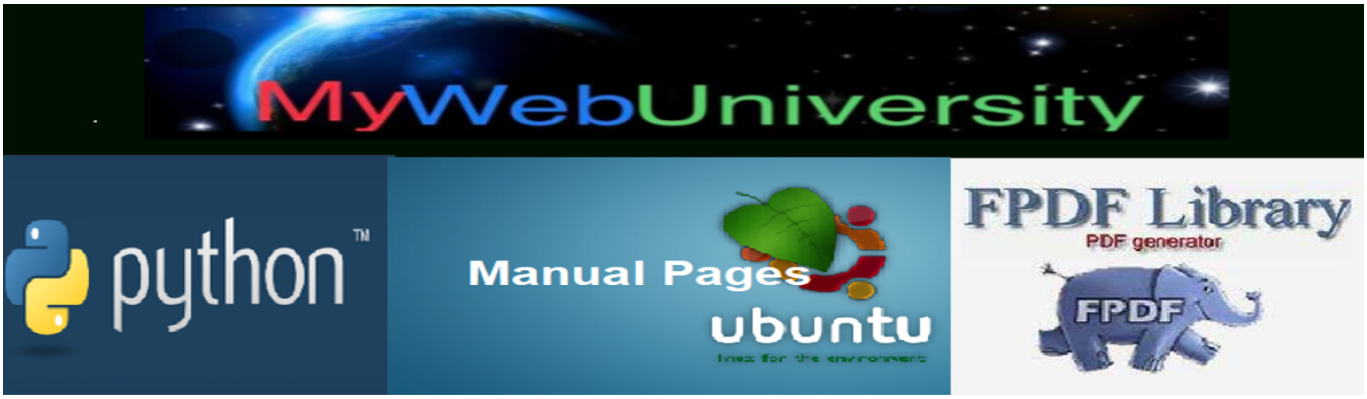

Rocky Enterprise Linux 9.2 Manual Pages on command 'Test2::Transition.3perl'

$ man Test2::Transition.3perl

Test2::Transition(3perl) Perl Programmers Reference Guide Test2::Transition(3perl)

NAME

Test2::Transition - Transition notes when upgrading to Test2

DESCRIPTION

This is where gotchas and breakages related to the Test2 upgrade are documented. The upgrade causes Test::Builder to defer to Test2 under the hood. This transition is mostly transparent, but there are a few cases that can trip you up.

THINGS THAT BREAK

This is the list of scenarios that break with the new internals.

Test::Builder1.5/2 conditionals

The Problem

a few years back there were two attempts to upgrade/replace Test::Builder. Confusingly these were called Test::Builder2 and Test::Builder1.5, in that order. Many people put conditionals in their code to check the Test::Builder version number and adapt their code accordingly.

The Test::Builder2/1.5 projects both died out. Now the conditional code people added has become a mine field. A vast majority of modules broken by Test2 fall into this category.

The Fix

The fix is to remove all Test::Builder1.5/2 related code. Either use the legacy Test::Builder API, or use Test2 directly.

Replacing the Test::Builder singleton

The Problem

Some test modules would replace the Test::Builder singleton instance with their own

instance or subclass. This was usually done to intercept or modify results as they happened.

The Test::Builder singleton is now a simple compatibility wrapper around Test2. The Test::Builder singleton is no longer the central place for results. Many results bypass the Test::Builder singleton completely, which breaks and behavior intended when replacing the singleton.

The Fix

If you simply want to intercept all results instead of letting them go to TAP, you should look at the Test2::API docs and read about pushing a new hub onto the hub stack. Replacing the hub temporarily is now the correct way to intercept results.

If your goal is purely monitoring of events use the "Test2::Hub->listen()" method exported by Test::More to watch events as they are fired. If you wish to modify results before they go to TAP look at the "Test2::Hub->filter()" method.

Directly Accessing Hash Elements

The Problem

Some modules look directly at hash keys on the Test::Builder singleton. The problem here is that the Test::Builder singleton no longer holds anything important.

The Fix

The fix is to use the API specified in Test2::API to look at or modify state as needed.

Subtest indentation

The Problem

An early change, in fact the change that made Test2 an idea, was a change to the indentation of the subtest note. It was decided it would be more readable to outdent the subtest note instead of having it inline with the subtest:

```
    # subtest foo
        ok 1 - blah
        1..1
    ok 1 - subtest foo
```

The old style indented the note:

```
        # subtest foo
        ok 1 - blah
        1..1
    ok 1 - subtest foo
```

This breaks tests that do string comparison of TAP output.

