

Using the SOA Gateway to access Shared Libraries or DLLs

Using the SOA Gateway to access Shared Libraries or DLLs

- Introduction
 - Building the DLL
 - SOA Gateway Configuration
 - Accessing DLL using the SOA Gateway
-

Introduction

You can also use the SOA Gateway to access existing shared libraries or DLLs without any rebuilding or reverse-engineering.

You just need to Know the name of the function you wish to call, and what the input and output parameters look like.

In this HOWTO, we'll build a piece of code into a DLL, and show you how to use the SOA Gateway to expose a function within this as a Web Service.

Building the DLL

Consider this simple piece of C++ code.

```
#include <string.h>

#ifdef WIN32
#define PLATFORM_EXPORT __declspec( dllexport )
#else
#define PLATFORM_EXPORT
#endif

extern "C" PLATFORM_EXPORT int calc( char operation[20], int *operand1, int *operand2, int *result );

extern "C" int calc( char operation[20], int *operand1, int *operand2, int *result ){

    if( !strcmp( operation, "add" ) ){
        *result = (*operand1) + (*operand2) ;
    }
    else if( !strcmp( operation, "sub" ) ){
        *result = (*operand1) - (*operand2) ;
    }
    else if( !strcmp( operation, "mul" ) ){
        *result = (*operand1) * (*operand2) ;
    }
    else if( !strcmp( operation, "div" ) ){
        *result = (*operand1) / (*operand2) ;
    }
    else{
        *result = 0;
        return -1;
    }
}
```

```

    }
    return 0;
}

```

This is simple calc method which will take a string, two integers, and return a integer based on the value of string.

For the purposes of this HOWTO, we are running the SOA Gateway on Windows, and will build the above code into a Windows DLL using MS VC++.

If the SOA Gateway is running on Linux, the DLL should be built there, and so on, depending on the platform.

A pre-built Windows (x86) DLL is available here

SOA Gateway Configuration

The SOA Gateway must now be configured to load and run this DLL.

Ideally, our new DLL will be available in the system PATH. On Unix-based systems, you should change to the directory where the DLL is located, and run the command `export PATH=$PATH:$PWD`. Now restart the SOA Gateway in this shell.

On z/OS, ensure the load library containing the DLL is in your STEPLIB concatenation.

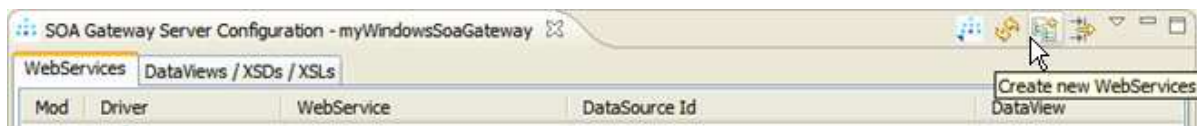
As Windows requires a restart before picking up a new system PATH, the simplest thing to do is copy the built DLL into the bin directory of your SOA Gateway installation. Assuming the default installation was used, this will be `C:\Program Files\Risaris Limited\SOA Gateway {v.r.m}\Apache\bin\` where `{v.r.m}` denotes the actual SOA Gateway server version in use.

Now start your SOA Gateway Control Center and choose one of the following methods of creating a WebService based on the "calc" example

- Create ("Discover") the WebService automatically
- Define the WebService manually

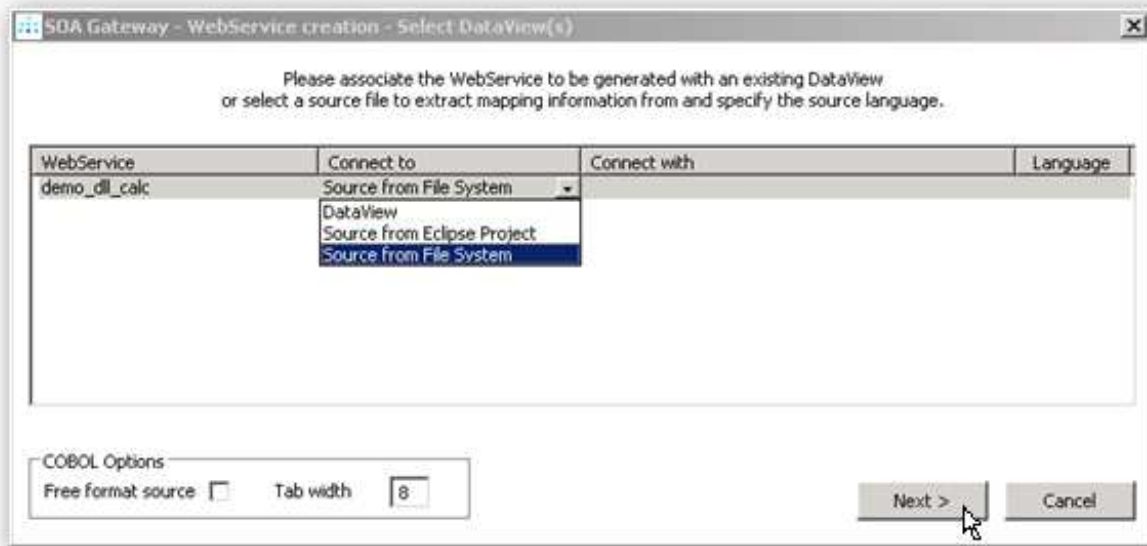
Create ("Discover") the WebService automatically

- Download the C-source to your local disk driver from here
- Start the discover process from the Configuration view by clicking the **Create new WebServices** icon.

