

## NAME

Parse::CPAN::Meta - Parse META.yml and META.json CPAN metadata files

## VERSION

version 1.4417

## SYNOPSIS

```
#####
# In your file

---
name: My-Distribution
version: 1.23
resources:
  homepage: "http://example.com/dist/My-Distribution"

#####
# In your program

use Parse::CPAN::Meta;

my $distmeta = Parse::CPAN::Meta->load_file('META.yml');

# Reading properties
my $name      = $distmeta->{name};
my $version   = $distmeta->{version};
my $homepage  = $distmeta->{resources}{homepage};
```

## DESCRIPTION

**Parse::CPAN::Meta** is a parser for *META.json* and *META.yml* files, using *JSON::PP* and/or *CPAN::Meta::YAML*.

**Parse::CPAN::Meta** provides three methods: `load_file`, `load_json_string`, and `load_yaml_string`. These will read and deserialize CPAN metafiles, and are described below in detail.

**Parse::CPAN::Meta** provides a legacy API of only two functions, based on the YAML functions of the same name. Wherever possible, identical calling semantics are used. These may only be used with YAML sources.

All error reporting is done with exceptions (`die`'ing).

Note that META files are expected to be in UTF-8 encoding, only. When converted string data, it must first be decoded from UTF-8.

## METHODS

### load\_file

```
my $metadata_structure = Parse::CPAN::Meta->load_file('META.json');

my $metadata_structure = Parse::CPAN::Meta->load_file('META.yml');
```

This method will read the named file and deserialize it to a data structure, determining whether it

should be JSON or YAML based on the filename. The file will be read using the ":utf8" IO layer.

### load\_yaml\_string

```
my $metadata_structure =  
Parse::CPAN::Meta->load_yaml_string($yaml_string);
```

This method deserializes the given string of YAML and returns the first document in it. (CPAN metadata files should always have only one document.) If the source was UTF-8 encoded, the string must be decoded before calling `load_yaml_string`.

### load\_json\_string

```
my $metadata_structure =  
Parse::CPAN::Meta->load_json_string($json_string);
```

This method deserializes the given string of JSON and the result. If the source was UTF-8 encoded, the string must be decoded before calling `load_json_string`.

### load\_string

```
my $metadata_structure = Parse::CPAN::Meta->load_string($some_string);
```

If you don't know whether a string contains YAML or JSON data, this method will use some heuristics and guess. If it can't tell, it assumes YAML.

### yaml\_backend

```
my $backend = Parse::CPAN::Meta->yaml_backend;
```

Returns the module name of the YAML serializer. See *ENVIRONMENT* for details.

### json\_backend

```
my $backend = Parse::CPAN::Meta->json_backend;
```

Returns the module name of the JSON serializer. This will either be *JSON::PP* or *JSON*. Even if `PERL_JSON_BACKEND` is set, this will return *JSON* as further delegation is handled by the *JSON* module. See *ENVIRONMENT* for details.

## FUNCTIONS

For maintenance clarity, no functions are exported by default. These functions are available for backwards compatibility only and are best avoided in favor of `load_file`.

### Load

```
my @yaml = Parse::CPAN::Meta::Load( $string );
```

Parses a string containing a valid YAML stream into a list of Perl data structures.

### LoadFile

```
my @yaml = Parse::CPAN::Meta::LoadFile( 'META.yml' );
```

Reads the YAML stream from a file instead of a string.

## ENVIRONMENT

### PERL\_JSON\_BACKEND

By default, *JSON::PP* will be used for deserializing JSON data. If the `PERL_JSON_BACKEND` environment variable exists, is true and is not "JSON::PP", then the *JSON* module (version 2.5 or

greater) will be loaded and used to interpret `PERL_JSON_BACKEND`. If `JSON` is not installed or is too old, an exception will be thrown.

## PERL\_YAML\_BACKEND

By default, `CPAN::Meta::YAML` will be used for deserializing YAML data. If the `PERL_YAML_BACKEND` environment variable is defined, then it is interpreted as a module to use for deserialization. The given module must be installed, must load correctly and must implement the `Load()` function or an exception will be thrown.

## SUPPORT

### Bugs / Feature Requests

Please report any bugs or feature requests through the issue tracker at <https://github.com/Perl-Toolchain-Gang/Parse-CPAN-Meta/issues>. You will be notified automatically of any progress on your issue.

### Source Code

This is open source software. The code repository is available for public review and contribution under the terms of the license.

<https://github.com/Perl-Toolchain-Gang/Parse-CPAN-Meta>

```
git clone https://github.com/Perl-Toolchain-Gang/Parse-CPAN-Meta.git
```

## AUTHORS

- Adam Kennedy <adamk@cpan.org>
- David Golden <dagolden@cpan.org>

## CONTRIBUTORS

- Graham Knop <haarg@haarg.org>
- Joshua ben Jore <jjore@cpan.org>
- Karen Etheridge <ether@cpan.org>
- Neil Bowers <neil@bowers.com>
- Ricardo Signes <rjbs@cpan.org>
- Steffen Mueller <smueller@cpan.org>

## COPYRIGHT AND LICENSE

This software is copyright (c) 2015 by Adam Kennedy and Contributors.

This is free software; you can redistribute it and/or modify it under the same terms as the Perl 5 programming language system itself.