The Fix

```
my $indent = $INC{'Test2/API.pm'} ? '' : '    ';
is(
    $subtest_output,
    "${indent}# subtest foo",
    "Got subtest note"
);
```

Check if $INC{'Test2/API.pm'} is set, if it is then no indentation should be expected. If it is not set, then the old Test::Builder is in use, indentation should be expected.

DISTRIBUTIONS THAT BREAK OR NEED TO BE UPGRADED

This is a list of cpan modules that have been known to have been broken by the upgrade at one point.

WORKS BUT TESTS WILL FAIL

These modules still function correctly, but their test suites will not pass. If you already have these modules installed then you can continue to use them. If you are trying to install them after upgrading Test::Builder you will need to force installation, or bypass the broken tests.

Test::DBIx::Class::Schema

This module has a test that appears to work around a Test::Builder bug. The bug appears to have been fixed by Test2, which means the workaround causes a failure. This can be easily updated, but nobody has done so yet.

Known broken in versions: 1.0.9 and older

Device::Chip

Tests break due to subtest indentation.

Known broken in version 0.07. Apparently works fine in 0.06 though. Patch has been submitted to fix the issue.

UPGRADE SUGGESTED

These are modules that did not break, but had broken test suites that have since been fixed.

Test::Exception

Old versions work fine, but have a minor test name behavior that breaks with Test2.

Old versions will no longer install because of this. The latest version on CPAN will

install just fine. Upgrading is not required, but is recommended.

Fixed in version: 0.43

Data::Peek

Some tests depended on $! and $? being modified in subtle ways. A patch was applied to correct things that changed.

The module itself works fine, there is no need to upgrade.

Fixed in version: 0.45

circular::require

Some tests were fragile and required base.pm to be loaded at a late stage. Test2 was loading base.pm too early. The tests were updated to fix this.

The module itself never broke, you do not need to upgrade.

Fixed in version: 0.12

Test::Module::Used

A test worked around a now-fixed planning bug. There is no need to upgrade if you have an old version installed. New versions install fine if you want them.

Fixed in version: 0.2.5

Test::Moose::More

Some tests were fragile, but have been fixed. The actual breakage was from the subtest comment indentation change.

No need to upgrade, old versions work fine. Only new versions will install.

Fixed in version: 0.025

Test::FITesque

This was broken by a bugfix to how planning is done. The test was updated after the bugfix.

Fixed in version: 0.04

Test::Kit

Old versions work fine, but would not install because Test::Aggregate was in the dependency chain. An upgrade should not be needed.

Fixed in version: 2.15

autouse

A test broke because it depended on Scalar::Util not being loaded. Test2 loads Scalar::Util. The test was updated to load Test2 after checking Scalar::Util's load status.

There is no need to upgrade if you already have it installed.

Fixed in version: 1.11

NEED TO UPGRADE

Test::SharedFork

Old versions need to directly access Test::Builder singleton hash elements. The latest version on CPAN will still do this on old Test::Builder, but will defer to Test2::IPC on Test2.

Fixed in version: 0.35

Test::Builder::Clutch

This works by doing overriding methods on the singleton, and directly accessing hash values on the singleton. A new version has been released that uses the Test2 API to accomplish the same result in a saner way.

Fixed in version: 0.07

Test::Dist::VersionSync

This had Test::Builder2 conditionals. This was fixed by removing the conditionals.

Fixed in version: 1.1.4

Test::Modern

This relied on "Test::Builder->_try()" which was a private method, documented as something nobody should use. This was fixed by using a different tool.

Fixed in version: 0.012

Test::UseAllModules

Version 0.14 relied on "Test::Builder->history" which was available in Test::Builder 1.5. Versions 0.12 and 0.13 relied on other Test::Builder internals.

Fixed in version: 0.15

Test::More::Prefix

Worked by applying a role that wrapped "Test::Builder->_print_comment". Fixed by adding an event filter that modifies the message instead when running under Test2.

Fixed in version: 0.007

STILL BROKEN

Test::Aggregate

This distribution directly accesses the hash keys in the Test::Builder singleton. It also approaches the problem from the wrong angle, please consider using Test2::Aggregate for similar functionality and Test2::Harness which allows module

preloading at the harness level.

Still broken as of version: 0.373

Test::Wrapper

This module directly uses hash keys in the Test::Builder singleton. This module is also obsolete thanks to the benefits of Test2. Use "intercept()" from Test2::API to achieve a similar result.

Still broken as of version: 0.3.0

Test::ParallelSubtest

This module overrides "Test::Builder::subtest()" and "Test::Builder::done_testing()". It also directly accesses hash elements of the singleton. It has not yet been fixed.