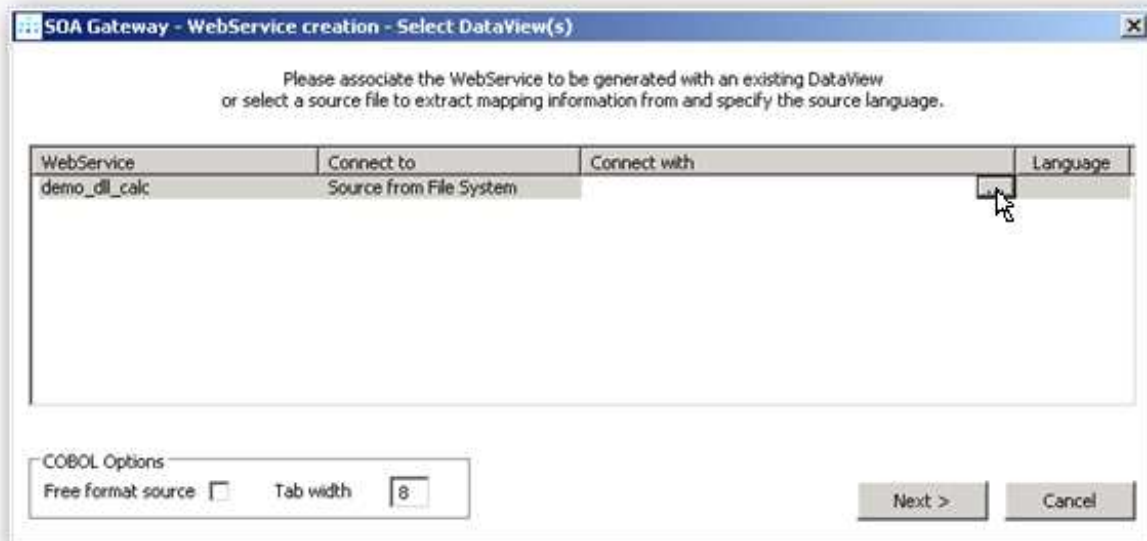


- Select the *DLL Driver* and click **Next >**.

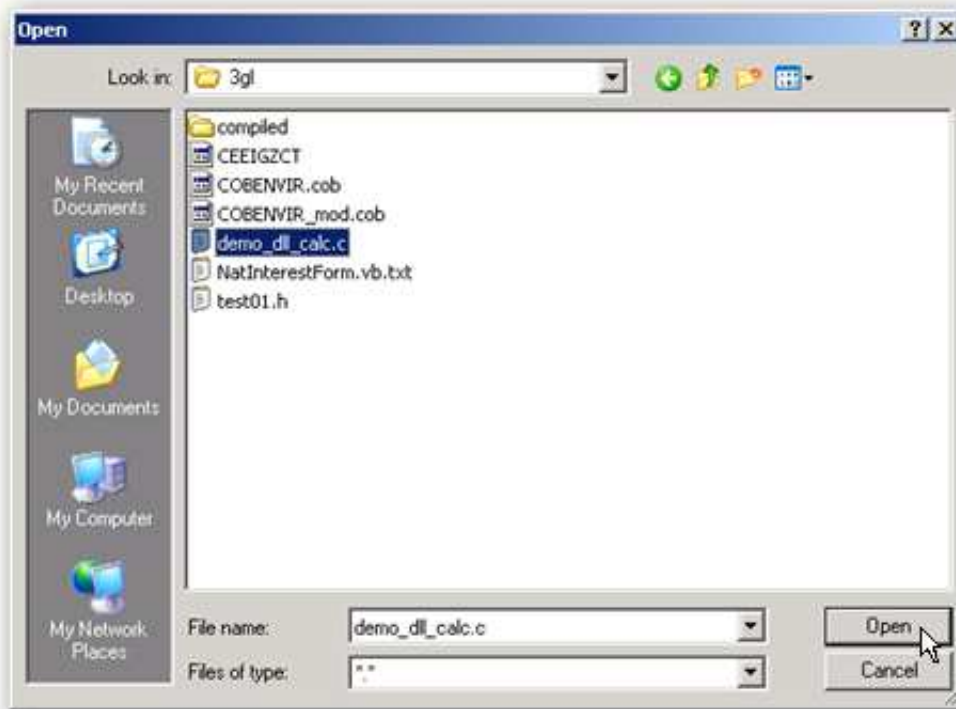
- You will now associate the DLL with the source, which will be used to generate the SOA Gateway DataView. From the *Connect to* column's dropdown box select *Source from File System*.



