NAME

CPAN::Meta::History::Meta_1_4 - Version 1.4 metadata specification for META.yml

PREFACE

This is a historical copy of the version 1.4 specification for *META.yml* files, copyright by Ken Williams.

Modifications from the original:

- Various spelling corrections
- Include list of valid licenses from Module::Build 0.2807 rather than linking to the module.

SYNOPSIS

```
--- #YAML:1.0
name: Module-Build
abstract: Build and install Perl modules
version: 0.20
author:
- Ken Williams <kwilliams@cpan.org>
license: perl
distribution_type: module
requires:
Config: 0
Cwd: 0
Data::Dumper: 0
ExtUtils::Install: 0
File::Basename: 0
File::Compare: 0
File::Copy: 0
File::Find: 0
File::Path: 0
File::Spec: 0
IO::File: 0
perl: 5.005_03
recommends:
Archive::Tar: 1.00
ExtUtils::Install: 0.3
ExtUtils::ParseXS: 2.02
Pod::Text: 0
YAML: 0.35
build_requires:
Test: 0
resources:
  license: http://dev.perl.org/licenses/
meta-spec:
version: 1.4
  url: http://module-build.sourceforge.net/META-spec-v1.3.html
generated_by: Module::Build version 0.20
```

DESCRIPTION

This document describes version 1.4 of the *META.yml* specification.

The *META.yml* file describes important properties of contributed Perl distributions such as the ones found on CPAN. It is typically created by tools like Module::Build, Module::Install, and ExtUtils::MakeMaker.

The fields in the *META.yml* file are meant to be helpful for people maintaining module collections (like CPAN), for people writing installation tools (like CPAN.pm or CPANPLUS), or just for people

who want to know some stuff about a distribution before downloading it and starting to install it.

Note: The latest stable version of this specification can always be found at <http://module-build.sourceforge.net/META-spec-current.html>, and the latest development version (which may include things that won't make it into the stable version) can always be found at <http://module-build.sourceforge.net/META-spec-blead.html>.

FORMAT

META.yml files are written in the YAML format (see <http://www.yaml.org/>).

See the following links to learn why we chose YAML instead of, say, XML or Data::Dumper:

Module::Build design plans
<http://nntp.x.perl.org/group/perl.makemaker/406>

Not keen on YAML
<http://nntp.x.perl.org/group/perl.module-authors/1353>

META Concerns
<http://nntp.x.perl.org/group/perl.module-authors/1385>

TERMINOLOGY

distribution
: This is the primary object described by the *META.yml* specification. In the context of this document it usually refers to a collection of modules, scripts, and/or documents that are distributed together for other developers to use. Examples of distributions are `Class-Container`, `libwww-perl`, or `DBI`.

module
: This refers to a reusable library of code typically contained in a single file. Currently, we primarily talk of perl modules, but this specification should be open enough to apply to other languages as well (ex. python, ruby). Examples of modules are `Class::Container` `LWP::Simple` or `DBD::File`

HEADER

The first line of a *META.yml* file should be a valid YAML document header like `"--- #YAML:1.0"`.

FIELDS

The rest of the *META.yml* file is one big YAML mapping whose keys are described here.

meta-spec

Example:

```
meta-spec:
version: 1.4
  url: http://module-build.sourceforge.net/META-spec-v1.3.html
```

(Spec 1.1) [required] {URL} This field indicates the location of the version of the META.yml specification used.

name

Example:

```
name: Module-Build
```

(Spec 1.0) [required] {string} The name of the distribution which is often created by taking the ''main module'' in the distribution and changing ''::'' to ''-''. Sometimes it's completely different, however, as in the case of the libwww-perl distribution (see <http://search.cpan.org/dist/libwww-perl/>).

version

Example:

```
version: 0.20
```

(Spec 1.0) [required] {version} The version of the distribution to which the *META.yml* file refers.

abstract

Example:

```
abstract: Build and install Perl modules.
```

(Spec 1.1) [required] {string} A short description of the purpose of the distribution.

author

Example:

```
author:
- Ken Williams <kwilliams@cpan.org>
```

(Spec 1.1) [required] {list of strings} A YAML sequence indicating the author(s) of the distribution. The preferred form is author-name <email-address>.

license

Example:

```
license: perl
```

(Spec 1.0) [required] {string} The license under which this distribution may be used and redistributed.

