NAME

Net::DBus::RemoteService − Access services provided on the bus

SYNOPSIS

```
my $bus = Net::DBus->find;
my $service = $bus->get_service("org.freedesktop.DBus");

my $object = $service->get_object("/org/freedesktop/DBus");
foreach (@{$object->ListNames}) {
  print "$_\n";
}
```

DESCRIPTION

This object provides a handle to a remote service on the bus. From this handle it is possible to access objects associated with the service. If a service is not running, an attempt will be made to activate it the first time a method is called against one of its objects.

METHODS

my `$service` = Net::DBus::RemoteService−>new($bus, `$owner`, `$service_name`);

Creates a new handle for a remote service. The `$bus` parameter is an instance of Net::DBus, `$owner` is the name of the client providing the service, while `$service_name` is the well known name of the service on the bus. Service names consist of two or more tokens, separated by periods, while the tokens comprise the letters a−z, A−Z, 0−9 and _, for example `org.freedesktop.DBus`. There is generally no need to call this constructor, instead the `get_service` method on Net::DBus should be used. This caches handles to remote services, eliminating repeated retrieval of introspection data.

my `$bus` = `$service`−>get_bus;

Retrieves a handle for the bus to which this service is attached. The returned object will be an instance of Net::DBus.

my `$service_name` = `$service`−>get_service_name

Retrieves the name of the remote service as known to the bus.

my `$owner_name` = `$service`−>get_owner_name;

Retrieves the name of the client owning the service at the time it was connected to.

my `$object` = `$service`−>get_object($object_path[, `$interface`]); =item my `$object` = `$service`−>get_object($object_path, \%params);

Retrieves a handle to the remote object provided by the service with the name of `$object_path`. If the optional `$interface` parameter is provided, the object will immediately be cast to the designated interface. NB, it is only necessary to cast an object to a specific interface if there are multiple interfaces on the object providing methods with the same name, or the remote object does support introspection. The returned object will be an instance of Net::DBus::RemoteObject.

An alternate form of the method is available, passing a hash reference of extra parameters. Valid keys in the hash are `interface` specifying the interface name to cast to, and `timeout` specifying a timeout in milliseconds

my `$timeout` = `$service`−>timeout(60 * 1000);

Sets or retrieves the timeout value which will be used for DBus requests belongs to this service. The value is in milliseconds. If the timeout for a service is undefined, then the default timeout from the bus will apply.

AUTHOR

Daniel Berrange <dan@berrange.com>

COPYRIGHT

SEE ALSO

Net::DBus::RemoteObject, Net::DBus::Service, Net::DBus