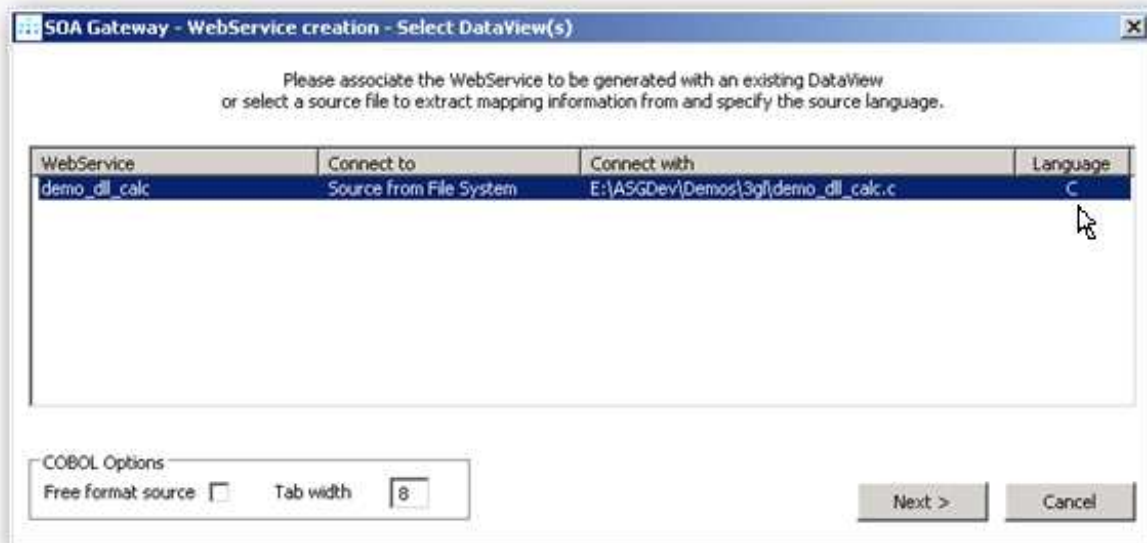
Advance to the *Connect with* column, click it, then click the push-button appearing at the right hand corner



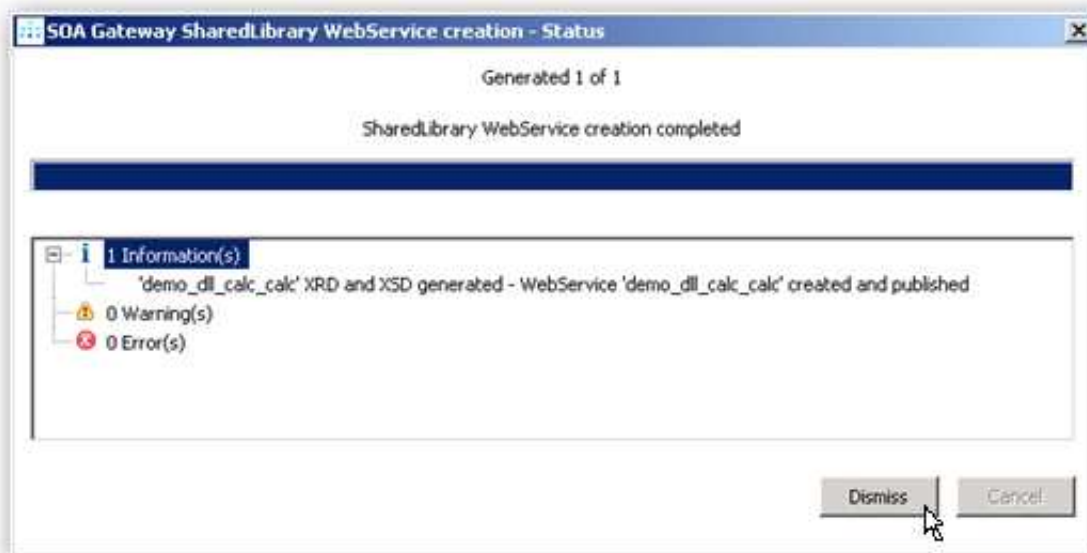
- In the File Dialog popping up, navigate to the location where the C-source for the demo DLL has been downloaded to, select it



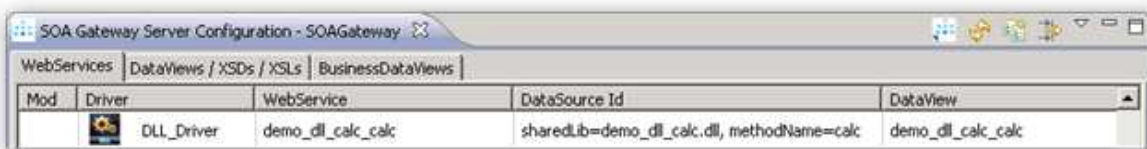
For files with an extension of *.c* or *.h* the *Language* will automatically be set to *C*. Click the **Next** button.



