

**Network Centric Computing**

**Formal Element: gRPC Calculator**

by

Jack Harding

This Report is submitted in partial fulfilment of the requirements of the Honours Degree in Electrical and Electronic Engineering (DT021A) of the Dublin Institute of Technology

18 April 2019

Lecturer: Dr Ray Lynch

**Contents**

[Introduction 2](#_Toc5006270)

[Objective 2](#_Toc5006271)

[Code 2](#_Toc5006272)

# Introduction

1. A background discussion on gRPC. Include a comparison with traditional RPC.

Google developed it for object serialisation and is a mechanism in gRPC for generating language specific code. Similar to RPC, the objects are defined in a definition file where the input and output parameters are defined, interfaces are replaced by messages in gRPC and are used in .proto files. Inside an RPC interface is a function which specifies input, output, and the parameters each take; the same is done with a proto file, a message is declared which describes the contents of the message e.g. a car message might have a string value for the model and a integer value for the price. The request message describes the request serialisation, which basically means when the client sends a request this is the format they wish to parse the data with.

message NewRequest {

int32 price = 1;

string model = 2;

}

The integers values above do not indicate the value of the variable rather the size in bytes they occupy when serialised, each variable in a message must have a unique number attached to it, the range is 1-15 but can be increased. The similar approach is used on the response message but the number/type of fields returned may differ, just like a normal function.

Creating a service involves the use of both response and request messages:

service NewService {

rpc MyFunc(NewRequest) returns (NewResponse);

rpc AnotherFunc(AnotherRequest) returns (AnotherResponse);

}

MyFunc is the function which is passed NewRequest this gRPC service returns the response. The code above is written in the .proto file which is then compiled into client and server stubs.

Similar to JSON and XML but is more efficient due to its binary encoding, making it far faster, when XML and JSON are compared to proto, the compressed data size is a third of XML and half that of JSON.

# Objective

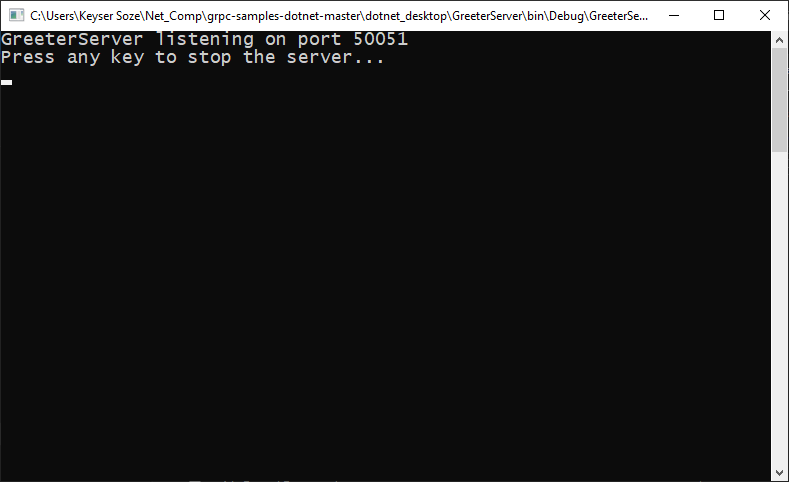
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Code | Description | Project | File | Line |
| CS0012 | The type 'Nullable<>' is defined in an assembly that is not referenced. You must add a reference to assembly 'netstandard, Version=2.0.0.0, Culture=neutral, PublicKeyToken=cc7b13ffcd2ddd51'. | SquaredClient | gRPC\SquaredClient\ClientProgram.cs | 30 |

