

Accessing Natural from C#

Contents

| | |
|---|----|
| 1. Introduction | 2 |
| 2. Prerequisites | 2 |
| 3. C# | 2 |
| 4. Natural | 2 |
| 5. Discovery..... | 2 |
| 6. Accessing Web Service with C#..... | 7 |
| 6.1. Initial Setup | 7 |
| 6.2. Designing the Form | 10 |
| 6.3. Writing the Code..... | 11 |
| 6.3.1. Global and Constructor Code..... | 11 |
| 6.3.2. Event Handling Code: Search Button | 11 |
| 6.4. Building the Code..... | 11 |
| 6.5. Running the code | 11 |
| 7. Conclusion..... | 12 |

1. Introduction

In this tutorial we will show you how to build a C# application to access Natural via the SOA Gateway.

2. Prerequisites

As a prerequisite you must install Natural and C#.

It is also assumed you already have a SOA Gateway server and Control Centre installed. See [here](#) for more info about installing the SOA Gateway.

3. C#

To build and run C# applications, you will need a Visual Studio IDE. If you do not already have a C# IDE installed, we recommend using the Microsoft Visual Studio Express range of products. They can be downloaded freely from Microsoft website, packaged for a number of languages, including C#. See [here](#) for more information about downloading, installing, and configuring VC# Express.

4. Natural

Natural is a programming language designed to simplify the implementation of business solutions. It takes a very pragmatic and non-theoretical approach to common programming tasks such as database access.

Natural is similar to JAVA in that it is a semi-interpreted language; Natural source programs are “compiled” into a platform independent byte code (object) format which is then executed (interpreted) by a platform dependent runtime environment. While originally developed for mainframe computers, Natural has been available on Windows and UNIX platforms for several years now.

This tutorial is based on the Natural Community Edition which can be downloaded from [Natural download site](#).

When Natural is installed successfully load the objects used for this tutorial. Details on this can be found [here](#). The sample objects are suitable for a Windows/Unix/Linux environment.

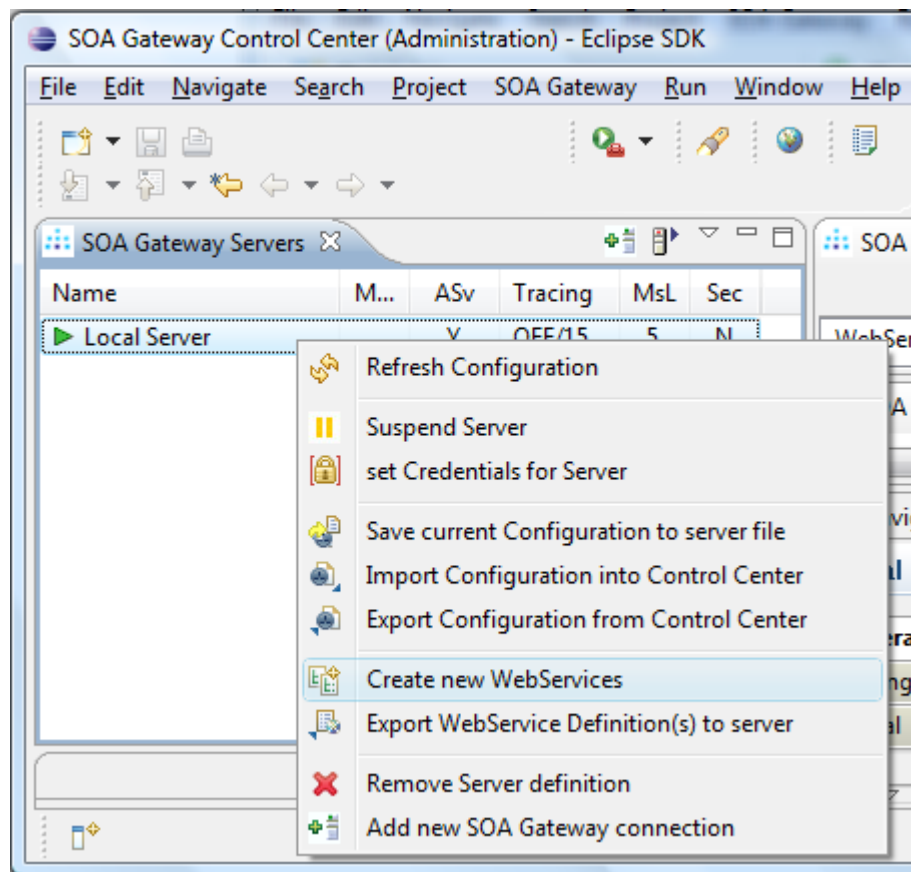
5. Discovery

In this section we'll show you how to create a web services for one of the programs. This web service can be used by InfoPath (and many others) to give you direct real-time access to your Natural program.