Progress and completion of the generation process will be shown with this dialog



- The WebService has been created and is usable

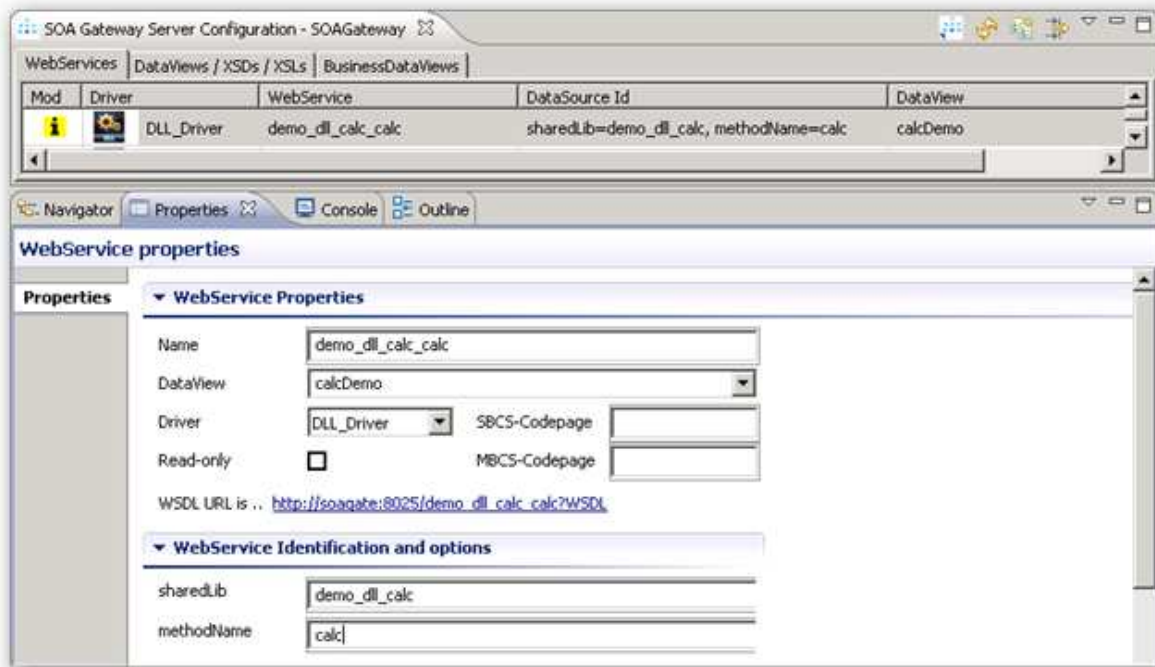


| Mod | Driver | WebService | DataSource Id | DataView |
|-----|------------|--------------------|---|--------------------|
| | DLL_Driver | demo_dll_calc_calc | shared.lib=demo_dll_calc.dll, methodName=calc | demo_dll_calc_calc |

