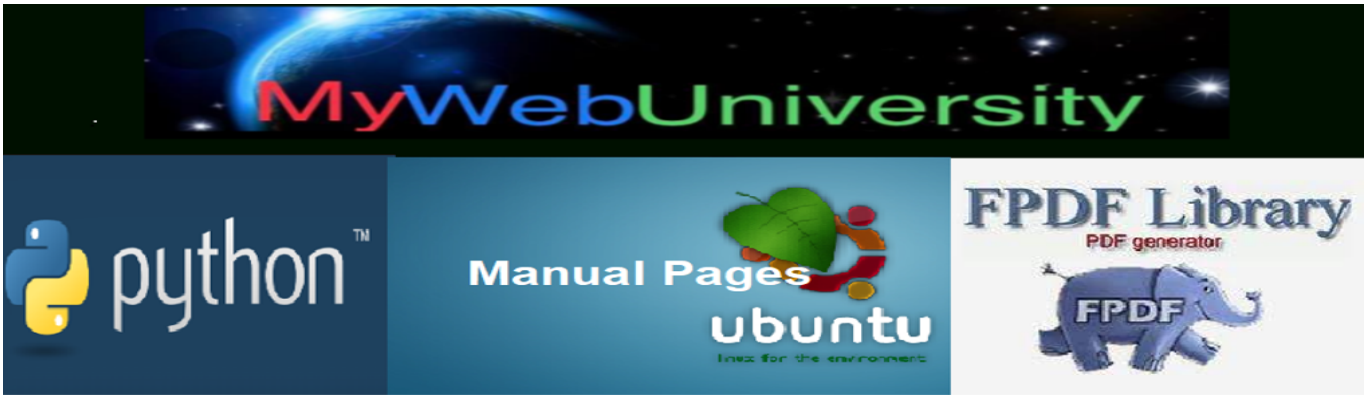

Rocky Enterprise Linux 9.2 Manual Pages on command 'Test2::API::Instance.3perl'

$ man Test2::API::Instance.3perl

NAME

Test2::API::Instance - Object used by Test2::API under the hood

DESCRIPTION

This object encapsulates the global shared state tracked by Test2. A single global instance of this package is stored (and obscured) by the Test2::API package.

There is no reason to directly use this package. This package is documented for completeness. This package can change, or go away completely at any time. Directly using, or monkeypatching this package is not supported in any way shape or form.

SYNOPSIS

```
use Test2::API::Instance;

my $obj = Test2::API::Instance->new;
```

$pid = $obj->pid

PID of this instance.

$obj->tid

Thread ID of this instance.

$obj->reset()

Reset the object to defaults.

$obj->load()

Set the internal state to loaded, and run and stored post-load callbacks.

$bool = $obj->loaded

Check if the state is set to loaded.

$arrayref = $obj->post_load_callbacks

Get the post-load callbacks.

$obj->add_post_load_callback(sub { ... })

Add a post-load callback. If "load()" has already been called then the callback will be immediately executed. If "load()" has not been called then the callback will be stored and executed later when "load()" is called.

$hashref = $obj->contexts()

Get a hashref of all active contexts keyed by hub id.

$arrayref = $obj->context_acquire_callbacks

Get all context acquire callbacks.

$arrayref = $obj->context_init_callbacks

Get all context init callbacks.

$arrayref = $obj->context_release_callbacks

Get all context release callbacks.

$arrayref = $obj->pre_subtest_callbacks

Get all pre-subtest callbacks.

$obj->add_context_init_callback(sub { ... })

Add a context init callback. Subs are called every time a context is created. Subs get the newly created context as their only argument.

$obj->add_context_release_callback(sub { ... })

Add a context release callback. Subs are called every time a context is released. Subs get the released context as their only argument. These callbacks should not call release on the context.

$obj->add_pre_subtest_callback(sub { ... })

Add a pre-subtest callback. Subs are called every time a subtest is going to be run. Subs get the subtest name, coderef, and any arguments.

$obj->set_exit()

This is intended to be called in an "END { ... }" block. This will look at test state and set $?. This will also call any end callbacks, and wait on child processes/threads.

$obj->set_ipc_pending($val)

Tell other processes and threads there is a pending event. $val should be a unique value no other thread/process will generate.

Note: This will also make the current process see a pending event.

$pending = $obj->get_ipc_pending()

This returns -1 if it is not possible to know.

This returns 0 if there are no pending events.

This returns 1 if there are pending events.

$timeout = $obj->ipc_timeout;

$obj->set_ipc_timeout($timeout);

How long to wait for child processes and threads before aborting.

$drivers = $obj->ipc_drivers

Get the list of IPC drivers.

$obj->add_ipc_driver($DRIVER_CLASS)

Add an IPC driver to the list. The most recently added IPC driver will become the global one during initialization. If a driver is added after initialization has occurred a warning will be generated:

"IPC driver $driver loaded too late to be used as the global ipc driver"

$bool = $obj->ipc_polling

Check if polling is enabled.

$obj->enable_ipc_polling

Turn on polling. This will cull events from other processes and threads every time a context is created.

$obj->disable_ipc_polling

Turn off IPC polling.

$bool = $obj->no_wait

$bool = $obj->set_no_wait($bool)

Get/Set no_wait. This option is used to turn off process/thread waiting at exit.

$arrayref = $obj->exit_callbacks

Get the exit callbacks.

$obj->add_exit_callback(sub { ... })

Add an exit callback. This callback will be called by "set_exit()".

$bool = $obj->finalized

Check if the object is finalized. Finalization happens when either "ipc()", "stack()", or "format()" are called on the object. Once finalization happens these fields are considered unchangeable (not enforced here, enforced by Test2).

$ipc = $obj->ipc

Get the one true IPC instance.

$obj->ipc_disable

Turn IPC off

$bool = $obj->ipc_disabled

Check if IPC is disabled

$stack = $obj->stack

Get the one true hub stack.

$formatter = $obj->formatter

Get the global formatter. By default this is the 'Test2::Formatter::TAP' package. This could be any package that implements the "write()" method. This can also be an instantiated object.

$bool = $obj->formatter_set()

Check if a formatter has been set.

$obj->add_formatter($class)

$obj->add_formatter($obj)

Add a formatter. The most recently added formatter will become the global one during initialization. If a formatter is added after initialization has occurred a warning will be generated:

"Formatter $formatter loaded too late to be used as the global formatter"

$obj->set_add_uuid_via(sub { ... })

$sub = $obj->add_uuid_via()

This allows you to provide a UUID generator. If provided UUIDs will be attached to all events, hubs, and contexts. This is useful for storing, tracking, and linking these objects.

The sub you provide should always return a unique identifier. Most things will expect a proper UUID string, however nothing in Test2::API enforces this.

The sub will receive exactly 1 argument, the type of thing being tagged 'context', 'hub', or 'event'. In the future additional things may be tagged, in which case new strings will be passed in. These are purely informative, you can (and usually should) ignore them.

SOURCE

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

MAINTAINERS

Chad Granum <exodist@cpan.org>

AUTHORS

Chad Granum <exodist@cpan.org>

COPYRIGHT