

## NAME

CPANPLUS::Module

## SYNOPSIS

```
### get a module object from the CPANPLUS::Backend object
my $mod = $cb->module_tree('Some::Module');

### accessors
$mod->version;
$mod->package;

### methods
$mod->fetch;
$mod->extract;
$mod->install;
```

## DESCRIPTION

CPANPLUS::Module creates objects from the information in the source files. These can then be used to query and perform actions on, like fetching or installing.

These objects should only be created internally. For fake objects, there's the CPANPLUS::Module::Fake class. To obtain a module object consult the CPANPLUS::Backend documentation.

## CLASS METHODS

### accessors ()

Returns a list of all accessor methods to the object

## ACCESSORS

An objects of this class has the following accessors:

name

Name of the module.

module

Name of the module.

version

Version of the module. Defaults to '0.0' if none was provided.

path

Extended path on the mirror.

comment

Any comment about the module -- largely unused.

package

The name of the package.

description

Description of the module -- only registered modules have this.

dslip

The five character dslip string, that represents meta-data of the module -- again, only

registered modules have this.

status

The `CPANPLUS::Module::Status` object associated with this object. (see below).

author

The `CPANPLUS::Module::Author` object associated with this object.

parent

The `CPANPLUS::Internals` object that spawned this module object.

## STATUS ACCESSORS

CPANPLUS caches a lot of results from method calls and saves data it collected along the road for later reuse.

CPANPLUS uses this internally, but it is also available for the end user. You can get a status object by calling:

```
$modobj->status
```

You can then query the object as follows:

installer\_type

The installer type used for this distribution. Will be one of 'makemaker' or 'build'. This determines whether `CPANPLUS::Dist::MM` or `CPANPLUS::Dist::Build` will be used to build this distribution.

dist\_cpan

The dist object used to do the CPAN-side of the installation. Either a `CPANPLUS::Dist::MM` or `CPANPLUS::Dist::Build` object.

dist

The custom dist object used to do the operating specific side of the installation, if you've chosen to use this. For example, if you've chosen to install using the `ports` format, this may be a `CPANPLUS::Dist::Ports` object.

Undefined if you didn't specify a separate format to install through.

prereqs | requires

A hashref of prereqs this distribution was found to have. Will look something like this:

```
{ Carp => 0.01, strict => 0 }
```

Might be undefined if the distribution didn't have any prerequisites.

configure\_requires

Like `prereqs`, but these are necessary to be installed before the build process can even begin.

signature

Flag indicating, if a signature check was done, whether it was OK or not.

extract

The directory this distribution was extracted to.

fetch

The location this distribution was fetched to.

readme

The text of this distributions README file.

`uninstall`

Flag indicating if an `uninstall` call was done successfully.

`created`

Flag indicating if the `create` call to your dist object was done successfully.

`installed`

Flag indicating if the `install` call to your dist object was done successfully.

`checksums`

The location of this distributions CHECKSUMS file.

`checksum_ok`

Flag indicating if the checksums check was done successfully.

`checksum_value`

The checksum value this distribution is expected to have

## METHODS

### **`$self = CPANPLUS::Module->new( OPTIONS )`**

This method returns a `CPANPLUS::Module` object. Normal users should never call this method directly, but instead use the `CPANPLUS::Backend` to obtain module objects.

This example illustrates a `new( )` call with all required arguments:

```
CPANPLUS::Module->new(  
    module    => 'Foo',  
    path      => 'authors/id/A/AA/AAA',  
    package   => 'Foo-1.0.tgz',  
    author    => $author_object,  
    _id       => INTERNALS_OBJECT_ID,  
);
```

Every accessor is also a valid option to pass to `new`.

Returns a module object on success and false on failure.

### **`$mod->package_name( [$package_string] )`**

Returns the name of the package a module is in. For `Acme::Bleach` that might be `Acme-Bleach`.