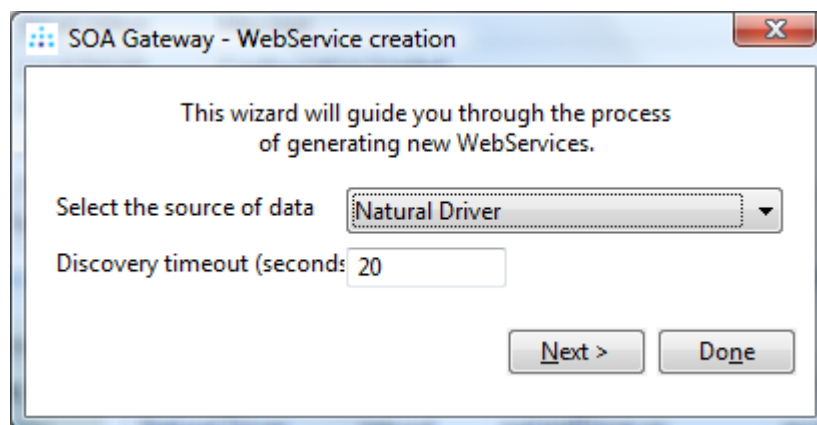
Web Service Creation using SOA Gateway

Start your SOA Gateway Control Centre. See [here](#) for an introduction to the Control Centre.

In your servers view, right click the entry which represents your local SOA Gateway Server. Select “Create new WebServices”.

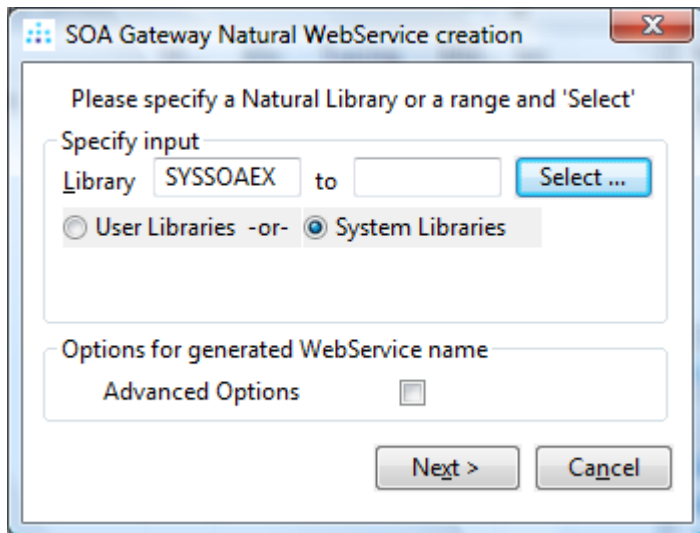


From the next dialog, choose “Natural Driver”. If you do not see have a Natural Driver in the list, see how to create one [here](#).

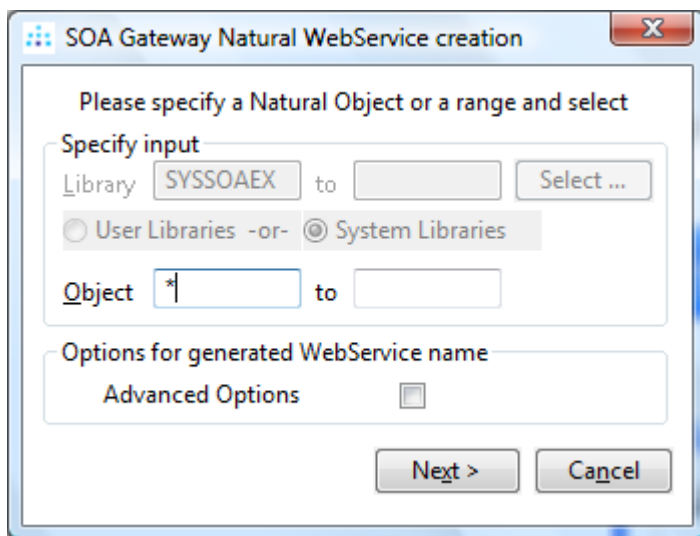


Click Next.

