declpkg — Including packages with dependencies [*]

Ting Wei Liu[†]

Released 2020-05-10

Abstract

The declpkg package includes packages with user provided order.

Contents

[*]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

`\declpkg`

```
\declpkg [⟨options⟩] {⟨package⟩}
\declpkg [⟨options⟩] {⟨package⟩} [⟨prereqs⟩] ⟨⟨antireqs⟩⟩
\declpkg [⟨options⟩] {⟨package⟩} (⟨load befores⟩) (⟨load after⟩)
\declpkg ⟨*⟩ [⟨options⟩] {⟨package⟩} [⟨prereqs⟩] [⟨postreqs⟩] [⟨coreqs⟩]
⟨⟨antireqs⟩⟩ (⟨load befores⟩) (⟨load after⟩)
```

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

`\IncludePackages`

Calling `\IncludePackages` will include all packages. There is no harm in calling this multiple times.

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

\IncludePackages
\begin{document}
```

`\AfterAll`
`\BeforeAll`

```
\AfterAll {⟨packages⟩} {⟨code⟩}
\BeforeAll {⟨packages⟩} {⟨code⟩}
```

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

`\AfterAny`
`\BeforeAny`

```
\AfterAny {⟨packages⟩} {⟨code⟩}
\BeforeAny {⟨packages⟩} {⟨code⟩}
```

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

`\AddOptionsToPackage`
`\RemoveOptionsToPackage`

```
\AddOptionsToPackage {⟨options⟩} {⟨packages⟩}
\RemoveOptionsToPackage {⟨options⟩} {⟨packages⟩}
```

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

3 Implementation

```
⟨*package⟩
⟨@@=declpkg⟩
```

Version data information.

```
\RequirePackage{expl3}
\ProvidesExplPackage{declpkg}{2020-05-10}{1.1}{Including packages with dependencies}
```

Required packages.

```
\RequirePackage { xparse }
```

3.1 Lambdas

\g__declpkg_lambda_counter_int

A counter for lambda functions declared.

```
\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.

```
\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`

```
\cs_new_protected:Nn \__declpkg_lambda:n
{
  \tl_gset:Nx \g__declpkg_last_lambda_tl
  {
    c__declpkg_lambda_
    \int_use:N \g__declpkg_lambda_counter_int
    _tl
  }

  \tl_const:cn { \tl_use:N \g__declpkg_last_lambda_tl } { #1 }
  \int_gincr:N \g__declpkg_lambda_counter_int
}
\cs_generate_variant:Nn \__declpkg_lambda:n { x }
```

(*End definition for* `\__declpkg_lambda:n`*.*)

__declpkg_transient_lambda:n
__declpkg_transient_lambda:nn

Create a lambda function that can only run once and store its name in `\g__declpkg_-last_lambda_tl` The two argument version takes a transiency variable and the code.

```
\cs_new:Nn \__declpkg_transient_lambda:Nn
{
  \__declpkg_lambda:n
  {
    \bool_if:NF #1
    {
      \bool_gset_true:N #1
      #2
    }
  }
}
\cs_generate_variant:Nn \__declpkg_transient_lambda:Nn { cn, Nx, cx }

```

```
\cs_new:Nn \__declpkg_transient_lambda:n
{
  \group_begin:
  \tl_set:Nn \l_tmpa_tl
  {
    g__declpkg_lambda_
    \int_use:N \g__declpkg_lambda_counter_int
    _bool
  }
  \bool_new:c { \tl_use:N \l_tmpa_tl }

  \__declpkg_transient_lambda:cn { \tl_use:N \l_tmpa_tl } { #1 }
  \group_end:
}
\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.

```
\cs_new_protected:Nn \__declpkg_new_var:Nn
{
  \cs:w #2 _if_exist:NF \cs_end: #1
  {
    \cs:w #2 _new:N \cs_end: #1
  }
}
\cs_generate_variant:Nn \__declpkg_new_var:Nn { cn }
```

(*End definition for* `\__declpkg_new_var:Nn`*.*)

