NAME

"IO::Async::Protocol::LineStream" − stream−based protocols using lines of text

SYNOPSIS

Most likely this class will be subclassed to implement a particular network protocol.

```
package Net::Async::HelloWorld;

use strict;
use warnings;
use base qw( IO::Async::Protocol::LineStream );

sub on_read_line
{
   my $self = shift;
   my ( $line ) = @_;

   if( $line =~ m/^HELLO (.*)/ ) {
      my $name = $1;

      $self->invoke_event( on_hello => $name );
   }
}

sub send_hello
{
   my $self = shift;
   my ( $name ) = @_;

   $self->write_line( "HELLO $name" );
}
```

This small example elides such details as error handling, which a real protocol implementation would be likely to contain.

DESCRIPTION

EVENTS

The following events are invoked, either using subclass methods or CODE references in parameters:

on_read_line `$line`

Invoked when a new complete line of input is received.

PARAMETERS

The following named parameters may be passed to `new` or `configure`:

on_read_line => CODE

CODE reference for the `on_read_line` event.

METHODS

write_line

```
$lineprotocol->write_line( $text )
```

Writes a line of text to the transport stream. The text will have the end-of-line marker appended to it; `$text` should not end with it.

AUTHOR

Paul Evans <leonerd@leonerd.org.uk>