NAME

Net::DBus::Binding::Server − A server to accept incoming connections

SYNOPSIS

Creating a new server and accepting client connections

```
use Net::DBus::Binding::Server;

my $server = Net::DBus::Binding::Server->new(address => "unix:path=/path/to/soc

$server->connection_callback(\&new_connection);

sub new_connection {
    my $connection = shift;

    .. work with new connection...
}
```

Managing the server and new connections in an event loop

```
my $reactor = Net::DBus::Binding::Reactor->new();

$reactor->manage($server);
$reactor->run();

sub new_connection {
    my $connection = shift;

    $reactor->manage($connection);
}
```

DESCRIPTION

A server for receiving connection from client programs. The methods defined on this module have a close correspondence to the dbus_server_XXX methods in the C API, so for further details on their behaviour, the C API documentation may be of use.

METHODS

my `$server` = Net::DBus::Binding::Server−>new(address => ‘‘unix:path=/path/to/socket’’);

Creates a new server binding it to the socket specified by the `address` parameter.

`$status` = `$server`−>**is_connected**();

Returns zero if the server has been disconnected, otherwise a positive value is returned.

`$server`−>**disconnect**()

Closes this server to the remote host. This method is called automatically during garbage collection (ie in the DESTROY method) if the programmer forgets to explicitly disconnect.

`$server`−>set_watch_callbacks(\&add_watch, \&remove_watch, \&toggle_watch);

Register a set of callbacks for adding, removing & updating watches in the application’s event loop. Each parameter should be a code reference, which on each invocation, will be supplied with two parameters, the server object and the watch object. If you are using a `Net::DBus::Binding::Reactor` object as the application event loop, then the ’manage’ method on that object will call this on your behalf.

`$server`−>set_timeout_callbacks(\&add_timeout, \&remove_timeout, \&toggle_timeout);

Register a set of callbacks for adding, removing & updating timeouts in the application’s event loop. Each parameter should be a code reference, which on each invocation, will be supplied with two parameters, the server object and the timeout object. If you are using a `Net::DBus::Binding::Reactor` object as the application event loop, then the ’manage’ method on that object will call this on your behalf.

`$server->`set_connection_callback(\&handler)

Registers the handler to use for dealing with new incoming connections from clients. The code reference will be invoked each time a new client connects and supplied with a single parameter which is the `Net::DBus::Binding::Connection` object representing the client.

AUTHOR

Daniel P. Berrange

COPYRIGHT

SEE ALSO

Net::DBus::Binding::Connection, Net::DBus::Binding::Bus, Net::DBus::Binding::Message::Signal, Net::DBus::Binding::Message::MethodCall, Net::DBus::Binding::Message::MethodReturn, Net::DBus::Binding::Message::Error