`\__declpkg_seq_ginsert:Nn`
`\__declpkg_seq_ginsert:Nn`***TF***

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

```
\prg_new_protected_conditional:Nnn \__declpkg_seq_ginsert:Nn { T, F, TF }
{
  \__declpkg_new_var:Nn #1 { seq }
  \seq_if_in:NnTF #1 { #2 }
  {
    \prg_return_false:
  } {
    \seq_gput_right:Nn #1 { #2 }
    \prg_return_true:
  }
}
\prg_generate_conditional_variant:Nnn \__declpkg_seq_ginsert:Nn
{ NV, Nv, No, Nx, cn, cV, cv, co, cx } { T, F, TF }

\cs_new:Nn \__declpkg_seq_ginsert:Nn
{
  \__declpkg_seq_ginsert:NnTF #1 { #2 } {} {}
}
\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

\g__declpkg_tmpa_tl
\g__declpkg_tmpb_tl
\l__declpkg_tmpa_tl
\l__declpkg_tmpa_clist

Temp variables.

```
\tl_new:N \g__declpkg_tmpa_tl
\tl_new:N \g__declpkg_tmpb_tl
\tl_new:N \l__declpkg_tmpa_tl
\clist_new:N \l__declpkg_tmpa_clist
```

(*End definition for* \g__declpkg_tmpa_tl *and others.*)

\g__declpkg_packages_seq

A list of packages declared with declpkg.

```
\seq_new:N \g__declpkg_packages_seq
```

(*End definition for* \g__declpkg_packages_seq*.*)

\g__declpkg_should_load_queue_seq
\g__declpkg_should_load_seq
\g__declpkg_should_not_load_seq

Lists of packages that should and should not be loaded.

```
\seq_new:N \g__declpkg_should_load_queue_seq
\seq_new:N \g__declpkg_should_load_seq
\seq_new:N \g__declpkg_should_not_load_seq
```

(*End definition for* \g__declpkg_should_load_queue_seq*,* \g__declpkg_should_load_seq*, and* \g__-declpkg_should_not_load_seq*.*)

\g__declpkg_loaded_seq
\g__declpkg_to_load_seq

Lists of packages that are loaded or queued to be loaded.

```
\seq_new:N \g__declpkg_loaded_seq
\seq_new:N \g__declpkg_to_load_seq
```

(*End definition for* \g__declpkg_loaded_seq *and* \g__declpkg_to_load_seq*.*)

\g__declpkg_<pkg>_add_options_seq
\g__declpkg_<pkg>_remove_options_seq

List of options that should be added or removed for a package. Removed takes priority over added.

(*End definition for* \g__declpkg_<pkg>_add_options_seq *and* \g__declpkg_<pkg>_remove_options_-seq*.*)

\g__declpkg_<pkg>_load_after_seq
\g__declpkg_<pkg>_load_before_seq

List of packages that if loaded, should be done before or after a package.

(*End definition for* \g__declpkg_<pkg>_load_after_seq *and* \g__declpkg_<pkg>_load_before_seq*.*)

\g__declpkg_<pkg>_coreq_seq
\g__declpkg_<pkg>_antireq_seq

List of packages that should be loaded or not with a package.

(*End definition for* \g__declpkg_<pkg>_coreq_seq *and* \g__declpkg_<pkg>_antireq_seq*.*)

\g__declpkg_<pkg>_pre_hook_seq
\g__declpkg_<pkg>_post_hook_seq

List of macros that should be run before/after loading a package.

(*End definition for* \g__declpkg_<pkg>_pre_hook_seq *and* \g__declpkg_<pkg>_post_hook_seq*.*)

\g__declpkg_<pkg>_load_reason_seq
\g__declpkg_<pkg>_not_load_reason_seq

List of reasons as to why a package should be loaded or not.

(*End definition for* \g__declpkg_<pkg>_load_reason_seq *and* \g__declpkg_<pkg>_not_load_reason_-seq*.*)

g__declpkg_<pkg>_preorder_graph_children_seq

List of packages that should be loaded before a package.

(*End definition for* \g__declpkg_<pkg>_preorder_graph_children_seq*.*)

\g__declpkg_<pkg>_to_load_parent_tl

The direct parent that loaded a package.

(*End definition for* \g__declpkg_<pkg>_to_load_parent_tl*.*)

3.4 Errors

__declpkg_multiple_declare_error:n

If a package is declared more than once.

```
\cs_new:Nn \__declpkg_multiple_declare_error:n
{
  \PackageError {declpkg}
  {
    Multiple~declare, \MessageBreak
    packages~should~only~be~declared~once
  } {
    Package~ #1 ~is~declared~more~than~once.
  }
}
```

(*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.

