

## NAME

CPAN::FirstTime - Utility for CPAN::Config file Initialization

## SYNOPSIS

```
CPAN::FirstTime::init()
```

## DESCRIPTION

The init routine asks a few questions and writes a CPAN/Config.pm or CPAN/MyConfig.pm file (depending on what it is currently using).

In the following all questions and explanations regarding config variables are collected.

### auto\_commit

Normally CPAN.pm keeps config variables in memory and changes need to be saved in a separate 'o conf commit' command to make them permanent between sessions. If you set the 'auto\_commit' option to true, changes to a config variable are always automatically committed to disk.

Always commit changes to config variables to disk?

### build\_cache

CPAN.pm can limit the size of the disk area for keeping the build directories with all the intermediate files.

Cache size for build directory (in MB)?

### build\_dir

Directory where the build process takes place?

### build\_dir\_reuse

Until version 1.88 CPAN.pm never trusted the contents of the build\_dir directory between sessions. Since 1.88\_58 CPAN.pm has a YAML-based mechanism that makes it possible to share the contents of the build\_dir/ directory between different sessions with the same version of perl. People who prefer to test things several days before installing will like this feature because it saves a lot of time.

If you say yes to the following question, CPAN will try to store enough information about the build process so that it can pick up in future sessions at the same state of affairs as it left a previous session.

Store and re-use state information about distributions between CPAN.pm sessions?

### build\_requires\_install\_policy

When a module declares another one as a 'build\_requires' prerequisite this means that the other module is only needed for building or testing the module but need not be installed permanently. In this case you may wish to install that other module nonetheless or just keep it in the 'build\_dir' directory to have it available only temporarily. Installing saves time on future installations but makes the perl installation bigger.

You can choose if you want to always install (yes), never install (no) or be always asked. In the latter case you can set the default answer for the question to yes (ask/yes) or no (ask/no).

Policy on installing 'build\_requires' modules (yes, no, ask/yes, ask/no)?

### cache\_metadata

To considerably speed up the initial CPAN shell startup, it is possible to use Storable to create a cache of metadata. If Storable is not available, the normal index mechanism will be used.

Note: this mechanism is not used when use\_sqlite is on and SQLite is running.

Cache metadata (yes/no)?

### check\_sigs

CPAN packages can be digitally signed by authors and thus verified with the security provided by strong cryptography. The exact mechanism is defined in the `Module::Signature` module. While this is generally considered a good thing, it is not always convenient to the end user to install modules that are signed incorrectly or where the key of the author is not available or where some prerequisite for `Module::Signature` has a bug and so on.

With the `check_sigs` parameter you can turn signature checking on and off. The default is off for now because the whole tool chain for the functionality is not yet considered mature by some. The author of `CPAN.pm` would recommend setting it to true most of the time and turning it off only if it turns out to be annoying.

Note that if you do not have `Module::Signature` installed, no signature checks will be performed at all.

Always try to check and verify signatures if a `SIGNATURE` file is in the package and `Module::Signature` is installed (yes/no)?

### colorize\_output

When you have `Term::ANSIColor` installed, you can turn on colorized output to have some visual differences between normal `CPAN.pm` output, warnings, debugging output, and the output of the modules being installed. Set your favorite colors after some experimenting with the `Term::ANSIColor` module.

Do you want to turn on colored output?

### colorize\_print

Color for normal output?

### colorize\_warn

Color for warnings?

### colorize\_debug

Color for debugging messages?

### commandnumber\_in\_prompt

The prompt of the `cpan` shell can contain the current command number for easier tracking of the session or be a plain string.

Do you want the command number in the prompt (yes/no)?

### ftp\_passive

Shall we always set the `FTP_PASSIVE` environment variable when dealing with ftp download (yes/no)?

### getcwd

`CPAN.pm` changes the current working directory often and needs to determine its own current working directory. Per default it uses `Cwd::cwd` but if this doesn't work on your system for some reason, alternatives can be configured according to the following table:

<code>cwd</code>	<code>Cwd:::cwd</code>
<code>getcwd</code>	<code>Cwd:::getcwd</code>
<code>fastcwd</code>	<code>Cwd:::fastcwd</code>
<code>backtickcwd</code>	external command <code>cwd</code>

Preferred method for determining the current working directory?

### histfile

If you have one of the readline packages (`Term::ReadLine::Perl`, `Term::ReadLine::Gnu`, possibly others) installed, the interactive `CPAN` shell will have history support. The next two questions deal

with the filename of the history file and with its size. If you do not want to set this variable, please hit SPACE RETURN to the following question.

