionsToPackage	359
prg commands:	
\prg_generate_conditional_variant:Nnn	68
\prg_new_protected_conditional:Nnn	57
\prg_return_false:	62
\prg_return_true:	65
\ProvidesExplPackage	4
R	
\RemoveOptionsToPackage	2, 526
\RequirePackage	3, 5, 362
S	
seq commands:	
\seq_clear:N	340
\seq_concat:NNN	238
\seq_gclear:N	136
\seq_gpop_left:NN	415
\seq_gput_left:Nn	140, 153
\seq_gput_right:Nn	64
\seq_if_empty:NnTF	357
\seq_if_empty_p:N	412
\seq_if_exist:NnTF	280, 291, 302, 314, 326, 342, 348, 369, 386
\seq_if_in:NnTF	60, 215, 220, 239, 249, 253, 265, 390, 419, 427
\seq_map_inline:Nn	282, 293, 304, 316, 328, 350, 371, 388, 425
\seq_new:N	80, 81, 82, 83, 84, 85
\seq_remove_all:Nn	352
\seq_remove_duplicates:N	345
\seq_set_eq:NN	344
\seq_use:Nn	162, 359
\seq_use:Nnnn	105, 112
\g_tmpa_seq	136, 140, 153, 162
\l_tmpa_seq	238, 239, 340, 344, 345, 352, 359
\space	104, 106, 107, 108, 111, 113, 114, 115, 161
str commands:	
\str_set:Nn	122
\str_show:N	123
\l_tmpa_str	122, 123

T		
tl commands:		
\tl_const:Nn	17
\tl_gset:Nn	
	10, 134, 135, 279, 368, 385, 405
\tl_gset_eq:NN	
	149, 416, 588, 589, 607, 608
\tl_if_blank:nTF	208, 230
\tl_if_eq_p:NN	138
\tl_new:N	7, 76, 77, 78
\tl_set:Nn	37, 207,
		229, 290, 301, 313, 325, 341, 347, 403
\tl_set_eq:NN	143, 583, 602
\tl_tmpa_seq	357
\tl_use:N	17, 43, 45,
		146, 255, 267, 280, 282, 284, 291,
		293, 302, 304, 314, 316, 326, 328,
		342, 344, 348, 350, 369, 371, 373,
		386, 388, 404, 405, 556, 560, 572, 576
\g_tmpa_tl	134, 138, 153
\l_tmpa_tl	
	...	37, 43, 45, 143, 149, 207, 208,
		210, 229, 230, 232, 290, 291, 293,
		301, 302, 304, 313, 314, 316, 325,
		326, 328, 341, 342, 344, 347, 348,
		350, 403, 404, 405, 587, 588, 606, 607
\g_tmpb_tl	135, 138, 140, 146, 149
\l_tmpb_tl	555, 556, 571, 572