NAME

"IO::Async::Protocol::Stream" – base class for stream–based protocols

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::Stream );

 sub on_read
 {
    my $self = shift;
    my ( $buffref, $eof ) = @_;

    return 0 unless $$buffref =~ s/^(.*)\n//;
    my $line = $1;

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

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

    return 1;
 }

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

    $self->write( "HELLO $name\n" );
 }
```

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

DESCRIPTION

This subclass of IO::Async::Protocol is intended to stand as a base class for implementing stream-based protocols. It provides an interface similar to IO::Async::Stream, primarily, a `write` method and an `on_read` event handler.

It contains an instance of an IO::Async::Stream object which it uses for actual communication, rather than being a subclass of it, allowing a level of independence from the actual stream being used. For example, the stream may actually be an IO::Async::SSLStream to allow the protocol to be used over SSL.

As with IO::Async::Stream, it is required that by the time the protocol object is added to a Loop, that it either has an `on_read` method, or has been configured with an `on_read` callback handler.

EVENTS

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

`$ret` **= on_read \$buffer,** `$eof`

on_read_eof

on_write_eof

The event handlers are invoked identically to IO::Async::Stream.

on_closed

The `on_closed` handler is optional, but if provided, will be invoked after the stream is closed by either side (either because the `close()` method has been invoked on it, or on an incoming EOF).

PARAMETERS

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

on_read => CODE
on_read_eof => CODE
on_write_eof => CODE

CODE references for the events.

handle => IO

A shortcut for the common case where the transport only needs to be a plain IO::Async::Stream object. If this argument is provided without a `transport` object, a new IO::Async::Stream object will be built around the given IO handle, and used as the transport.

METHODS

write

```
    $protocol->write( $data )
```

Writes the given data by calling the `write` method on the contained transport stream.

connect

```
    $protocol->connect( %args )
```

Sets up a connection to a peer, and configures the underlying `transport` for the Protocol. Calls IO::Async::Protocol `connect` with `socktype` set to `"stream"`.

AUTHOR

Paul Evans <leonerd@leonerd.org.uk>