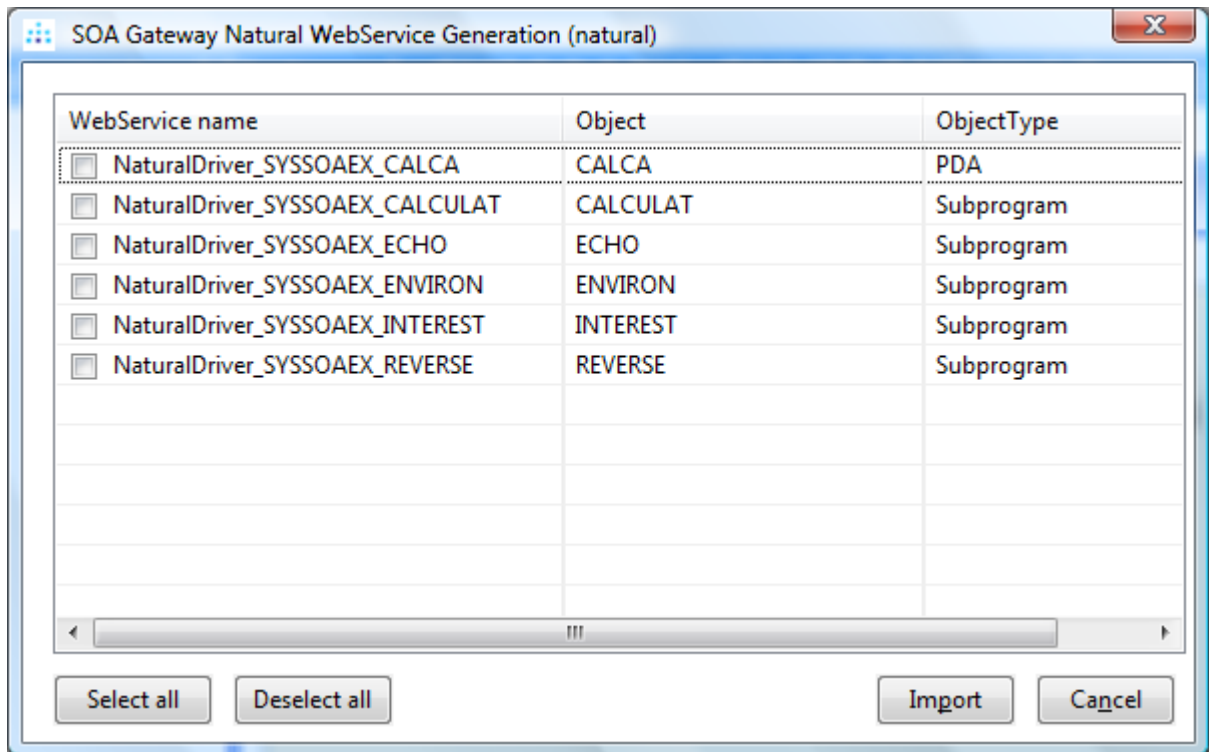
Enter “SYSSOAEX” for the Library name, select “System Libraries”, click “Next >”



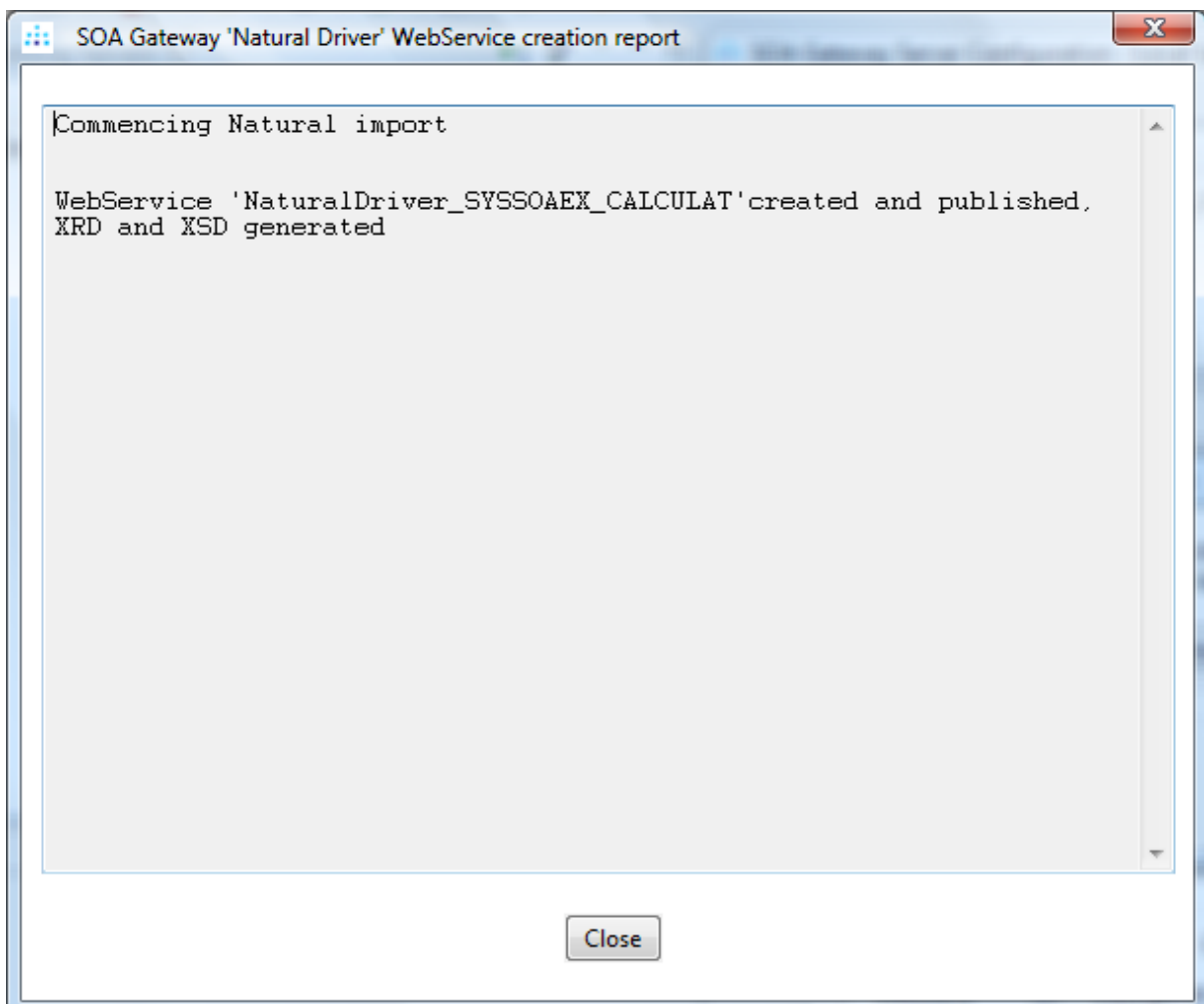
Put an asterisk (*) in the "Object" field and click "Next >"