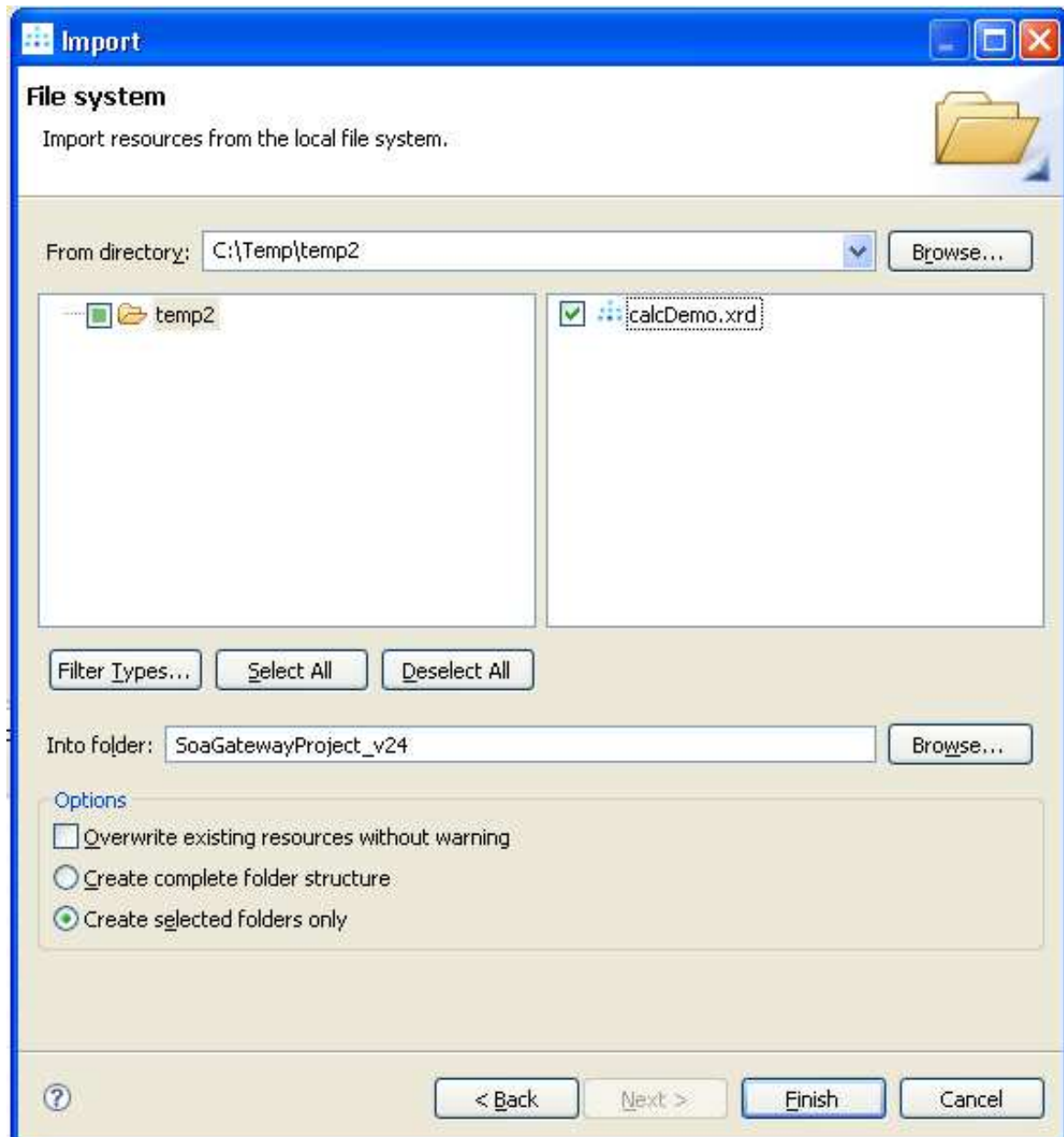
Define the WebService manually

- Add a new Web Service. See here for more information
- Add the following information
 - Name : "demo_dll_calc_calc"
 - DataView : "calcDemo"
 - Driver : "DLL Driver" (or equivalent on your system)
 - sharedLib : "demo_dll_calc.dll "(the name of the DLL you've built)
 - methodName : "calc"

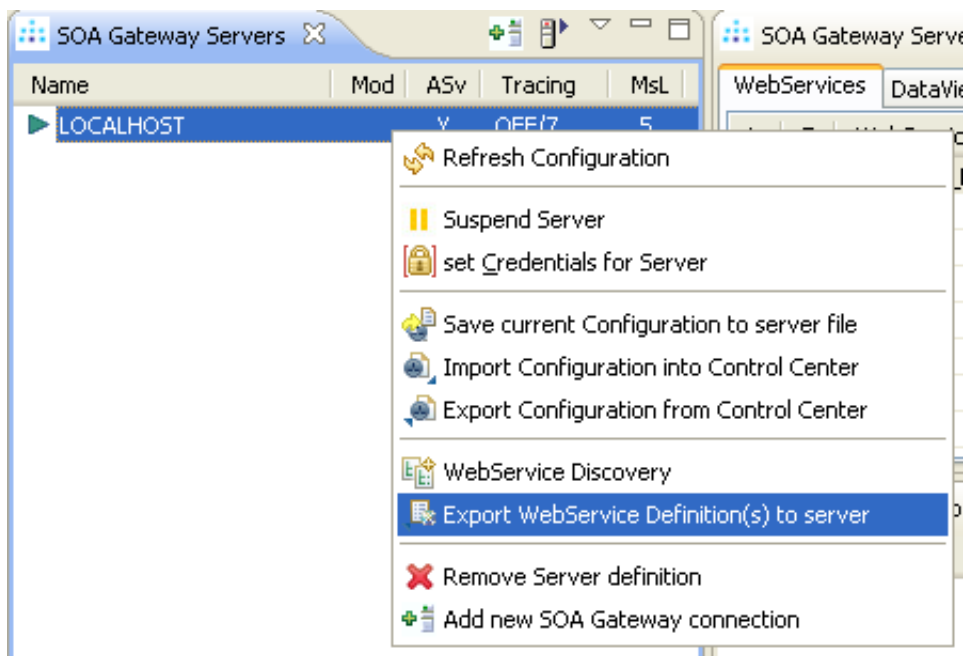
Click **Publish** to publish your changes to the server.