```
\cs_new_protected:Nn \__declpkg_should_and_should_not_error:n
{
  \PackageError {declpkg}
  {
    Package~conflict, \MessageBreak
    package~ #1 ~is~set~to~be~included~and~not~included
  } {
    Package~ #1 ~should~be~included
    \MessageBreak \space \space
    \seq_use:cnnn { g__declpkg_ #1 _load_reason_seq }
    { ~and \MessageBreak \space \space }
    { , \MessageBreak \space \space }
    { ,~and \MessageBreak \space \space }
    ,\MessageBreak
    but~should~not~be~included
    \MessageBreak \space \space
    \seq_use:cnnn { g__declpkg_ #1 _not_load_reason_seq }
    { ~and \MessageBreak \space \space }
    { , \MessageBreak \space \space }
    { ,~and \MessageBreak \space \space }
    .
  }
}
```

(*End definition for* __declpkg_should_and_should_not_error:n*.*)

__declpkg_late_pre_hook_error:nn

```
\cs_new:Nn \__declpkg_late_pre_hook_error:nn
{
  \cs_show:c { #2 }
  \str_set:Nx \l_tmpa_str { \cs_to_str:N \cs:w #2 \cs_end: }
  \str_show:N \l_tmpa_str
  \PackageError { declpkg }
  {
    Pre-hook~defined~too~late, \MessageBreak
    pre-hooks~must~be~defined~before~packages~have~loaded
  } {
    Pre-hook~ #2 ~ is~defined~after~package~ #1 ~has~loaded.
  }
}
```

(*End definition for* `\__declpkg_late_pre_hook_error:nn`.)

`\__declpkg_dag_error:n`

If the dependency tree is not a DAG.

```
\cs_new_protected:Nn \__declpkg_dag_error:n
{
  \tl_gset:Nn \g_tmpa_tl { #1 }
  \tl_gset:Nn \g_tmpb_tl { #1 }
  \seq_gclear:N \g_tmpa_seq

  \bool_do_until:nn { \tl_if_eq_p:NN \g_tmpa_tl \g_tmpb_tl }
  {
    \seq_gput_left:NV \g_tmpa_seq \g_tmpb_tl

    \group_begin:
    \tl_set_eq:Nc \l_tmpa_tl
    {
      g__declpkg_
      \tl_use:N \g_tmpb_tl
      _to_load_parent_tl
    }
    \tl_gset_eq:NN \g_tmpb_tl \l_tmpa_tl
    \group_end:
  }

  \seq_gput_left:NV \g_tmpa_seq \g_tmpa_tl

  \PackageError { declpkg }
  {
    The~dependency~tree~is~not~a~DAG, \MessageBreak
    including~package~ #1 ~created~a~cycle
  } {
    The~cycle~is
    \MessageBreak \space \space
    \seq_use:Nn \g_tmpa_seq { ~-->~ }
  }
}
```

(*End definition for* `\__declpkg_dag_error:n`.)

3.5 Internal interfaces

`\__declpkg_add_option:nn`
`\__declpkg_remove_option:nn`

Add or remove an option from a package.

```
\cs_new:Nn \__declpkg_add_option:nn
{
  \__declpkg_seq_ginsert:cn { g__declpkg_ #1 _add_options_seq } { #2 }
}

\cs_new:Nn \__declpkg_remove_option:nn
{
  \__declpkg_seq_ginsert:cn { g__declpkg_ #1 _remove_options_seq } { #2 }
}
```

(*End definition for* `\__declpkg_add_option:nn` *and* `\__declpkg_remove_option:nn`*.*)

`\__declpkg_load_after:nn`
`\__declpkg_load_before:nn`

Introduce a precedence or posteriority relation between two packages. ⟨*package2*⟩ loads after/before ⟨*package1*⟩

```
\cs_new:Nn \__declpkg_load_before:nn
{
  \__declpkg_seq_ginsert:cn { g__declpkg_ #1 _load_before_seq } { #2 }
}

\cs_new:Nn \__declpkg_load_after:nn
{
  \__declpkg_seq_ginsert:cn { g__declpkg_ #1 _load_after_seq } { #2 }
}
```

(*End definition for* `\__declpkg_load_after:nn` *and* `\__declpkg_load_before:nn`*.*)

`\__declpkg_add_coreq:nn`
`\__declpkg_add_prereq:nn`
`\__declpkg_add_postreq:nn`
`\__declpkg_add_antireq:nn`

Introduce a dependency or a conflict relation between two packages.

