NAME

Net::DBus::RemoteObject − Access objects provided on the bus

SYNOPSIS

```
my $service = $bus->get_service("org.freedesktop.DBus");
my $object = $service->get_object("/org/freedesktop/DBus");

print "Names on the bus {\n";
foreach my $name (sort @{$object->ListNames}) {
    print "  ", $name, "\n";
}
print "}\n";
```

DESCRIPTION

This module provides the API for accessing remote objects available on the bus. It uses the autoloader to fake the presence of methods based on the API of the remote object. There is also support for setting callbacks against signals, and accessing properties of the object.

METHODS

my `$object` = Net::DBus::RemoteObject->new($service, `$object_path`[, `$interface`], \%params);

Creates a new handle to a remote object. The `$service` parameter is an instance of the Net::DBus::RemoteService method, and `$object_path` is the identifier of an object exported by this service, for example `/org/freedesktop/DBus`. For remote objects which implement more than one interface it is possible to specify an optional name of an interface as the third parameter. This is only really required, however, if two interfaces in the object provide methods with the same name, since introspection data can be used to automatically resolve the correct interface to call cases where method names are unique. Rather than using this constructor directly, it is preferable to use the `get_object` method on Net::DBus::RemoteService, since this caches handles to remote objects, eliminating unnecessary introspection data lookups.

The `%params` parameter contains extra configuration parameters for the object. Currently a single parameter is supported, `timeout` which takes a value in milliseconds to use as the timeout for method calls on the object.

my `$object` = `$object`->as_interface($interface);

Casts the object to a specific interface, returning a new instance of the Net::DBus::RemoteObject specialized to the desired interface. It is only necessary to cast objects to a specific interface, if two interfaces export methods or signals with the same name, or the remote object does not support introspection.

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

Retrieves a handle for the remote service on which this object is attached. The returned object is an instance of Net::DBus::RemoteService

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

Retrieves the unique path identifier for this object within the service.

my `$object` = `$object`->get_child_object($subpath, [$interface])

Retrieves a handle to a child of this object, identified by the relative path `$subpath`. The returned object is an instance of `Net::DBus::RemoteObject`. The optional `$interface` parameter can be used to immediately cast the object to a specific type.

my `$sigid` = `$object`->connect_to_signal($name, `$coderef`);

Connects a callback to a signal emitted by the object. The `$name` parameter is the name of the signal within the object, and `$coderef` is a reference to an anonymous subroutine. When the signal `$name` is emitted by the remote object, the subroutine `$coderef` will be invoked, and passed the parameters from the signal. A unique `$sigid` will be returned, which can be later passed to `disconnect_from_signal` to remove the handler

`$object`->disconnect_from_signal($name, `$sigid`);

Disconnects from a signal emitted by the object. The `$name` parameter is the name of the signal within the object. The `$sigid` must be the unique signal handler ID returned by a previous `connect_to_signal` method call.

AUTHOR

Daniel Berrange <dan@berrange.com>

COPYRIGHT

SEE ALSO

Net::DBus::RemoteService, Net::DBus::Object, Net::DBus::Annotation