

**NAME**

CPAN::Meta::Prereqs - a set of distribution prerequisites by phase and type

**VERSION**

version 2.150001

**DESCRIPTION**

A CPAN::Meta::Prereqs object represents the prerequisites for a CPAN distribution or one of its optional features. Each set of prereqs is organized by phase and type, as described in CPAN::Meta::Prereqs.

**METHODS****new**

```
my $prereq = CPAN::Meta::Prereqs->new( \%prereq_spec );
```

This method returns a new set of Prereqs. The input should look like the contents of the `prereqs` field described in [CPAN::Meta::Spec](#), meaning something more or less like this:

```
my $prereq = CPAN::Meta::Prereqs->new( {
  runtime => {
    requires => {
      'Some::Module' => '1.234',
      ....
    },
    ....
  },
  ....
});
```

You can also construct an empty set of prereqs with:

```
my $prereqs = CPAN::Meta::Prereqs->new;
```

This empty set of prereqs is useful for accumulating new prereqs before finally dumping the whole set into a structure or string.

**requirements\_for**

```
my $requirements = $prereqs->requirements_for( $phase, $type );
```

This method returns a [CPAN::Meta::Requirements](#) object for the given phase/type combination. If no prerequisites are registered for that combination, a new [CPAN::Meta::Requirements](#) object will be returned, and it may be added to as needed.

If `$phase` or `$type` are undefined or otherwise invalid, an exception will be raised.

**with\_merged\_prereqs**

```
my $new_prereqs = $prereqs->with_merged_prereqs( $other_prereqs );
```

```
my $new_prereqs = $prereqs->with_merged_prereqs( \@other_prereqs );
```

This method returns a new [CPAN::Meta::Prereqs](#) objects in which all the other prerequisites given are merged into the current set. This is primarily provided for combining a distribution's core prereqs with the prereqs of one of its optional features.

The new prereqs object has no ties to the originals, and altering it further will not alter them.

**merged\_requirements**

```
my $new_reqs = $prereqs->merged_requirements( \@phases, \@types );
```

```
my $new_reqs = $prereqs->merged_requirements( \@phases );
```

```
my $new_reqs = $preerqs->merged_requirements();
```

This method joins together all requirements across a number of phases and types into a new [CPAN::Meta::Requirements](#) object. If arguments are omitted, it defaults to “runtime”, “build” and “test” for phases and “requires” and “recommends” for types.

**as\_string\_hash**

This method returns a hashref containing structures suitable for dumping into a distmeta data structure. It is made up of hashes and strings, only; there will be no Prereqs, [CPAN::Meta::Requirements](#), or version objects inside it.

**is\_finalized**

This method returns true if the set of prereqs has been marked “finalized,” and cannot be altered.

**finalize**

Calling `finalize` on a Prereqs object will close it for further modification. Attempting to make any changes that would actually alter the prereqs will result in an exception being thrown.

**clone**

```
my $cloned_prereqs = $prereqs->clone;
```

This method returns a Prereqs object that is identical to the original object, but can be altered without affecting the original object. Finalization does not survive cloning, meaning that you may clone a finalized set of prereqs and then modify the clone.

**BUGS**

Please report any bugs or feature using the CPAN Request Tracker. Bugs can be submitted through the web interface at <http://rt.cpan.org/Dist/Display.html?Queue=CPAN-Meta>

When submitting a bug or request, please include a test-file or a patch to an existing test-file that illustrates the bug or desired feature.

**AUTHORS**

- David Golden <dagolden@cpan.org>
- Ricardo Signes <rjbs@cpan.org>

**COPYRIGHT AND LICENSE**

This software is copyright (c) 2010 by David Golden and Ricardo Signes.

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