Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 22.4.5 Manual Pages on command 'Log::Any::Adapter.3pm'

$ man Log::Any::Adapter.3pm

Log::Any::Adapter(3pm) User Contributed Perl Documentation Log::Any::Adapter(3pm)

NAME

Log::Any::Adapter - Tell Log::Any where to send its logs

VERSION

version 1.708

SYNOPSIS

```
# Log to a file, or stdout, or stderr for all categories
#
use Log::Any::Adapter ('File', '/path/to/file.log');
use Log::Any::Adapter ('Stdout');
use Log::Any::Adapter ('Stderr');
# Use Log::Log4perl for all categories
#
Log::Log4perl::init('/etc/log4perl.conf');
Log::Any::Adapter->set('Log4perl');
# Use Log::Dispatch for Foo::Baz
#
use Log::Dispatch;
my $log = Log::Dispatch->new(outputs => [[ ... ]]);
Log::Any::Adapter->set( { category => 'Foo::Baz' },
    'Dispatch', dispatcher => $log );
# Use Log::Dispatch::Config for Foo::Baz and its subcategories
```

```
    #
    use Log::Dispatch::Config;
    Log::Dispatch::Config->configure('/path/to/log.conf');
    Log::Any::Adapter->set(
      { category => qr/^Foo::Baz/ },
      'Dispatch', dispatcher => Log::Dispatch::Config->instance() );
    # Use your own adapter for all categories
    #
    Log::Any::Adapter->set('+My::Log::Any::Adapter', ...);
```

DESCRIPTION

Log::Any::Adapter connects log producers and log consumers. Its methods instantiate a logging adapter (a subclass of Log::Any::Adapter::Base) and route log messages from one or more categories to it.

ADAPTERS

In order to use a logging mechanism with "Log::Any", there needs to be an adapter class for it. Typically this is named Log::Any::Adapter::something.

Adapters in this distribution

Three basic adapters come with this distribution -- Log::Any::Adapter::File, Log::Any::Adapter::Stdout and Log::Any::Adapter::Stderr:

```
    use Log::Any::Adapter ('File', '/path/to/file.log');
    use Log::Any::Adapter ('Stdout');
    use Log::Any::Adapter ('Stderr');
    # or
    use Log::Any::Adapter;
    Log::Any::Adapter->set('File', '/path/to/file.log');
    Log::Any::Adapter->set('Stdout');
    Log::Any::Adapter->set('Stderr');
```

All of them simply output the message and newline to the specified destination; a datestamp prefix is added in the "File" case. For anything more complex you'll want to use a more robust adapter from CPAN.

Adapters on CPAN

A sampling of adapters available on CPAN as of this writing:

? Log::Any::Adapter::Log4perl

? Log::Any::Adapter::Dispatch

? Log::Any::Adapter::FileHandle

? Log::Any::Adapter::Syslog

You may find other adapters on CPAN by searching for "Log::Any::Adapter", or create your own adapter. See Log::Any::Adapter::Development for more information on the latter.

SETTING AND REMOVING ADAPTERS

Log::Any::Adapter->set ([options,]adapter_name, adapter_params...)

This method sets the adapter to use for all log categories, or for a particular set of categories.

adapter_name is the name of an adapter. It is automatically prepended with "Log::Any::Adapter::". If instead you want to pass the full name of an adapter, prefix it with a "+". e.g.

```
# Use My::Adapter class
Log::Any::Adapter->set('+My::Adapter', arg => $value);
```

adapter_params are passed along to the adapter constructor. See the documentation for the individual adapter classes for more information.

An optional hash of options may be passed as the first argument. Options are:

category

A string containing a category name, or a regex (created with "qr//") matching multiple categories. If not specified, all categories will be routed to the adapter.

lexically

A reference to a lexical variable. When the variable goes out of scope, the adapter setting will be removed. e.g.

```
{
    Log::Any::Adapter->set({lexically => \my $lex}, ...);
    # in effect here
    ...
}
# no longer in effect here
```

"set" returns an entry object, which can be passed to "remove". If you call "set" repeatedly without calling "remove" you will leak memory. For most

programs that set an adapter once until the end of the program, this shouldn't matter.

use Log::Any::Adapter (...)

If you pass arguments to "use Log::Any::Adapter", it calls "Log::Any::Adapter->set" with those arguments.

Log::Any::Adapter->remove (entry)

Remove an entry previously returned by "set".

USING MORE THAN ONE ADAPTER

"Log::Any" maintains a stack of entries created via "set". If you call "set" repeatedly, you will leak memory unless you do one of the following:

When getting a logger for a particular category, "Log::Any" will work its way down the stack and use the first matching entry.

Whenever the stack changes, any "Log::Any" loggers that have previously been created will automatically adjust to the new stack. For example:

```
my $log = Log::Any->get_logger();
$log->error("aiggh!");   # this goes nowhere
...
{
    Log::Any::Adapter->set({ lexically => \my $lex }, 'Log4perl');
    $log->error("aiggh!");   # this goes to log4perl
    ...
}
$log->error("aiggh!");   # this goes nowhere again
```

BUILDING ON THE Log::Any BACKEND

get

```
my $adapter= Log::Any::Adapter->get($category);
```

The primary intended way to extend the producing-side of Log::Any is with a custom Log::Any::Proxy class. However, for special logging scenarios you might also just want access to the adapter for a given category. The API of an adapter object is described in Log::Any::Adapter::Development. Beware that adapter objects can be "rewritten" on the fly, so any conditional behavior you write depending on the capabilities of an adapter must be re-checked every time you access the adapter.

SEE ALSO

Log::Any

AUTHORS

- Jonathan Swartz <swartz@pobox.com>
- David Golden <dagolden@cpan.org>
- Doug Bell <preaction@cpan.org>
- Daniel Pittman <daniel@rimspace.net>
- Stephen Thirlwall <sdt@cpan.org>

COPYRIGHT AND LICENSE