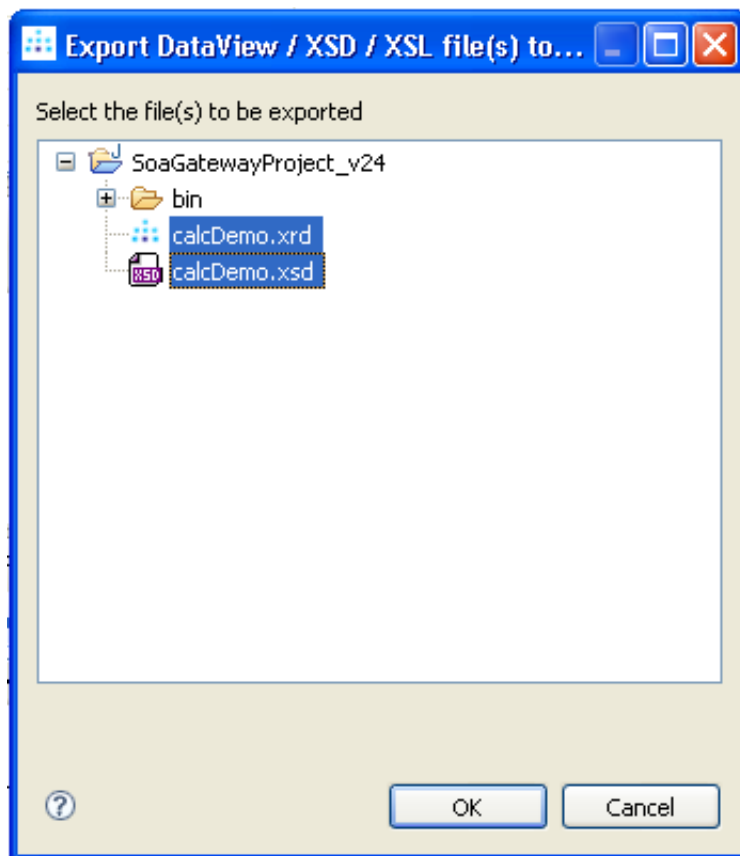
- Save the following XRD file to disk. This is file which maps the calc functions parameters to a format that the SOA Gateway can understand.
- Switch over to your Java/Resource perspective (see here for more info)
- Right-click your project, and select **Import**
- Expand **General** and select **File System**. Click **Next**
- Click **Browse** and select the directory where you saved the above XRD. Check the XRD, and click **Finish**



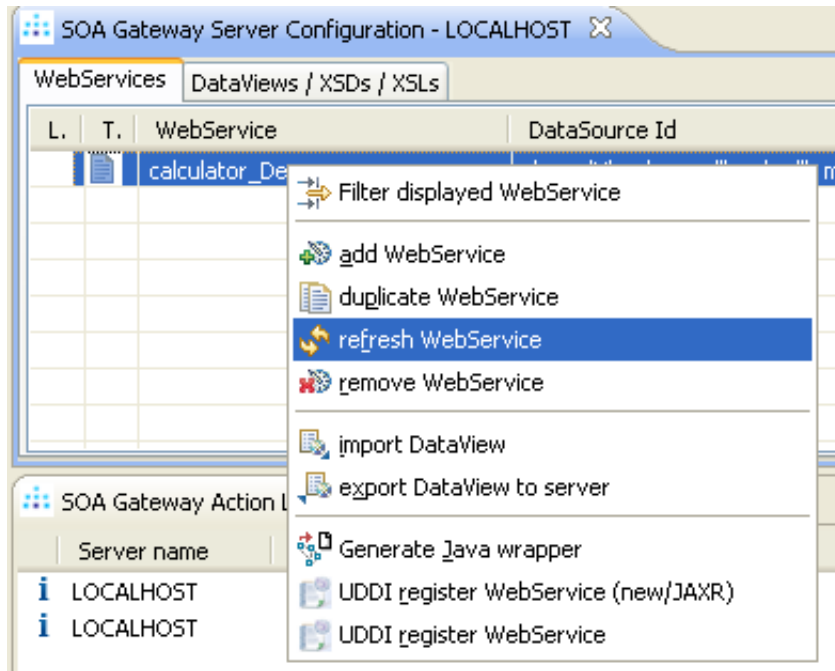
- Back to the SOA Gateway perspective, right-click your server, and select **Export Web Service Definitions to server**