### **`$mod->package_version( [$package_string] )`**

Returns the version of the package a module is in. For a module in the package `Acme-Bleach-1.1.tar.gz` this would be `1.1`.

### **`$mod->package_extension( [$package_string] )`**

Returns the suffix added by the compression method of a package a certain module is in. For a module in `Acme-Bleach-1.1.tar.gz`, this would be `tar.gz`.

### **`$mod->package_is_perl_core`**

Returns a boolean indicating of the package a particular module is in, is actually a core perl distribution.

**\$mod->module\_is\_supplied\_with\_perl\_core( [version => \$]] )**

Returns a boolean indicating whether ANY VERSION of this module was supplied with the current running perl's core package.

**\$mod->is\_bundle**

Returns a boolean indicating if the module you are looking at, is actually a bundle. Bundles are identified as modules whose name starts with `Bundle::`.

**\$mod->is\_autobundle;**

Returns a boolean indicating if the module you are looking at, is actually an autobundle as generated by `$cb->autobundle`.

**\$mod->is\_third\_party**

Returns a boolean indicating whether the package is a known third-party module (i.e. it's not provided by the standard Perl distribution and is not available on the CPAN, but on a third party software provider). See *Module::ThirdParty* for more details.

**\$mod->third\_party\_information**

Returns a reference to a hash with more information about a third-party module. See the documentation about `module_information()` in *Module::ThirdParty* for more details.

**\$clone = \$self->clone**

Clones the current module object for tinkering with. It will have a clean `CPANPLUS::Module::Status` object, as well as a fake `CPANPLUS::Module::Author` object.

**\$where = \$self->fetch**

Fetches the module from a CPAN mirror. Look at *CPANPLUS::Internals::Fetch::\_fetch()* for details on the options you can pass.

**\$path = \$self->extract**

Extracts the fetched module. Look at *CPANPLUS::Internals::Extract::\_extract()* for details on the options you can pass.

**\$type = \$self->get\_installer\_type([prefer\_makefile => BOOL])**

Gets the installer type for this module. This may either be `build` or `makemaker`. If `Module::Build` is unavailable or no installer type is available, it will fall back to `makemaker`. If both are available, it will pick the one indicated by your config, or by the `prefer_makefile` option you can pass to this function.

Returns the installer type on success, and false on error.

**\$dist = \$self->dist([target => 'prepare|create', format => DISTRIBUTION\_TYPE, args => {key => val}]);**

Create a distribution object, ready to be installed. Distribution type defaults to your config settings

The optional `args` hashref is passed on to the specific distribution types' `create` method after being dereferenced.

Returns a distribution object on success, false on failure.

See `CPANPLUS::Dist` for details.

**\$bool = \$mod->prepare( )**

Convenience method around `install()` that prepares a module without actually building it. This is equivalent to invoking `install` with `target` set to `prepare`

Returns true on success, false on failure.

## **\$bool = \$mod->create( )**

Convenience method around `install( )` that creates a module. This is equivalent to invoking `install` with `target` set to `create`

Returns true on success, false on failure.

## **\$bool = \$mod->test( )**

Convenience wrapper around `install( )` that tests a module, without installing it. It's the equivalent to invoking `install( )` with `target` set to `create` and `skiptest` set to 0.

Returns true on success, false on failure.

## **\$bool = \$self->install([ target => 'init|prepare|create|install', format => FORMAT\_TYPE, extractdir => DIRECTORY, fetchdir => DIRECTORY, prefer\_bin => BOOL, force => BOOL, verbose => BOOL, ..... ]);**

Installs the current module. This includes fetching it and extracting it, if this hasn't been done yet, as well as creating a distribution object for it.

This means you can pass it more arguments than described above, which will be passed on to the relevant methods as they are called.

See `CPANPLUS::Internals::Fetch`, `CPANPLUS::Internals::Extract` and `CPANPLUS::Dist` for details.

Returns true on success, false on failure.

Returns a list of module objects the Bundle specifies.

This requires you to have extracted the bundle already, using the `extract( )` method.