Must be one of the following licenses:

apache
: The distribution is licensed under the Apache Software License (<http://opensource.org/licenses/apachepl.php>).

artistic
: The distribution is licensed under the Artistic License, as specified by the Artistic file in the standard perl distribution.

bsd
: The distribution is licensed under the BSD License (<http://www.opensource.org/licenses/bsd-license.php>).

gpl
: The distribution is licensed under the terms of the Gnu General Public License (<http://www.opensource.org/licenses/gpl-license.php>).

lgpl
: The distribution is licensed under the terms of the Gnu Lesser General Public License (<http://www.opensource.org/licenses/lgpl-license.php>).

mit
: The distribution is licensed under the MIT License (<http://opensource.org/licenses/mit-license.php>).

mozilla
: The distribution is licensed under the Mozilla Public License. (<http://opensource.org/licenses/mozilla1.0.php> or <http://opensource.org/licenses/mozilla1.1.php>)

open_source
: The distribution is licensed under some other Open Source Initiative-approved license listed at <http://www.opensource.org/licenses/>.

perl
: The distribution may be copied and redistributed under the same terms as perl itself (this is by far the most common licensing option for modules on CPAN). This is a dual license, in which the user may choose between either the GPL or the Artistic license.

restrictive
: The distribution may not be redistributed without special permission from the author and/or copyright holder.

unrestricted

The distribution is licensed under a license that is not approved by www.opensource.org <http://www.opensource.org/> but that allows distribution without restrictions.

distribution_type

Example:

```
distribution_type: module
```

(Spec 1.0) [optional] {string} What kind of stuff is contained in this distribution. Most things on CPAN are `module`s (which can also mean a collection of modules), but some things are `script`s.

Unfortunately this field is basically meaningless, since many distributions are hybrids of several kinds of things, or some new thing, or subjectively different in focus depending on who's using them. Tools like Module::Build and MakeMaker will likely stop generating this field.

requires

Example:

```
requires:
Data::Dumper: 0
File::Find: 1.03
```

(Spec 1.0) [optional] {map} A YAML mapping indicating the Perl prerequisites this distribution requires for proper operation. The keys are the names of the prerequisites (module names or 'perl'), and the values are version specifications as described in VERSION SPECIFICATIONS.

recommends

Example:

```
recommends:
Data::Dumper: 0
File::Find: 1.03
```

(Spec 1.0) [optional] {map} A YAML mapping indicating the Perl prerequisites this distribution recommends for enhanced operation. The keys are the names of the prerequisites (module names or 'perl'), and the values are version specifications as described in VERSION SPECIFICATIONS.

ALTERNATIVE: It may be desirable to present to the user which features depend on which modules so they can make an informed decision about which recommended modules to install.

Example:

```
optional_features:
foo:
description: Provides the ability to blah.
requires:
Data::Dumper: 0
File::Find: 1.03
```

(Spec 1.1) [optional] {map} A YAML mapping of names for optional features which are made available when its requirements are met. For each feature a description is provided along with any of ''requires'', ''build_requires'', and ''conflicts'', which have the same meaning in this subcontext as described elsewhere in this document.

build_requires

Example:

```
build_requires:
Data::Dumper: 0
File::Find: 1.03
```

(Spec 1.0) [optional] {map} A YAML mapping indicating the Perl prerequisites required for building and/or testing of this distribution. The keys are the names of the prerequisites (module names or 'perl'), and the values are version specifications as described in ''VERSION

SPECIFICATIONS''. These dependencies are not required after the distribution is installed.

configure_requires

Example:

```
configure_requires:
Module::Build: 0.2809
Data::Dumper: 0
File::Find: 1.03
```

(Spec 1.4) [optional] {map} A YAML mapping indicating the Perl prerequisites required before configuring this distribution. The keys are the names of the prerequisites (module names or 'perl'), and the values are version specifications as described in ''VERSION SPECIFICATIONS''. These dependencies are not required after the distribution is installed.