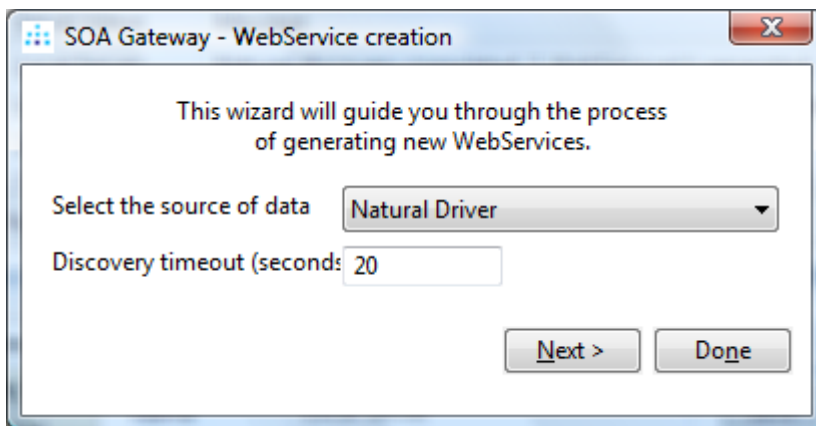
A list of possible WebServices is presented:



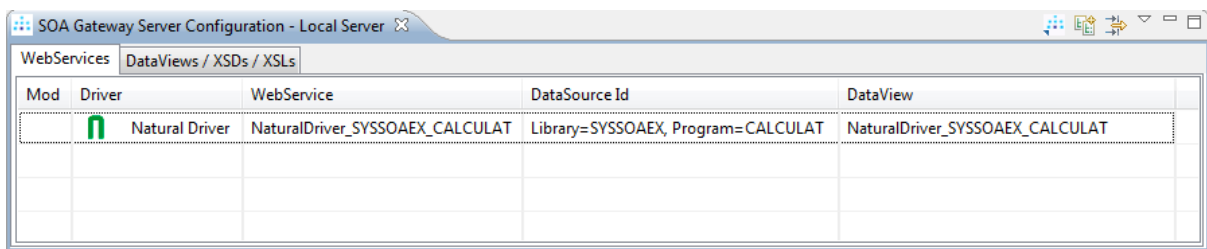
Select the "Natural_SYSSOAEX_CALCULAT" WebService and click "Import".