Alternatives: Test2::AsyncSubtest and Test2::Workflow (not stable).

Still broken as of version: 0.05

Test::Pretty

See https://github.com/tokuhirom/Test-Pretty/issues/25

The author admits the module is crazy, and he is awaiting a stable release of something new (Test2) to completely rewrite it in a sane way.

Still broken as of version: 0.32

Net::BitTorrent

The tests for this module directly access Test::Builder hash keys. Most, if not all of these hash keys have public API methods that could be used instead to avoid the problem.

Still broken in version: 0.052

Test::Group

It monkeypatches Test::Builder, and calls it "black magic" in the code.

Still broken as of version: 0.20

Test::Flatten

This modifies the Test::Builder internals in many ways. A better was to accomplish the goal of this module is to write your own subtest function.

Still broken as of version: 0.11

Log::Dispatch::Config::TestLog

Modifies Test::Builder internals.

Still broken as of version: 0.02

Test::Able

Modifies Test::Builder internals.

Still broken as of version: 0.11

MAKE ASSERTIONS -> SEND EVENTS

LEGACY

```
use Test::Builder;
# A majority of tools out there do this:
# my $TB = Test::Builder->new;
# This works, but has always been wrong, forcing Test::Builder to implement
# subtests as a horrific hack. It also causes problems for tools that try
# to replace the singleton (also discouraged).
sub my_ok($;$) {
    my ($bool, $name) = @_;
    my $TB = Test::Builder->new;
    $TB->ok($bool, $name);
}
sub my_diag($) {
    my ($msg) = @_;
    my $TB = Test::Builder->new;
    $TB->diag($msg);
}
```

TEST2

```
use Test2::API qw/context/;
sub my_ok($;$) {
    my ($bool, $name) = @_;
    my $ctx = context();
    $ctx->ok($bool, $name);
    $ctx->release;
}
sub my_diag($) {
    my ($msg) = @_;
    my $ctx = context();
    $ctx->diag($msg);
    $ctx->release;
```

```
    }
```

The context object has API compatible implementations of the following methods:

ok($bool, $name)

diag(@messages)

note(@messages)

subtest($name, $code)

If you are looking for helpers with "is", "like", and others, see Test2::Suite.

WRAP EXISTING TOOLS

LEGACY

```
use Test::More;
sub exclusive_ok {
    my ($bool1, $bool2, $name) = @_;
    # Ensure errors are reported 1 level higher
    local $Test::Builder::Level = $Test::Builder::Level + 1;
    $ok = $bool1 || $bool2;
    $ok &&= !($bool1 && $bool2);
    ok($ok, $name);
    return $bool;
}
```

Every single tool in the chain from this, to "ok", to anything "ok" calls needs to increment the $Level variable. When an error occurs Test::Builder will do a trace to the stack frame determined by $Level, and report that file+line as the one where the error occurred. If you or any other tool you use forgets to set $Level then errors will be reported to the wrong place.

TEST2

```
use Test::More;
sub exclusive_ok {
    my ($bool1, $bool2, $name) = @_;
    # Grab and store the context, even if you do not need to use it
    # directly.
    my $ctx = context();
    $ok = $bool1 || $bool2;
    $ok &&= !($bool1 && $bool2);
```

```
    ok($ok, $name);
    $ctx->release;
    return $bool;
}
```

Instead of using $Level to perform a backtrace, Test2 uses a context object. In this sample you create a context object and store it. This locks the context (errors report 1 level up from here) for all wrapped tools to find. You do not need to use the context object, but you do need to store it in a variable. Once the sub ends the $ctx variable is destroyed which lets future tools find their own.

USING UTF8

LEGACY

```
# Set the mode BEFORE anything loads Test::Builder
use open ':std', ':encoding(utf8)';
use Test::More;
```

Or

```
# Modify the filehandles
my $builder = Test::More->builder;
binmode $builder->output,         ":encoding(utf8)";
binmode $builder->failure_output, ":encoding(utf8)";
binmode $builder->todo_output,    ":encoding(utf8)";
```

TEST2

```
use Test2::API qw/test2_stack/;
test2_stack->top->format->encoding('utf8');
```

Though a much better way is to use the Test2::Plugin::UTF8 plugin, which is part of Test2::Suite.

AUTHORS, CONTRIBUTORS AND REVIEWERS

The following people have all contributed to this document in some way, even if only for review.

Chad Granum (EXODIST) <exodist@cpan.org>

SOURCE

The source code repository for Test2 can be found at http://github.com/Test-More/test-more/.

MAINTAINER

Chad Granum <exodist@cpan.org>

COPYRIGHT