- Select both the calcDemo files, and click **OK**



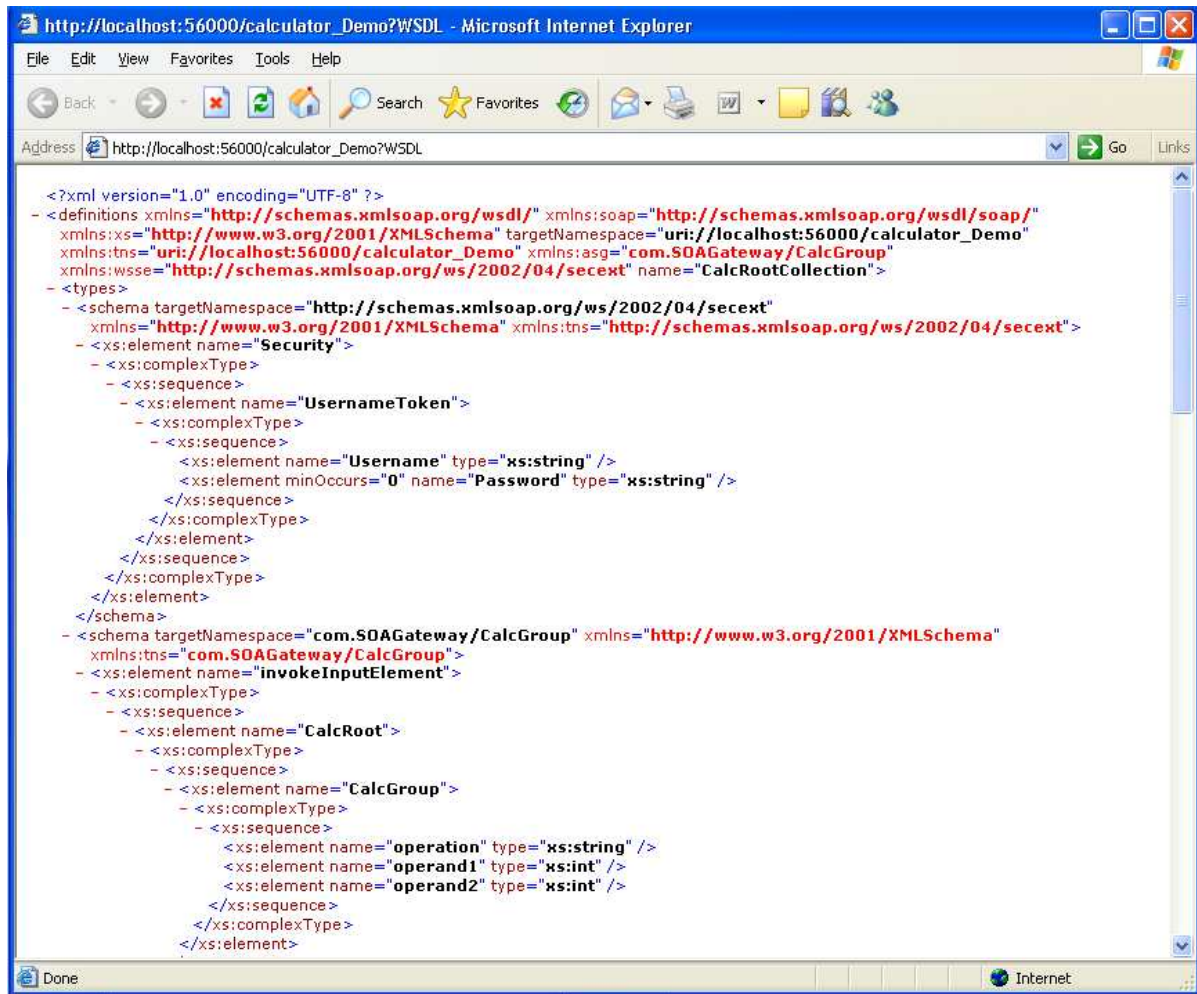
- In the configuration view, Right-click your Web Service and select **Refresh Web Service**



Accessing DLL using the SOA Gateway

This section will show you how to access our new DLL Web service.