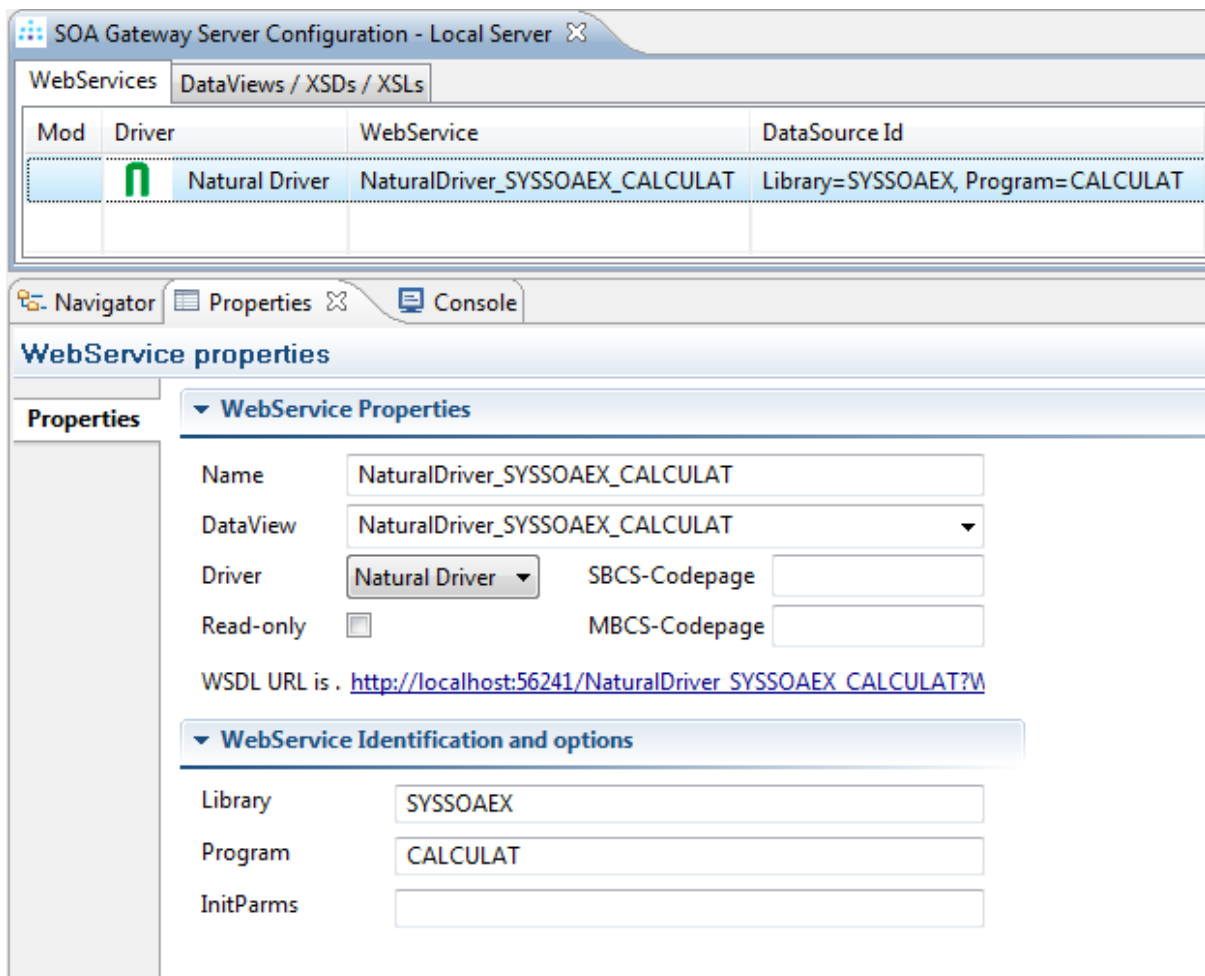
Select Close.



Select Done.



Click the WebService in the Configuration View. The Properties View displays the details.



Web Service Description Language (WSDL) is a standard, XML-based language that is used to describe a Web Service. The WSDL itself is usually interpreted by a web service client, such as InfoPath.

As WSDL is XML-based, it will open in your browser of choice. To see the WSDL click on the link at 'WSDL URL is' in the WebService Properties window.

