NAME

Module::Build::Bundling - How to bundle Module::Build with a distribution

SYNOPSIS

```
# Build.PL
use inc::latest 'Module::Build';

Module::Build->new(
module_name => 'Foo::Bar',
license => 'perl',
)->create_build_script;
```

DESCRIPTION

WARNING — THIS IS AN EXPERIMENTAL FEATURE

In order to install a distribution using Module::Build, users must have Module::Build available on their systems. There are two ways to do this. The first way is to include Module::Build in the `configure_requires` metadata field. This field is supported by recent versions CPAN and CPANPLUS and is a standard feature in the Perl core as of Perl 5.10.1. Module::Build now adds itself to `configure_requires` by default.

The second way supports older Perls that have not upgraded CPAN or CPANPLUS and involves bundling an entire copy of Module::Build into the distribution's `inc/` directory. This is the same approach used by Module::Install, a modern wrapper around ExtUtils::MakeMaker for Makefile.PL based distributions.

The "trick" to making this work for Module::Build is making sure the highest version Module::Build is used, whether this is in `inc/` or already installed on the user's system. This ensures that all necessary features are available as well as any new bug fixes. This is done using the new inc::latest module.

A "normal" Build.PL looks like this (with only the minimum required fields):

```
use Module::Build;

Module::Build->new(
module_name => 'Foo::Bar',
license => 'perl',
)->create_build_script;
```

A "bundling" Build.PL replaces the initial "use" line with a nearly transparent replacement:

```
use inc::latest 'Module::Build';

Module::Build->new(
module_name => 'Foo::Bar',
license => 'perl',
)->create_build_script;
```

For *authors*, when "Build dist" is run, Module::Build will be automatically bundled into `inc` according to the rules for inc::latest.

For *users*, inc::latest will load the latest Module::Build, whether installed or bundled in `inc/`.

BUNDLING OTHER CONFIGURATION DEPENDENCIES

The same approach works for other configuration dependencies — modules that *must* be available for Build.PL to run. All other dependencies can be specified as usual in the Build.PL and CPAN or CPANPLUS will install them after Build.PL finishes.

For example, to bundle the Devel::AssertOS::Unix module (which ensures a "Unix-like" operating system), one could do this:

```
use inc::latest 'Devel::AssertOS::Unix';
use inc::latest 'Module::Build';

Module::Build->new(
module_name => 'Foo::Bar',
license => 'perl',
)->create_build_script;
```

The `inc::latest` module creates bundled directories based on the packlist file of an installed distribution. Even though `inc::latest` takes module name arguments, it is better to think of it as bundling and making available entire *distributions*. When a module is loaded through `inc::latest` it looks in all bundled distributions in `inc/` for a newer module than can be found in the existing `@INC` array.

Thus, the module-name provided should usually be the "top-level" module name of a distribution, though this is not strictly required. For example, Module::Build has a number of heuristics to map module names to packlists, allowing users to do things like this:

```
use inc::latest 'Devel::AssertOS::Unix';
```

even though Devel::AssertOS::Unix is contained within the Devel-CheckOS distribution.

At the current time, packlists are required. Thus, bundling dual-core modules, *including Module::Build* may require a 'forced install' over versions in the latest version of perl in order to create the necessary packlist for bundling. This limitation will hopefully be addressed in a future version of Module::Build.

WARNING — How to Manage Dependency Chains

Before bundling a distribution you must ensure that all prerequisites are also bundled and load in the correct order. For Module::Build itself, this should not be necessary, but it is necessary for any other distribution. (A future release of Module::Build will hopefully address this deficiency.)

For example, if you need `Wibble`, but `Wibble` depends on `Wobble`, your Build.PL might look like this:

```
use inc::latest 'Wobble';
use inc::latest 'Wibble';
use inc::latest 'Module::Build';

Module::Build->new(
module_name => 'Foo::Bar',
license => 'perl',
)->create_build_script;
```

Authors are strongly suggested to limit the bundling of additional dependencies if at all possible and to carefully test their distribution tarballs on older versions of Perl before uploading to CPAN.

AUTHOR

David Golden <dagolden@cpan.org>

Development questions, bug reports, and patches should be sent to the Module-Build mailing list at <module-build@perl.org>.

Bug reports are also welcome at <http://rt.cpan.org/NoAuth/Bugs.html?Dist=Module-Build>.

SEE ALSO

perl(1), inc::latest, Module::Build(3), Module::Build::API(3), Module::Build::Cookbook(3),