```
\cs_new:Nn \__declpkg_add_coreq:nn
{
  \__declpkg_seq_ginsert:cn { g__declpkg_ #1 _coreq_seq } { #2 }
}

\cs_new_protected:Nn \__declpkg_add_prereq:nn
{
  \__declpkg_add_coreq:nn { #1 } { #2 }
  \__declpkg_load_before:nn { #1 } { #2 }
}

\cs_new_protected:Nn \__declpkg_add_postreq:nn
{
  \__declpkg_add_coreq:nn { #1 } { #2 }
  \__declpkg_load_after:nn { #1 } { #2 }
}

\cs_new:Nn \__declpkg_add_antireq:nn
{
  \__declpkg_seq_ginsert:cn { g__declpkg_ #1 _antireq_seq } { #2 }
}
```

(*End definition for* `\__declpkg_add_coreq:nn` *and others.*)

`\__declpkg_should_load:nn`
`\__declpkg_should_not_load:nn`

Indicate if a package should be loaded or not with a reason.

```
\cs_new_protected:Nn \__declpkg_should_load:nn
{
  \group_begin:
  \tl_set:Nn \l_tmpa_tl { #2 }
  \tl_if_blank:VF \l_tmpa_tl
  {
    \__declpkg_seq_ginsert:cV { g__declpkg_ #1 _load_reason_seq } \l_tmpa_tl
  }
  \group_end:

  % error on conflict
  \seq_if_in:NnT \g__declpkg_should_not_load_seq { #1 }
  {
    \__declpkg_should_and_should_not_error:n { #1 }
  }

  \seq_if_in:NnF \g__declpkg_should_load_seq { #1 }
  {
    \__declpkg_seq_ginsert:Nn \g__declpkg_should_load_queue_seq { #1 }
  }
}

\cs_new_protected:Nn \__declpkg_should_not_load:nn
{
  \group_begin:
  \tl_set:Nn \l_tmpa_tl { #2 }
  \tl_if_blank:VF \l_tmpa_tl
  {
    \__declpkg_seq_ginsert:cV { g__declpkg_ #1 _not_load_reason_seq } \l_tmpa_tl
  }
  \group_end:

  % error on conflict
  \group_begin:
  \seq_concat:NNN \l_tmpa_seq \g__declpkg_should_load_seq \g__declpkg_should_load_queue_seq
  \seq_if_in:NnT \l_tmpa_seq { #1 }
  {
    \__declpkg_should_and_should_not_error:n { #1 }
  }
  \group_end:

  \__declpkg_seq_ginsert:Nn \g__declpkg_should_not_load_seq { #1 }
}
```

(*End definition for* `\__declpkg_should_load:nn` *and* `\__declpkg_should_not_load:nn`.)

`\__declpkg_add_pre_hook:nn`
`\__declpkg_add_post_hook:nn`

Add hooks to a package that will be ran before/after a package is loaded.

```
\cs_new_protected:Nn \__declpkg_add_pre_hook:nn
{
  \seq_if_in:NnTF \g__declpkg_loaded_seq { #1 }
  {
    \__declpkg_late_pre_hook_error:nn { #1 } { #2 }
  } {
    \seq_if_in:NnTF \g__declpkg_should_load_seq { #1 }
    {
      \tl_use:c { #2 }
    } {
      \__declpkg_seq_ginsert:cn { g__declpkg_ #1 _pre_hook_seq } { #2 }
    }
  }
}
\cs_generate_variant:Nn \__declpkg_add_pre_hook:nn { nV, nv, nx, Vn, VV, Vv, Vx, vn, vV, vv,