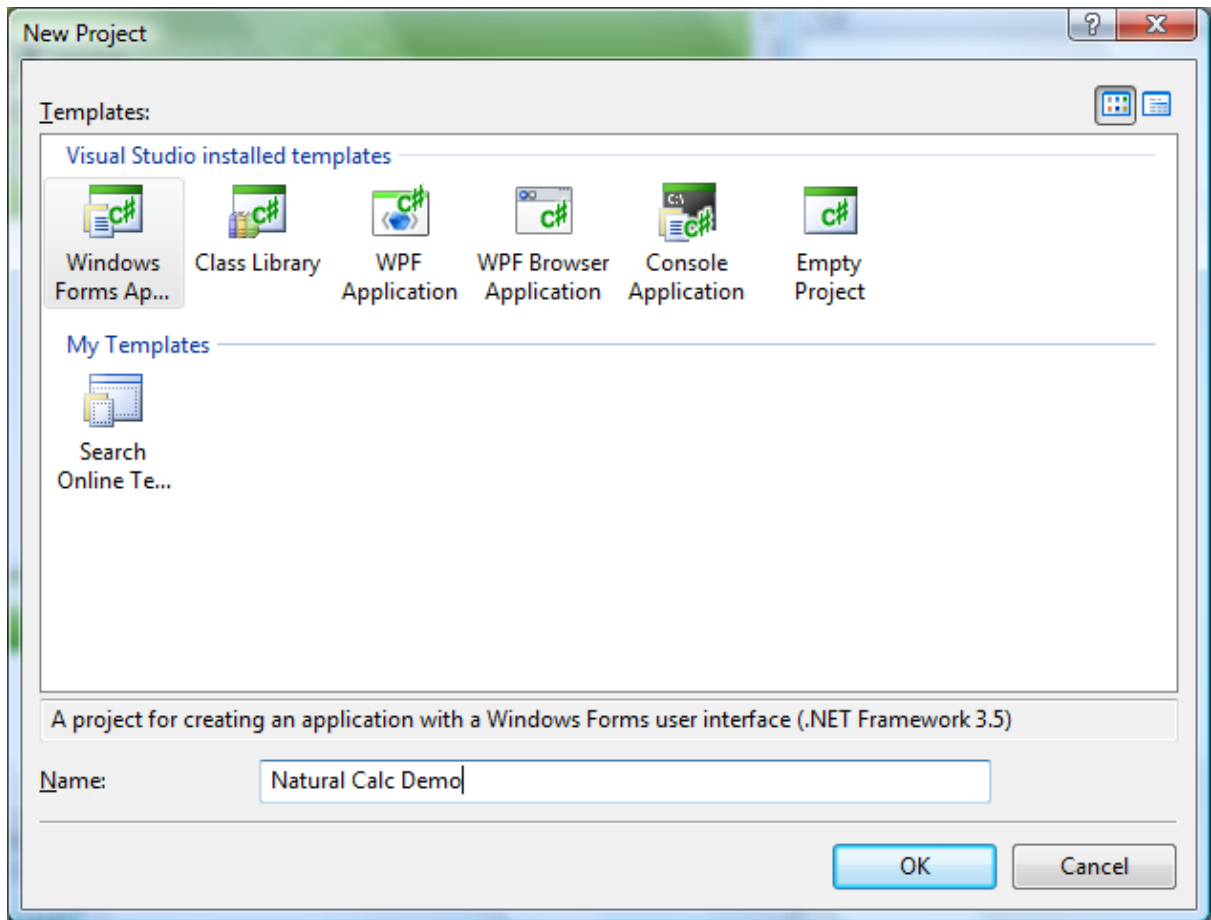
This WSDL is the starting point for using Web Services, and can be used time and again by different web service clients.

6. Accessing Web Service with C#

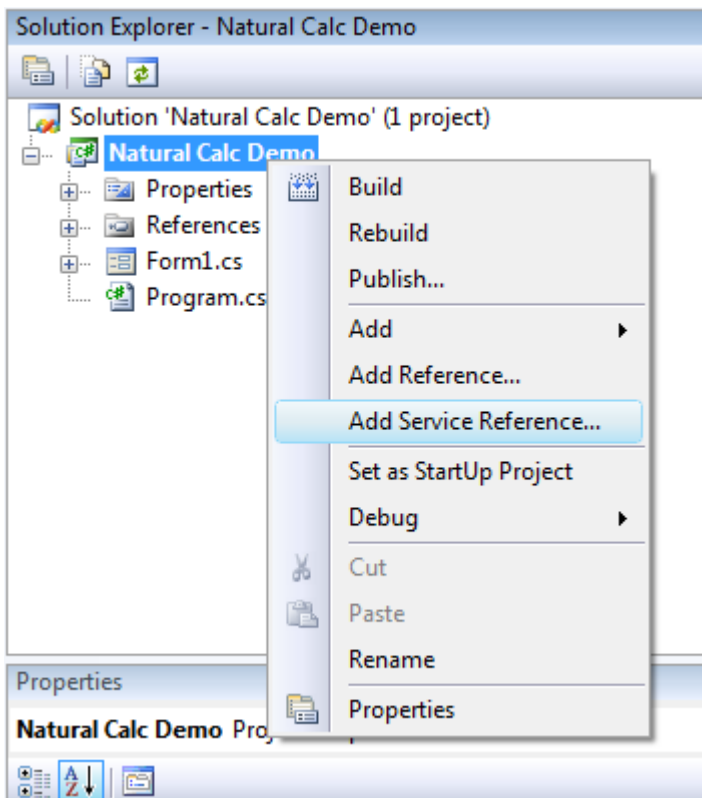
C# is an object orientated programming language from Microsoft as part of the .NET initiative. Its syntax is loosely based on C++, while borrowing other aspects from programming languages such as Java.