File to save your history?

histsize

Number of lines to save?

inactivity\_timeout

Sometimes you may wish to leave the processes run by CPAN alone without caring about them. Because the Makefile.PL or the Build.PL sometimes contains question you're expected to answer, you can set a timer that will kill a 'perl Makefile.PL' process after the specified time in seconds.

If you set this value to 0, these processes will wait forever. This is the default and recommended setting.

Timeout for inactivity during {Makefile,Build}.PL?

index\_expire

The CPAN indexes are usually rebuilt once or twice per hour, but the typical CPAN mirror mirrors only once or twice per day. Depending on the quality of your mirror and your desire to be on the bleeding edge, you may want to set the following value to more or less than one day (which is the default). It determines after how many days CPAN.pm downloads new indexes.

Let the index expire after how many days?

inhibit\_startup\_message

When the CPAN shell is started it normally displays a greeting message that contains the running version and the status of readline support.

Do you want to turn this message off?

keep\_source\_where

Unless you are accessing the CPAN on your filesystem via a file: URL, CPAN.pm needs to keep the source files it downloads somewhere. Please supply a directory where the downloaded files are to be kept.

Download target directory?

load\_module\_verbosity

When CPAN.pm loads a module it needs for some optional feature, it usually reports about module name and version. Choose 'v' to get this message, 'none' to suppress it.

Verbosity level for loading modules (none or v)?

makepl\_arg

Every Makefile.PL is run by perl in a separate process. Likewise we run 'make' and 'make install' in separate processes. If you have any parameters (e.g. PREFIX, LIB, UNINST or the like) you want to pass to the calls, please specify them here.

If you don't understand this question, just press ENTER.

Typical frequently used settings:

```
PREFIX=~/.perl # non-root users (please see manual for more hints)
```

Parameters for the 'perl Makefile.PL' command?

make\_arg

Parameters for the 'make' command? Typical frequently used setting:

```
-j3 # dual processor system (on GNU make)
```

Your choice:

make\_install\_arg

Parameters for the 'make install' command? Typical frequently used setting:

```
UNINST=1          # to always uninstall potentially conflicting files
```

Your choice:

make\_install\_make\_command

Do you want to use a different make command for 'make install'? Cautious people will probably prefer:

```
su root -c make
or
sudo make
or
/path1/to/sudo -u admin_account /path2/to/make
```

or some such. Your choice:

mbuildpl\_arg

A Build.PL is run by perl in a separate process. Likewise we run './Build' and './Build install' in separate processes. If you have any parameters you want to pass to the calls, please specify them here.

Typical frequently used settings:

```
--install_base /home/xxx          # different installation
directory
```

Parameters for the 'perl Build.PL' command?

mbuild\_arg

Parameters for the './Build' command? Setting might be:

```
--extra_linker_flags -L/usr/foo/lib # non-standard library location
```

Your choice:

mbuild\_install\_arg

Parameters for the './Build install' command? Typical frequently used setting:

```
--uninst 1          # uninstall conflicting files
```

Your choice:

mbuild\_install\_build\_command

Do you want to use a different command for './Build install'? Sudo users will probably prefer:

```
su root -c ./Build
or
sudo ./Build
or
/path1/to/sudo -u admin_account ./Build
```

or some such. Your choice:

pager

What is your favorite pager program?

### prefer\_installer

When you have Module::Build installed and a module comes with both a Makefile.PL and a Build.PL, which shall have precedence?

The main two standard installer modules are the old and well established ExtUtils::MakeMaker (for short: EUMM) which uses the Makefile.PL. And the next generation installer Module::Build (MB) which works with the Build.PL (and often comes with a Makefile.PL too). If a module comes only with one of the two we will use that one but if both are supplied then a decision must be made between EUMM and MB. See also <http://rt.cpan.org/Ticket/Display.html?id=29235> for a discussion about the right default.

Or, as a third option you can choose RAND which will make a random decision (something regular CPAN testers will enjoy).

In case you can choose between running a Makefile.PL or a Build.PL, which installer would you prefer (EUMM or MB or RAND)?

### prefs\_dir

CPAN.pm can store customized build environments based on regular expressions for distribution names. These are YAML files where the default options for CPAN.pm and the environment can be overridden and dialog sequences can be stored that can later be executed by an Expect.pm object. The CPAN.pm distribution comes with some prefab YAML files that cover sample distributions that can be used as blueprints to store one own prefs. Please check out the distroprefs/ directory of the CPAN.pm distribution to get a quick start into the prefs system.