\cs_new_protected:Nn \__declpkg_add_post_hook:nn
{
  \seq_if_in:NnTF \g__declpkg_loaded_seq { #1 }
  {
    \tl_use:c { #2 }
  } {
    \__declpkg_seq_ginsert:cn { g__declpkg_ #1 _post_hook_seq } { #2 }
  }
}
\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.

```
\cs_new_protected:Nn \__declpkg_prep_load:n
{
  % add to should load
  \__declpkg_seq_ginsert:Nn \g__declpkg_should_load_seq { #1 }

  % run pre hooks
  \tl_gset:Nn \g__declpkg_tmpa_tl { g__declpkg_ #1 _pre_hook_seq }
  \seq_if_exist:cT { \tl_use:N \g__declpkg_tmpa_tl }
  {
    \seq_map_inline:cn { \tl_use:N \g__declpkg_tmpa_tl }
    {
      \tl_use:c { ##1 }
    }
  }

  % setup preorder graph
  \group_begin:
  \tl_set:Nn \l_tmpa_tl { g__declpkg_ #1 _load_before_seq }
  \seq_if_exist:cT { \tl_use:N \l_tmpa_tl }
  {
    \seq_map_inline:cn { \tl_use:N \l_tmpa_tl }
    {
      \__declpkg_seq_ginsert:cn { g__declpkg_ #1 _preorder_graph_children_seq } { ##1 }
    }
  }
  \group_end:

  \group_begin:
  \tl_set:Nn \l_tmpa_tl { g__declpkg_ #1 _load_after_seq }
  \seq_if_exist:cT { \tl_use:N \l_tmpa_tl }
  {
    \seq_map_inline:cn { \tl_use:N \l_tmpa_tl }
    {
      \__declpkg_seq_ginsert:cn { g__declpkg_ ##1 _preorder_graph_children_seq } { #1 }
    }
  }
  \group_end:

  % load coreqs
  \group_begin:
```

```
  \tl_set:Nn \l_tmpa_tl { g__declpkg_ #1 _coreq_seq }
  \seq_if_exist:cT { \tl_use:N \l_tmpa_tl }
  {
    \seq_map_inline:cn { \tl_use:N \l_tmpa_tl }
    {
      \__declpkg_should_load:nn { ##1 } { per~request~of~#1 }
    }
  }
  \group_end:

  % not load antireqs
  \group_begin:
  \tl_set:Nn \l_tmpa_tl { g__declpkg_ #1 _anticoreq_seq }
  \seq_if_exist:cT { \tl_use:N \l_tmpa_tl }
  {
    \seq_map_inline:cn { \tl_use:N \l_tmpa_tl }
    {
      \__declpkg_should_not_load:nn { ##1 } { per~request~of~#1 }
    }
  }
  \group_end:
}
\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.

```
\cs_new_protected:Nn \__declpkg_load:n
{
  % require package with options
  \group_begin:
  \seq_clear:N \l_tmpa_seq
  \tl_set:Nn \l_tmpa_tl { g__declpkg_ #1 _add_options_seq }
  \seq_if_exist:cT { \tl_use:N \l_tmpa_tl }
  {
    \seq_set_eq:Nc \l_tmpa_seq { \tl_use:N \l_tmpa_tl }
    \seq_remove_duplicates:N \l_tmpa_seq

    \tl_set:Nn \l_tmpa_tl { g__declpkg_ #1 _remove_options_seq }
    \seq_if_exist:cT { \tl_use:N \l_tmpa_tl }
    {
      \seq_map_inline:cn { \tl_use:N \l_tmpa_tl }
      {
        \seq_remove_all:Nn \l_tmpa_seq { ##1 }
      }
    }
  }

  \seq_if_empty:NF \tl_tmpa_seq
  {
    \PassOptionsToPackage { \seq_use:Nn \l_tmpa_seq { , } } { #1 }
  }
  \group_end:
  \RequirePackage { #1 }

  % add to loaded
  \__declpkg_seq_ginsert:Nn \g__declpkg_loaded_seq { #1 }

  % run post hooks
  \tl_gset:Nn \g__declpkg_tmpa_tl { g__declpkg_ #1 _post_hook_seq }
  \seq_if_exist:cT { \tl_use:N \g__declpkg_tmpa_tl }
  {
    \seq_map_inline:cn { \tl_use:N \g__declpkg_tmpa_tl }
    {
      \tl_use:c { ##1 }
    }
  }
}
\cs_generate_variant:Nn \__declpkg_load:n { V }
```

(*End definition for* `\__declpkg_load:n`*.*)

`\__declpkg_to_load:n`
`\__declpkg_to_load:nn`

Loads a package while making sure all dependencies are loaded first. The first arugment is {⟨*package*⟩} and {⟨*parent*⟩} is an optional extra parameter.

