



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

Rocky Enterprise Linux 9.2 Manual Pages on command 'Net::DBus::Test::MockObject.3pm'

```
$ man Net::DBus::Test::MockObject.3pm
```

```
Net::DBus::Test::MockObject(3pmUser Contributed Perl DocumentatioNet::DBus::Test::MockObject(3pm)
```

NAME

Net::DBus::Test::MockObject - Fake an object from the bus for unit testing

SYNOPSIS

```
use Net::DBus;

use Net::DBus::Test::MockObject;

my $bus = Net::DBus->test

# Lets fake presence of HAL...

# First we need to define the service
my $service = $bus->export_service("org.freedesktop.Hal");

# Then create a mock object
my $object = Net::DBus::Test::MockObject->new($service,
                                             "/org/freedesktop/Hal/Manager");

# Fake the 'GetAllDevices' method
$object->seed_action("org.freedesktop.Hal.Manager",
                   "GetAllDevices",
```

```
reply => {  
    return => [ "/org/freedesktop/Hal/devices/computer_i8042_Aux_Port",  
              "/org/freedesktop/Hal/devices/computer_i8042_Aux_Port_logicaldev_input",  
              "/org/freedesktop/Hal/devices/computer_i8042_Kbd_Port",  
              "/org/freedesktop/Hal/devices/computer_i8042_Kbd_Port_logicaldev_input"  
            ],  
    });
```

Now can test any class which calls out to 'GetAllDevices' in HAL

....test stuff....

DESCRIPTION

This provides an alternate for `Net::DBus::Object` to enable bus objects to be quickly mocked up, thus facilitating creation of unit tests for services which may need to call out to objects provided by 3rd party services on the bus. It is typically used as a companion to the `Net::DBus::MockBus` object, to enable complex services to be tested without actually starting a real bus.

!!!! WARNING !!!

This object & its APIs should be considered very experimental at this point in time, and no guarantees about future API compatibility are provided what-so-ever. Comments & suggestions on how to evolve this framework are, however, welcome & encouraged.

METHODS

```
my $object = Net::DBus::Test::MockObject->new($service, $path, $interface);
```

Create a new mock object, attaching to the service defined by the `$service` parameter.

This would be an instance of the `Net::DBus::Service` object. The `$path` parameter defines the object path at which to attach this mock object, and `$interface` defines the interface it will support.

```
my $service = $object->get_service
```

Retrieves the `Net::DBus::Service` object within which this object is exported.

```
my $path = $object->get_object_path
```

Retrieves the path under which this object is exported

```
my $msg = $object->get_last_message
```

Retrieves the last message processed by this object. The returned object is an instance of `Net::DBus::Binding::Message`

```
my $sig = $object->get_last_message_signature
```

Retrieves the type signature of the last processed message.

```
my $value = $object->get_last_message_param
```

Returns the first value supplied as an argument to the last processed message.

```
my @values = $object->get_last_message_param_list
```

Returns a list of all the values supplied as arguments to the last processed message.

```
$object->seed_action($interface, $method, %action);
```

Registers an action to be performed when a message corresponding to the method `$method` within the interface `$interface` is received. The `%action` parameter can have a number of possible keys set:

signals

Causes a signal to be emitted when the method is invoked. The value associated with this key should be an instance of the `Net::DBus::Binding::Message::Signal` class.

error

Causes an error to be generated when the method is invoked. The value associated with this key should be a hash reference, with two elements. The first, "name", giving the error name, and the second, "description", providing the descriptive text.

reply

Causes a normal method return to be generated. The value associated with this key should be an array reference, whose elements are the values to be returned by the method.

BUGS

It doesn't completely replicate the API of `Net::DBus::Binding::Object`, merely enough to make the high level bindings work in a test scenario.

AUTHOR

Daniel P. Berrange

COPYRIGHT

Copyright (C) 2004-2009 Daniel P. Berrange

SEE ALSO

`Net::DBus`, `Net::DBus::Object`, `Net::DBus::Test::MockConnection`,
<<http://www.mockobjects.com/Faq.html>>

perl v5.34.0

2022-02-06

`Net::DBus::Test::MockObject(3pm)`