NAME

"IO::Async::Protocol" − base class for transport−based protocols

DESCRIPTION

This subclass of IO::Async:Notifier provides storage for a IO::Async::Handle object, to act as a transport for some protocol. It contains an instance of the transport object, which it adds as a child notifier, allowing a level of independence from the actual transport being used. For example, a stream may actually be an IO::Async::SSLStream to allow the protocol to be used over SSL.

This class is not intended to be used directly, instead, see one of the subclasses

IO::Async::Protocol::Stream − base class for stream-based protocols

EVENTS

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

on_closed

Optional. Invoked when the transport handle becomes closed.

PARAMETERS

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

transport => IO::Async::Handle

The IO::Async::Handle to delegate communications to.

on_closed => CODE

CODE reference for the `on_closed` event.

When a new `transport` object is given, it will be configured by calling the `setup_transport` method, then added as a child notifier. If a different transport object was already configured, this will first be removed and deconfigured using the `teardown_transport`.

METHODS

transport

```
$transport = $protocol->transport
```

Returns the stored transport object

connect

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

Sets up a connection to a peer, and configures the underlying `transport` for the Protocol.

Takes the following named arguments:

socktype => STRING or INT

Required. Identifies the socket type, and the type of continuation that will be used. If this value is `"stream"` or `SOCK_STREAM` then `on_stream` continuation will be used; otherwise `on_socket` will be used.

on_connected => CODE

Optional. If supplied, will be invoked once the connection has been established.

```
$on_connected->( $protocol )
```

transport => IO::Async::Handle

Optional. If this is provided, it will immediately be configured as the transport (by calling `configure`), and the `on_connected` callback will be invoked. This is provided as a convenient shortcut.

Other arguments will be passed to the underlying IO::Async::Loop `connect` call.

TRANSPORT DELEGATION

The following methods are delegated to the transport object

```
close
```

SUBCLASS METHODS

`IO::Async::Protocol` is a base class provided so that specific subclasses of it provide more specific behaviour. The base class provides a number of methods that subclasses may wish to override.

If a subclass implements any of these, be sure to invoke the superclass method at some point within the code.

setup_transport

```
$protocol->setup_transport( $transport )
```

Called by `configure` when a new `transport` object is given, this method should perform whatever setup is required to wire the new transport object into the protocol object; typically by setting up event handlers.

teardown_transport

```
$protocol->teardown_transport( $transport )
```

The reverse of `setup_transport`; called by `configure` when a previously set-up transport object is about to be replaced.

AUTHOR

Paul Evans <leonerd@leonerd.org.uk>