```
\cs_new_protected:Nn \__declpkg_to_load:n
{
  \__declpkg_seq_ginsert:NnF \g__declpkg_to_load_seq { #1 }
  {
    \__declpkg_dag_error:n { #1 }
  }

  \tl_gset:Nn \g__declpkg_tmpa_tl { g__declpkg_ #1 _preorder_graph_children_seq }
  \seq_if_exist:cT { \tl_use:N \g__declpkg_tmpa_tl }
  {
    \seq_map_inline:cn { \tl_use:N \g__declpkg_tmpa_tl }
    {
      \seq_if_in:NnF \g__declpkg_loaded_seq { ##1 }
      {
        \__declpkg_to_load:nn { ##1 } { #1 }
      }
    }
  }

  \__declpkg_load:n { #1 }
}

\cs_new_protected:Nn \__declpkg_to_load:nn
{
  \group_begin:
  \tl_set:Nn \l_tmpa_tl { g__declpkg_ #1 _to_load_parent_tl }
  \__declpkg_new_var:cn { \tl_use:N \l_tmpa_tl } { tl }
  \tl_gset:cn { \tl_use:N \l_tmpa_tl } { #2 }
  \group_end:

  \__declpkg_to_load:n { #1 }
}
```

(*End definition for* `\__declpkg_to_load:n` *and* `\__declpkg_to_load:nn`*.*)

__declpkg_include_packages:

```
\cs_new:Nn \__declpkg_include_packages:
{
  \bool_until_do:nn { \seq_if_empty_p:N \g__declpkg_should_load_queue_seq }
  {
    \group_begin:
    \seq_gpop_left:NN \g__declpkg_should_load_queue_seq \l__declpkg_tmpa_tl
    \tl_gset_eq:NN \g__declpkg_tmpa_tl \l__declpkg_tmpa_tl
    \group_end:

    \seq_if_in:NVF \g__declpkg_should_load_seq \g__declpkg_tmpa_tl
    {
      \__declpkg_prep_load:V \g__declpkg_tmpa_tl
    }
  }

  \seq_map_inline:Nn \g__declpkg_should_load_seq
  {
    \seq_if_in:NnF \g__declpkg_to_load_seq { ##1 }
    {
      \__declpkg_to_load:n { ##1 }
    }
  }
}
```

(*End definition for* __*declpkg_include_packages:*.)

3.7 Public interfaces

\declpkg The main command to declare a package.

```
\NewDocumentCommand \declpkg { s o m d[] d[] d[] d<> d() d() }
{
  \__declpkg_seq_ginsert:NnF \g__declpkg_package_seq { #3 }
  {
    \__declpkg_multiple_declare_error:n { #3 }
  }

  \IfBooleanTF { #1 }
  {
    \__declpkg_should_not_load:nn { #3 } { per~user~request }
  } {
    \__declpkg_should_load:nn { #3 } { per~user~request }
  }

  \IfValueT { #2 }
  {
    \group_begin:
    \clist_set:Nn \l_tmpa_clist { #2 }
    \clist_map_inline:Nn \l_tmpa_clist
    {
      \__declpkg_add_option:nn { #3 } { ##1 }
    }
    \group_end:
  }
```

```

\IfValueT { #4 }
{
  \group_begin:
  \clist_set:Nn \l_tmpa_clist { #4 }
  \clist_map_inline:Nn \l_tmpa_clist
  {
    \__declpkg_add_prereq:nn { #3 } { ##1 }
  }
  \group_end:
}

\IfValueT { #5 }
{
  \group_begin:
  \clist_set:Nn \l_tmpa_clist { #5 }
  \clist_map_inline:Nn \l_tmpa_clist
  {
    \__declpkg_add_postreq:nn { #3 } { ##1 }
  }
  \group_end:
}

\IfValueT { #6 }
{
  \group_begin:
  \clist_set:Nn \l_tmpa_clist { #6 }
  \clist_map_inline:Nn \l_tmpa_clist
  {
    \__declpkg_add_coreq:nn { #3 } { ##1 }
  }
  \group_end:
}


\IfValueT { #7 }
{
  \group_begin:
  \clist_set:Nn \l_tmpa_clist { #7 }
  \clist_map_inline:Nn \l_tmpa_clist
  {
    \__declpkg_add_antireq:nn { #3 } { ##1 }
  }
  \group_end:
}

\IfValueT { #8 }
{
  \group_begin:
  \clist_set:Nn \l_tmpa_clist { #8 }
  \clist_map_inline:Nn \l_tmpa_clist
  {
    \__declpkg_load_before:nn { #3 } { ##1 }
  }
```