Returns false on error.

## **\$text = \$self->readme**

Fetches the readme belonging to this module and stores it under `$obj->status->readme`. Returns the readme as a string on success and returns false on failure.

## **\$version = \$self->installed\_version()**

Returns the currently installed version of this module, if any.

## **\$where = \$self->installed\_file()**

Returns the location of the currently installed file of this module, if any.

## **\$dir = \$self->installed\_dir()**

Returns the directory (or more accurately, the `@INC` handle) from which this module was loaded, if any.

## **\$bool = \$self->is\_uptodate([version => VERSION\_NUMBER])**

Returns a boolean indicating if this module is uptodate or not.

## **\$href = \$self->details()**

Returns a hashref with key/value pairs offering more information about a particular module. For example, for `Time::HiRes` it might look like this:

Author	Jarkko Hietaniemi (jhi@iki.fi)
Description	High resolution time, sleep, and alarm
Development Stage	Released
Installed File	/usr/local/perl/lib/Time/Hires.pm
Interface Style	plain Functions, no references used
Language Used	C and perl, a C compiler will be needed

Package	Time-HiRes-1.65.tar.gz
Public License	Unknown
Support Level	Developer
Version Installed	1.52
Version on CPAN	1.65

**@list = \$self->contains()**

Returns a list of module objects that represent the modules also present in the package of this module.

For example, for `Archive::Tar` this might return:

```
Archive::Tar
Archive::Tar::Constant
Archive::Tar::File
```

**@list\_of\_hrefs = \$self->fetch\_report()**

This function queries the CPAN testers database at <http://testers.cpan.org/> for test results of specified module objects, module names or distributions.

Look at `CPANPLUS::Internals::Report::_query_report()` for details on the options you can pass and the return value to expect.

**\$bool = \$self->uninstall([type => [all|man|prog]])**

This function uninstalls the specified module object.

You can install 2 types of files, either `man` pages or `program` files. Alternately you can specify `all` to uninstall both (which is the default).

Returns true on success and false on failure.

Do note that this does an uninstall via the so-called `.packlist`, so if you used a module installer like `say`, `ports` or `apt`, you should not use this, but use your package manager instead.

**@modobj = \$self->distributions()**

Returns a list of module objects representing all releases for this module on success, false on failure.

**@list = \$self->files ()**

Returns a list of files used by this module, if it is installed.

**@list = \$self->directory\_tree ()**

Returns a list of directories used by this module.

**@list = \$self->packlist ()**

Returns the `ExtUtils::Packlist` object for this module.

**@list = \$self->validate ()**

Returns a list of files that are missing for this modules, but are present in the `.packlist` file.

**\$bool = \$self->add\_to\_includepath;**

Adds the current modules path to `@INC` and `$PERL5LIB`. This allows you to add the module from its build dir to your path.

You can reset `@INC` and `$PERL5LIB` to its original state when you started the program, by calling:

```
$self->parent->flush('lib');
```

```
$path = $self->best_path_to_module_build();
```

#### **OBSOLETE**

If a newer version of `Module::Build` is found in your path, it will return this `special` path. If the newest version of `Module::Build` is found in your regular `@INC`, the method will return `false`. This indicates you do not need to add a special directory to your `@INC`.

Note that this is only relevant if you're building your own `CPANPLUS::Dist::*` plugin -- the built-in dist types already have this taken care of.

#### **BUG REPORTS**

Please report bugs or other issues to [<bug-cpanplus@rt.cpan.org>](mailto:bug-cpanplus@rt.cpan.org).

#### **AUTHOR**

This module by Jos Boumans [<kane@cpan.org>](mailto:kane@cpan.org).

#### **COPYRIGHT**

The CPAN++ interface (of which this module is a part of) is copyright (c) 2001 - 2007, Jos Boumans [<kane@cpan.org>](mailto:kane@cpan.org). All rights reserved.

This library is free software; you may redistribute and/or modify it under the same terms as Perl itself.