6.1. Initial Setup

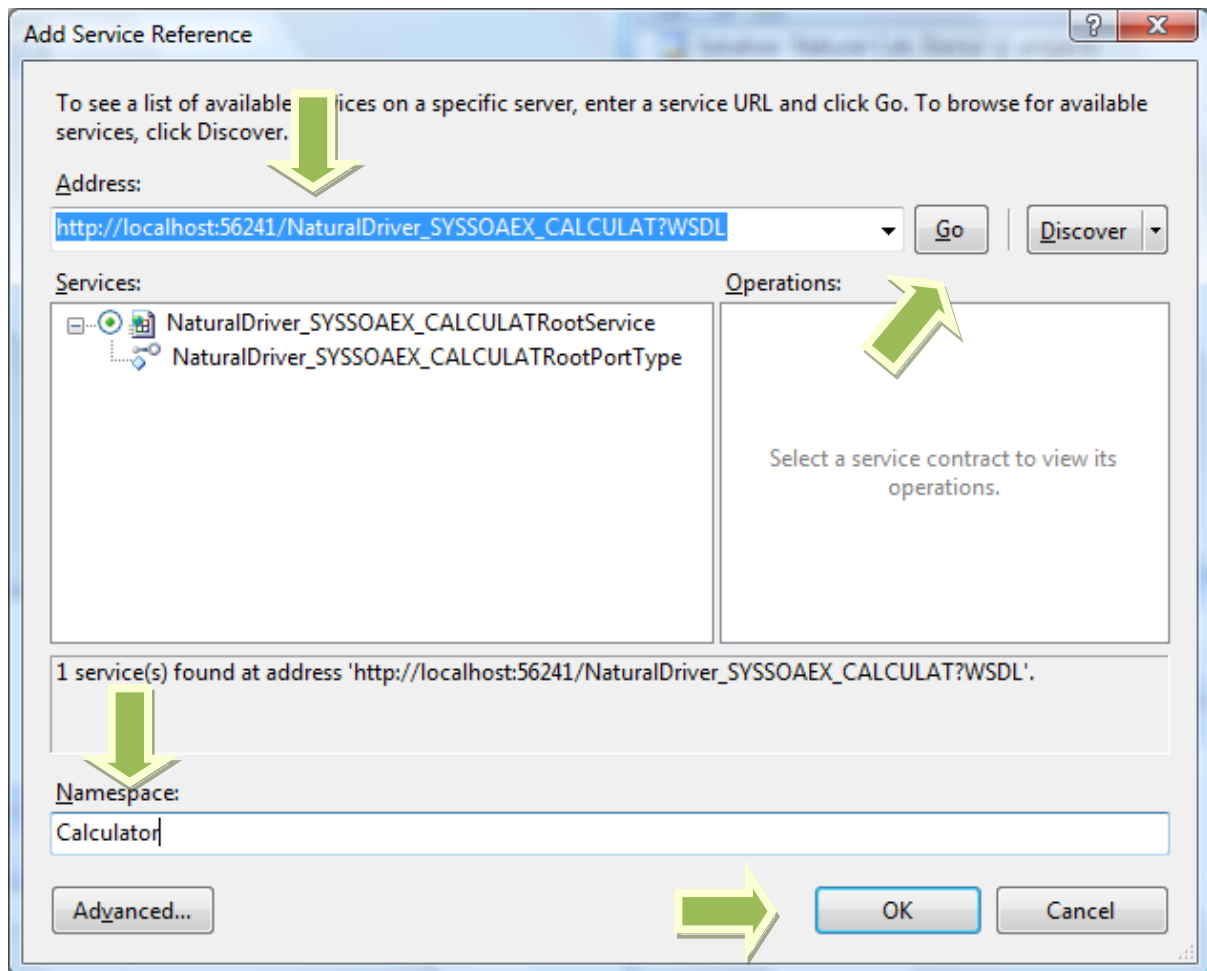
Start *Microsoft Visual C# Express* and create a New Windows Application Project named **Natural Calc Demo**. This demo uses Visual C# 2008.