conflicts

Example:

```
conflicts:
Data::Dumper: 0
File::Find: 1.03
```

(Spec 1.0) [optional] {map} A YAML mapping indicating any items that cannot be installed while this distribution is installed. This is a pretty uncommon situation. The keys for `conflicts` are the item names (module names or 'perl'), and the values are version specifications as described in ''VERSION SPECIFICATIONS''.

dynamic_config

Example:

```
dynamic_config: 0
```

(Spec 1.0) [optional] {boolean} A boolean flag indicating whether a *Build.PL* or *Makefile.PL* (or similar) must be executed when building this distribution, or whether it can be built, tested and installed solely from consulting its metadata file. The main reason to set this to a true value is that your module performs some dynamic configuration (asking questions, sensing the environment, etc.) as part of its build/install process.

Currently Module::Build doesn't actually do anything with this flag - it's probably going to be up to higher-level tools like CPAN to do something useful with it. It can potentially bring lots of security, packaging, and convenience improvements.

If this field is omitted, it defaults to 1 (true).

private

(Deprecated) (Spec 1.0) [optional] {map} This field has been renamed to ''no_index''. See below.

provides

Example:

```
provides:
Foo::Bar:
file: lib/Foo/Bar.pm
version: 0.27_02
Foo::Bar::Blah:
file: lib/Foo/Bar/Blah.pm
Foo::Bar::Baz:
file: lib/Foo/Bar/Baz.pm
version: 0.3
```

(Spec 1.1) [optional] {map} A YAML mapping that describes all packages provided by this distribution. This information can be (and, in some cases, is) used by distribution and automation mechanisms like PAUSE, CPAN, and search.cpan.org to build indexes saying in which distribution

various packages can be found.

When using tools like `Module::Build` that can generate the `provides` mapping for your distribution automatically, make sure you examine what it generates to make sure it makes sense - indexers will usually trust the `provides` field if it's present, rather than scanning through the distribution files themselves to figure out packages and versions. This is a good thing, because it means you can use the `provides` field to tell the indexers precisely what you want indexed about your distribution, rather than relying on them to essentially guess what you want indexed.

no_index

Example:

```
no_index:
file:
- My/Module.pm
directory:
- My/Private
package:
- My::Module::Stuff
namespace:
- My::Module::Stuff
```

(Spec 1.1) [optional] {map} A YAML mapping that describes any files, directories, packages, and namespaces that are private (i.e. implementation artifacts) that are not of interest to searching and indexing tools. This is useful when no `provides` field is present.

For example, `search.cpan.org` excludes items listed in `no_index` when searching for POD, meaning files in these directories will not converted to HTML and made public - which is useful if you have example or test PODs that you don't want the search engine to go through.

file

(Spec 1.1) [optional] Exclude any listed file(s).

directory

(Spec 1.1) [optional] Exclude anything below the listed directory(ies).

[Note: previous editions of the spec had `dir` instead of `directory`, but I think MakeMaker and various users started using `directory`, so in deference we switched to that.]

package

(Spec 1.1) [optional] Exclude the listed package(s).

namespace

(Spec 1.1) [optional] Excludes anything below the listed namespace(s), but *not* the listed namespace(s) its self.

keywords

Example:

```
keywords:
- make
- build
- install
```

(Spec 1.1) [optional] {list} A sequence of keywords/phrases that describe this distribution.

resources

Example:

```
resources:
   license: http://dev.perl.org/licenses/
   homepage: http://sourceforge.net/projects/module-build
   bugtracker: http://rt.cpan.org/NoAuth/Bugs.html?Dist=Module-Build
   repository: http://sourceforge.net/cvs/?group_id=45731
   MailingList: http://lists.sourceforge.net/lists/listinfo/module-build-general
```

(Spec 1.1) [optional] {map} A mapping of any URL resources related to this distribution. All-lower-case keys, such as `homepage`, `license`, and `bugtracker`, are reserved by this specification, as they have ''official'' meanings defined here in this specification. If you'd like to add your own ''special'' entries (like the ''MailingList'' entry above), use at least one upper-case letter.