Directory where to store default options/environment/dialogs for building modules that need some customization?

### prerequisites\_policy

The CPAN module can detect when a module which you are trying to build depends on prerequisites. If this happens, it can build the prerequisites for you automatically ('follow'), ask you for confirmation ('ask'), or just ignore them ('ignore'). Please set your policy to one of the three values.

Policy on building prerequisites (follow, ask or ignore)?

### randomize\_urllist

CPAN.pm can introduce some randomness when using hosts for download that are configured in the urllist parameter. Enter a numeric value between 0 and 1 to indicate how often you want to let CPAN.pm try a random host from the urllist. A value of one specifies to always use a random host as the first try. A value of zero means no randomness at all. Anything in between specifies how often, on average, a random host should be tried first.

Randomize parameter

### scan\_cache

By default, each time the CPAN module is started, cache scanning is performed to keep the cache size in sync. To prevent this, answer 'never'.

Perform cache scanning (atstart or never)?

### shell

What is your favorite shell?

### show\_unparsable\_versions

During the 'r' command CPAN.pm finds modules without version number. When the command finishes, it prints a report about this. If you want this report to be very verbose, say yes to the following variable.

Show all individual modules that have no \$VERSION?

#### show\_upload\_date

The 'd' and the 'm' command normally only show you information they have in their in-memory database and thus will never connect to the internet. If you set the 'show\_upload\_date' variable to true, 'm' and 'd' will additionally show you the upload date of the module or distribution. Per default this feature is off because it may require a net connection to get at the upload date.

Always try to show upload date with 'd' and 'm' command (yes/no)?

#### show\_zero\_versions

During the 'r' command CPAN.pm finds modules with a version number of zero. When the command finishes, it prints a report about this. If you want this report to be very verbose, say yes to the following variable.

Show all individual modules that have a \$VERSION of zero?

#### tar\_verbosity

When CPAN.pm uses the tar command, which switch for the verbosity shall be used? Choose 'none' for quiet operation, 'v' for file name listing, 'vv' for full listing.

Tar command verbosity level (none or v or vv)?

#### term\_is\_latin

The next option deals with the charset (aka character set) your terminal supports. In general, CPAN is English speaking territory, so the charset does not matter much but some CPAN have names that are outside the ASCII range. If your terminal supports UTF-8, you should say no to the next question. If it expects ISO-8859-1 (also known as LATIN1) then you should say yes. If it supports neither, your answer does not matter because you will not be able to read the names of some authors anyway. If you answer no, names will be output in UTF-8.

Your terminal expects ISO-8859-1 (yes/no)?

#### term\_ornaments

When using Term::ReadLine, you can turn ornaments on so that your input stands out against the output from CPAN.pm.

Do you want to turn ornaments on?

#### test\_report

The goal of the CPAN Testers project (<http://testers.cpan.org/>) is to test as many CPAN packages as possible on as many platforms as possible. This provides valuable feedback to module authors and potential users to identify bugs or platform compatibility issues and improves the overall quality and value of CPAN.

One way you can contribute is to send test results for each module that you install. If you install the CPAN::Reporter module, you have the option to automatically generate and email test reports to CPAN Testers whenever you run tests on a CPAN package.

See the CPAN::Reporter documentation for additional details and configuration settings. If your firewall blocks outgoing email, you will need to configure CPAN::Reporter before sending reports.

Email test reports if CPAN::Reporter is installed (yes/no)?

#### use\_sqlite

CPAN::SQLite is a layer between the index files that are downloaded from the CPAN and CPAN.pm that speeds up metadata queries and reduces memory consumption of CPAN.pm considerably.

Use CPAN::SQLite if available? (yes/no)?

#### yaml\_load\_code

Both YAML.pm and YAML::Syck are capable of deserialising code. As this requires a string eval, which might be a security risk, you can use this option to enable or disable the deserialisation of

code.Do you want to enable code deserialisation (yes/no)?

yaml\_module

At the time of this writing there are two competing YAML modules, YAML.pm and YAML::Syck. The latter is faster but needs a C compiler installed on your system. There may be more alternative YAML conforming modules but at the time of writing a potential third player, YAML::Tiny, seemed not powerful enough to work with CPAN.pm.

Which YAML implementation would you prefer?

## LICENSE

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