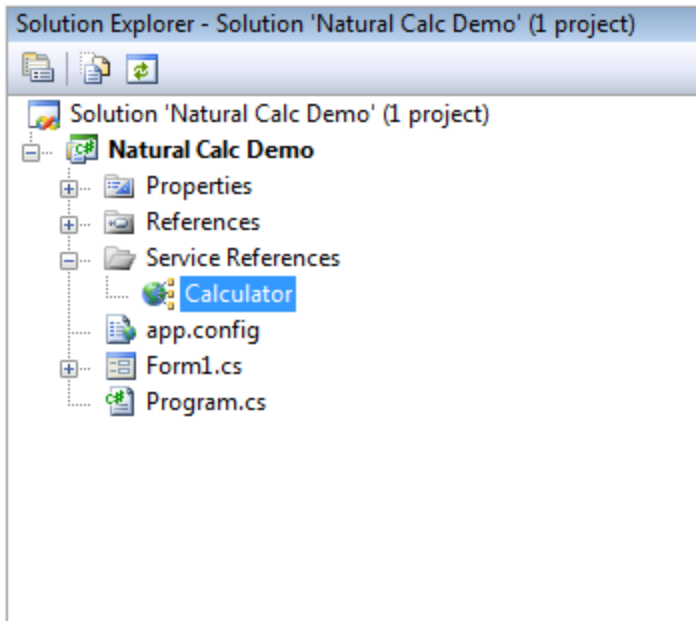
Right-click your project and select "Add Service Reference"



- Copy the URL of the calculator web services WSDL (in this example http://localhost:56241/NaturalDriver_SYSSOAEX_CALCULAT?WSDL) into the Address box. Change the server and port as appropriate.
- Click Go.
- Once the WSDL has been loaded, change the Namespace to **Calculator**
- Click **OK**

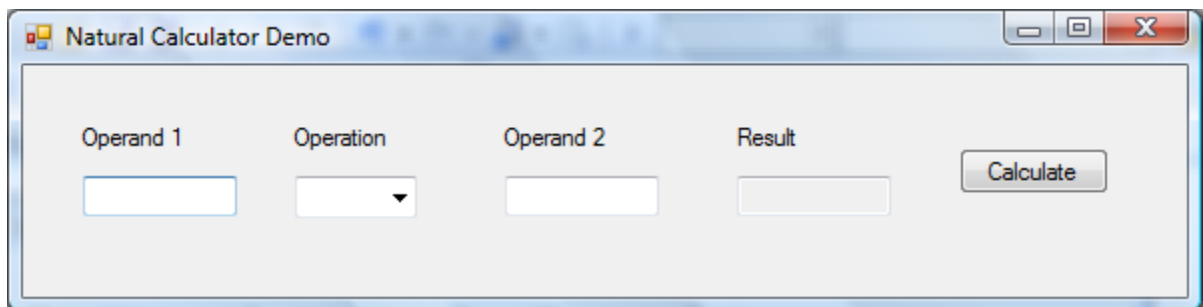


You should now that a reference to the Calculator Webservice.



6.2. Designing the Form

In this section we'll add the necessary controls to our Form. Before you start ensure, that you are in the Designer View (View -> Designer), and that you have the control Toolbox available (View->Toolbox)



I've used the following controls in this Form

- Label1 (Text property set to Operand 1)
- Label2 (Text property set to Operation)
- Label3 (Text property set to Operand 2)
- Label4 (Text property set to Result)
- TextBox1 (placed under Operand 1)
- TextBox2 (placed under Operand 2)
- TextBox3 (placed under Result)
- Button (Text property set to Calculate)
- ComboBox (placed under Operation). Click on ComboBox control and hit F4 to go the Properties tab. Find the Items property and click on the Collection button. Add the following operations: ADD, SUB, DIV, MUL and MOD.

6.3. Writing the Code

6.3.1. Global and Constructor Code

Now that the Form controls have been added, we need to write the code to call our Web Services when the buttons are clicked.

Switch to your Code view, by clicking View->Code

After the “using” statements, enter the following

```
using Natural_Calc_Demo.Calculator;
```

6.3.2. Event Handling Code: Search Button

Switch back to your Designer view, and double-click the “Calculate” button in your Form. Your IDE will switch over to the code view, and a new member function to handle the button click will be created.

When this button is clicked, we want to take the contents of textBox1, textBox2 and the operation entry from the ComboBox and send this to our Calculator web service. The web service should return the correct result.

The complete code is available [here](#). Replace ALL of the code in Form1.cs file with the downloaded code.

6.4. Building the Code

You can compile the code by hitting F6, or Build -> Build Solution.

Hopefully, all is well, but in the case of errors try the following:

- Obviously misspellings are often the cause of compilation errors. Ensure that all object names, and variable names are spelt correctly.
- The C# has a neat trick where you can type the first few letters of an object, and by hitting Control + Space, it will bring up the suggested object names.
- Similarly, if you need the name of a member of an object, type the object name, followed by a dot (“.”) and Control+Space. The list of available proposals should appear.

6.5. Running the code

By hitting F5 or Debug -> Start Debugging, you can run your code.

If you hit problems, you may wish to debug your code by adding breakpoints in your code. See the IDE documentation for further information.

7. Conclusion

This tutorial shows how to access Natural from C# using the SOA Gateway. As you can see, you have built a powerful application that uses Web Services to retrieve information in real-time.