The current set of official keys is:

homepage
: The official home of this project on the web.

license
: An URL for an official statement of this distribution's license.

bugtracker
: An URL for a bug tracker (e.g. Bugzilla or RT queue) for this project.

generated_by

Example:

```
generated_by: Module::Build version 0.20
```

(Spec 1.0) [required] {string} Indicates the tool that was used to create this *META.yml* file. It's good form to include both the name of the tool and its version, but this field is essentially opaque, at least for the moment. If *META.yml* was generated by hand, it is suggested that the author be specified here.

[Note: My *meta_stats.pl* script which I use to gather statistics regarding *META.yml* usage prefers the form listed above, i.e. it splits on /s+versions+/ taking the first field as the name of the tool that generated the file and the second field as version of that tool. RWS]

VERSION SPECIFICATIONS

Some fields require a version specification (ex. ''requires'', ''recommends'', ''build_requires'', etc.) to indicate the particular version(s) of some other module that may be required as a prerequisite. This section details the version specification formats that are currently supported.

The simplest format for a version specification is just the version number itself, e.g. `2.4`. This means that **at least** version 2.4 must be present. To indicate that **any** version of a prerequisite is okay, even if the prerequisite doesn't define a version at all, use the version `0`.

You may also use the operators < (less than), <= (less than or equal), > (greater than), >= (greater than or equal), == (equal), and != (not equal). For example, the specification `< 2.0` means that any version of the prerequisite less than 2.0 is suitable.

For more complicated situations, version specifications may be AND-ed together using commas. The specification `>= 1.2, != 1.5, < 2.0` indicates a version that must be **at least** 1.2, **less than** 2.0, and **not equal to** 1.5.

SEE ALSO

CPAN, <http://www.cpan.org/>

CPAN.pm, <http://search.cpan.org/dist/CPAN/>

CPANPLUS, <http://search.cpan.org/dist/CPANPLUS/>

Data::Dumper, <http://search.cpan.org/dist/Data-Dumper/>

ExtUtils::MakeMaker, <http://search.cpan.org/dist/ExtUtils-MakeMaker/>

Module::Build, <http://search.cpan.org/dist/Module-Build/>

Module::Install, <http://search.cpan.org/dist/Module-Install/>

XML, <http://www.w3.org/XML/>

YAML, <http://www.yaml.org/>

HISTORY

March 14, 2003 (Pi day)

- Created version 1.0 of this document.

May 8, 2003

- Added the ''dynamic_config'' field, which was missing from the initial version.

November 13, 2003

- Added more YAML rationale articles.
- Fixed existing link to YAML discussion thread to point to new <http://nntp.x.perl.org/group/> site.
- Added and deprecated the ''private'' field.
- Added ''abstract'', ''configure'', ''requires_packages'', ''requires_os'', ''excludes_os'', and ''no_index'' fields.
- Bumped version.

November 16, 2003

- Added ''generation'', ''authored_by'' fields.
- Add alternative proposal to the ''recommends'' field.
- Add proposal for a ''requires_build_tools'' field.

December 9, 2003

- Added link to latest version of this specification on CPAN.
- Added section ''VERSION SPECIFICATIONS''.
- Chang name from Module::Build::META-spec to CPAN::META::Specification.
- Add proposal for ''auto_regenerate'' field.

December 15, 2003

- Add ''index'' field as a compliment to ''no_index''
- Add ''keywords'' field as a means to aid searching distributions.
- Add ''TERMINOLOGY'' section to explain certain terms that may be ambiguous.

July 26, 2005

- Removed a bunch of items (generation, requires_build_tools, requires_packages, configure, requires_os, excludes_os, auto_regenerate) that have never actually been supported, but were more like records of brainstorming.
- Changed `authored_by` to `author`, since that's always been what it's actually called in actual *META.yml* files.
- Added the ''=='' operator to the list of supported version-checking operators.
- Noted that the `distribution_type` field is basically meaningless, and shouldn't really be used.
- Clarified `dynamic_config` a bit.

August 23, 2005

- Removed the name `CPAN::META::Specification` since that implies a module that doesn't actually exist.

June 12, 2007

- Added `configure_requires`.