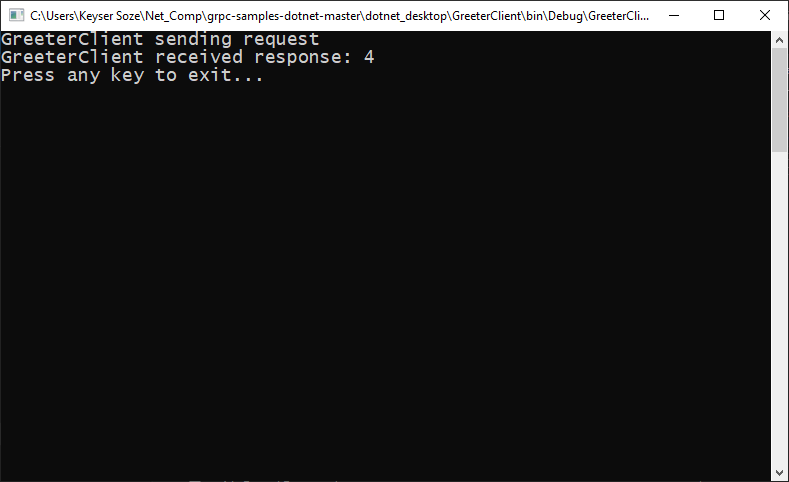
System.IO.FileNotFoundException

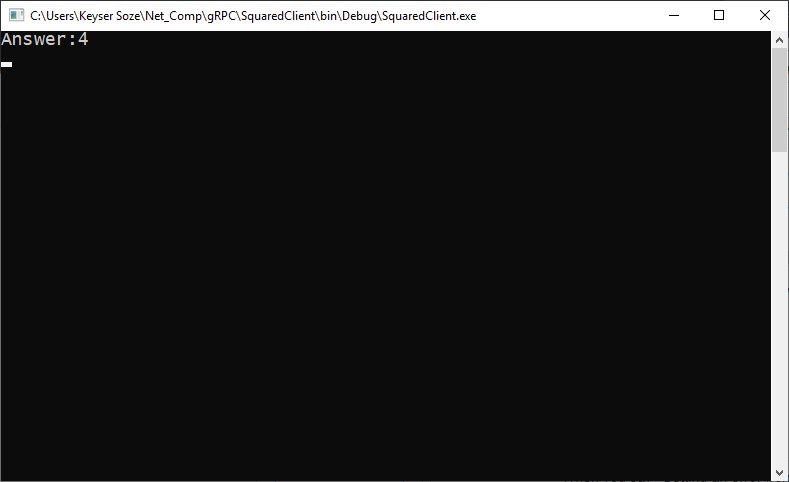
HResult=0x80070002

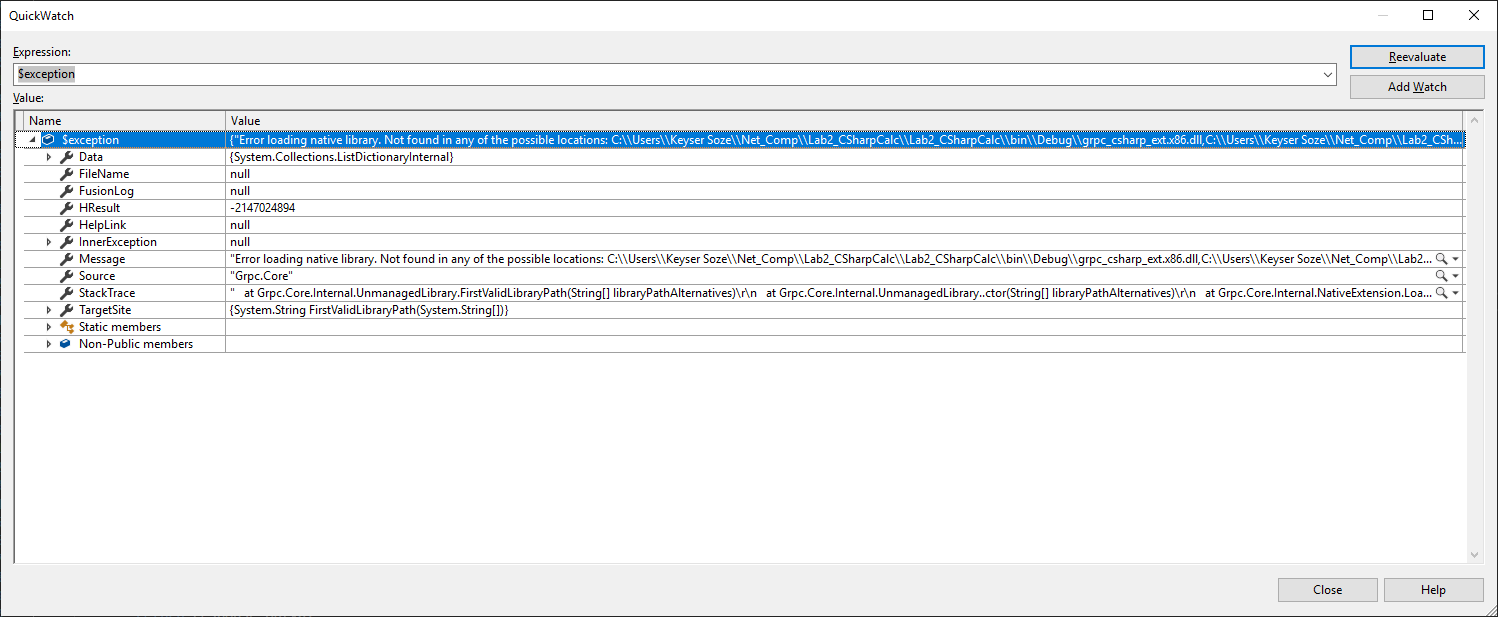
Message=Error loading native library. Not found in any of the possible locations: C:\Users\Keyser

# Code









1. Add a button to your C# calculator
2. Calls a gRPC remote function and displays the return value on the calculator screen.
3. It is up to you to decide what the remote function will do.

* A detailed description of your particular implementation of the code above.
* You should include an explanation of the particular code snippets related to gRPC.
* Also include flowcharts etc. that will demonstrate your understanding of the technology.