```
    \group_end:
  }

  \IfValueT { #9 }
  {
    \group_begin:
    \clist_set:Nn \l_tmpa_clist { #9 }
    \clist_map_inline:Nn \l_tmpa_clist
    {
      \__declpkg_load_after:nn { #3 } { ##1 }
    }
    \group_end:
  }
}
```

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

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

```
\NewDocumentCommand \IncludePackages {} { \__declpkg_include_packages: }
```

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

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

```
\NewDocumentCommand \AddOptionsToPackage { m m }
{
  \group_begin:
  \clist_set:Nn \l_tmpa_clist { #1 }
  \clist_map_inline:Nn \l_tmpa_clist
  {
    \__declpkg_add_option:nn { #2 } { #1 }
  }
  \group_end:
}

\NewDocumentCommand \RemoveOptionsToPackage { m m }
{
  \group_begin:
  \clist_set:Nn \l_tmpa_clist { #1 }
  \clist_map_inline:Nn \l_tmpa_clist
  {
    \__declpkg_remove_option:nn { #2 } { #1 }
  }
  \group_end:
}
```

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

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

```
\NewDocumentCommand \AfterAll { m +m }
{
  \group_begin:
  \clist_set:Nn \l_tmpa_clist { #1 }
  \__declpkg_lambda:n { #2 }

  \bool_until_do:nn { \clist_if_empty_p:N \l_tmpa_clist }
  {
    \clist_pop:NN \l_tmpa_clist \l_tmpb_tl
    \__declpkg_lambda:x { \__declpkg_add_post_hook:nn { \tl_use:N \l_tmpb_tl } { \tl_use:N \g
  }
  \group_end:

  \tl_use:c { \tl_use:N \g__declpkg_last_lambda_tl }
}

\NewDocumentCommand \BeforeAll { m +m }
{
  \group_begin:
  \clist_set:Nn \l_tmpa_clist { #1 }
  \__declpkg_lambda:n { #2 }

  \bool_until_do:nn { \clist_if_empty_p:N \l_tmpa_clist }
  {
    \clist_pop:NN \l_tmpa_clist \l_tmpb_tl
    \__declpkg_lambda:x { \__declpkg_add_pre_hook:nn { \tl_use:N \l_tmpb_tl } { \tl_use:N \g_
  }
  \group_end:

  \tl_use:c { \tl_use:N \g__declpkg_last_lambda_tl }
}
```

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

`\AfterAny`
`\BeforeAny`

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

```
\NewDocumentCommand \AfterAny { m +m }
{
  \group_begin:
  \clist_set:Nn \l__declpkg_tmpa_clist { #1 }
  \__declpkg_transient_lambda:n { #2 }
  \tl_set_eq:NN \l__declpkg_tmpa_tl \g__declpkg_last_lambda_tl

  \bool_until_do:nn { \clist_if_empty_p:N \l__declpkg_tmpa_clist }
  {
    \clist_pop:NN \l__declpkg_tmpa_clist \l_tmpa_tl
    \tl_gset_eq:NN \g__declpkg_tmpa_tl \l_tmpa_tl
    \tl_gset_eq:NN \g__declpkg_tmpb_tl \l__declpkg_tmpa_tl
    \group_end:
    \__declpkg_add_post_hook:VV \g__declpkg_tmpa_tl \g__declpkg_tmpb_tl
    \group_begin:
  }
  \group_end:
}

\NewDocumentCommand \BeforeAny { m +m }
{
  \group_begin:
  \clist_set:Nn \l__declpkg_tmpa_clist { #1 }
  \__declpkg_transient_lambda:n { #2 }
  \tl_set_eq:NN \l__declpkg_tmpa_tl \g__declpkg_last_lambda_tl

  \bool_until_do:nn { \clist_if_empty_p:N \l__declpkg_tmpa_clist }
  {
    \clist_pop:NN \l__declpkg_tmpa_clist \l_tmpa_tl
    \tl_gset_eq:NN \g__declpkg_tmpa_tl \l_tmpa_tl
    \tl_gset_eq:NN \g__declpkg_tmpb_tl \l__declpkg_tmpa_tl
    \group_end:
    \__declpkg_add_pre_hook:VV \g__declpkg_tmpa_tl \g__declpkg_tmpb_tl
    \group_begin:
  }
  \group_end:
}
```

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

```
⟨/package⟩
```

Change History

v1.0
General: First public release 1

v1.1
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.