- In the Web Service properties, click the WSDL URL to open your the WSDL for your web service in a browser.



```

<?xml version="1.0" encoding="UTF-8" ?>
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="uri://localhost:56000/calculator_Demo"
  xmlns:tns="uri://localhost:56000/calculator_Demo" xmlns:asg="com.SOAGateway/CalcGroup"
  xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext" name="CalcRootCollection">
- <types>
- <schema targetNamespace="http://schemas.xmlsoap.org/ws/2002/04/secext"
  xmlns="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://schemas.xmlsoap.org/ws/2002/04/secext">
- <xs:element name="Security">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="UsernameToken">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="Username" type="xs:string" />
- <xs:element minOccurs="0" name="Password" type="xs:string" />
- </xs:sequence>
- </xs:complexType>
- </xs:element>
- </xs:sequence>
- </xs:complexType>
- </xs:element>
- </schema>
- <schema targetNamespace="com.SOAGateway/CalcGroup" xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="com.SOAGateway/CalcGroup">
- <xs:element name="invokeInputElement">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="CalcRoot">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="CalcGroup">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="operation" type="xs:string" />
- <xs:element name="operand1" type="xs:int" />
- <xs:element name="operand2" type="xs:int" />
- </xs:sequence>
- </xs:complexType>
- </xs:sequence>
- </xs:complexType>
- </xs:element>

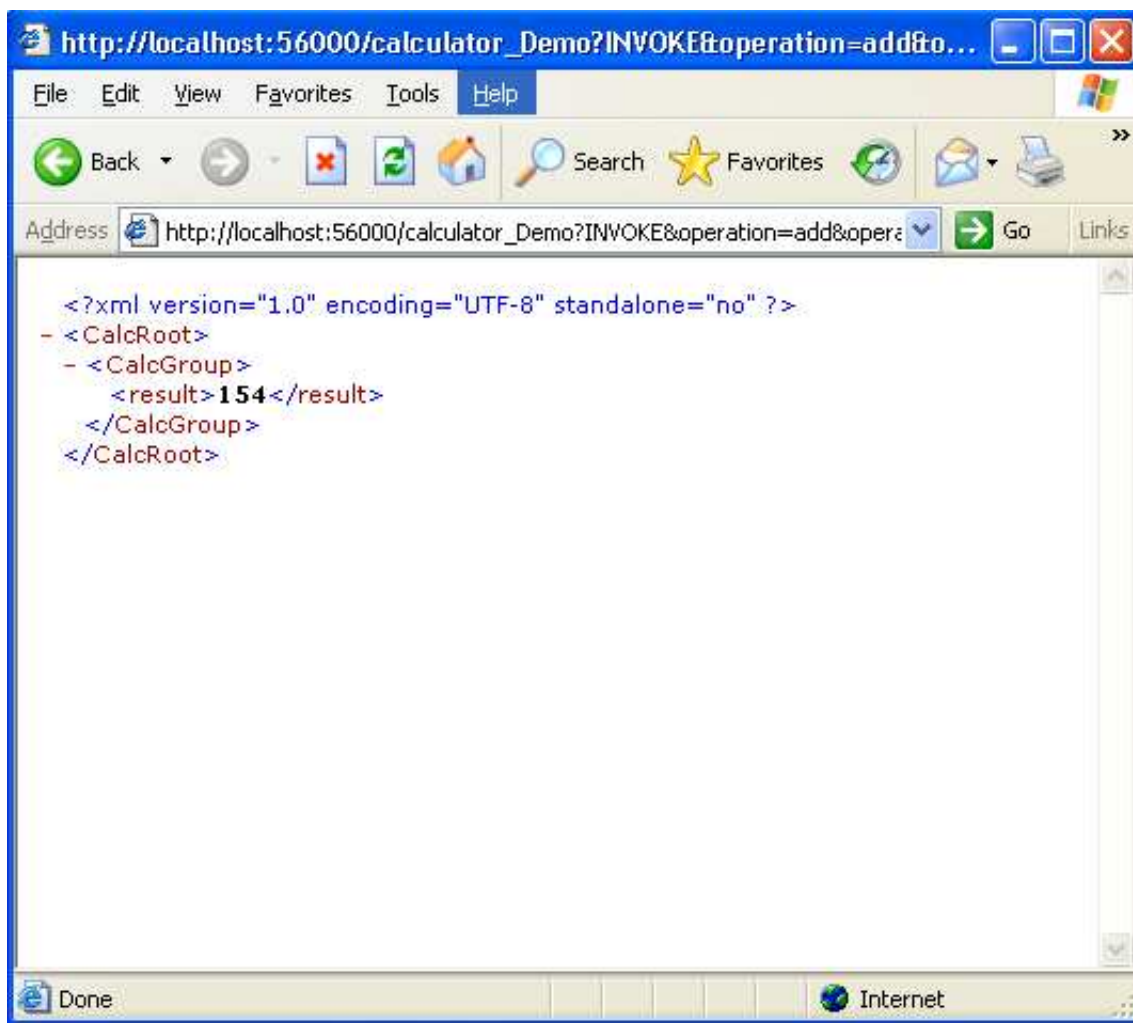
```

- You may import this WSDL to the WS Client of your choice, but for the purpose of this demo, we will use REST-style requests to access the DLL. More information about REST can be found here.

In the browser URL box, remove the *?WSDL* characters, and add the following

"?INVOKE&operation=add&operand1=55&operand2=99"

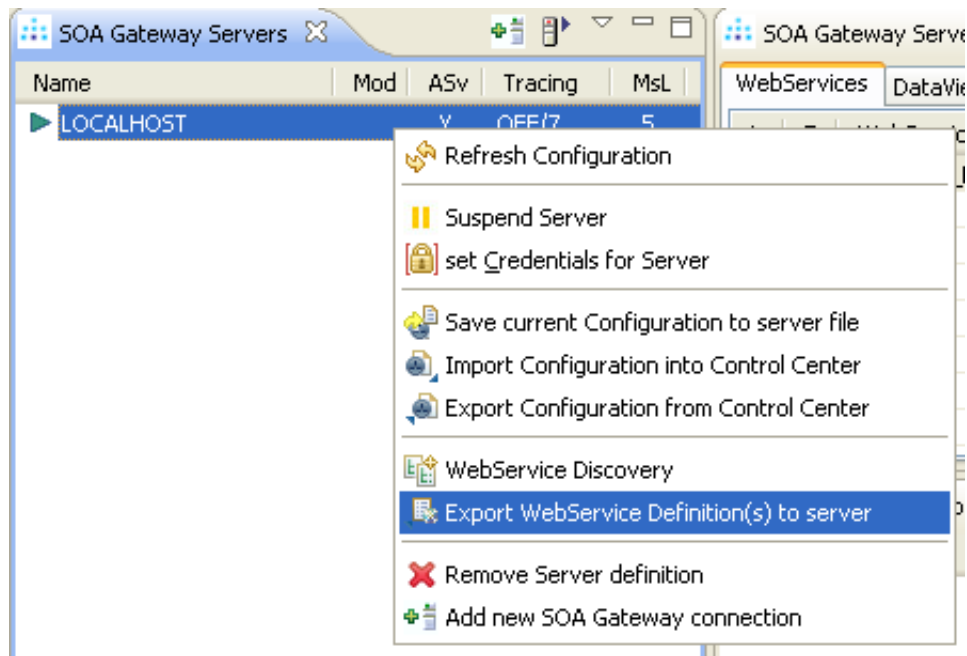
You should get a results something like this



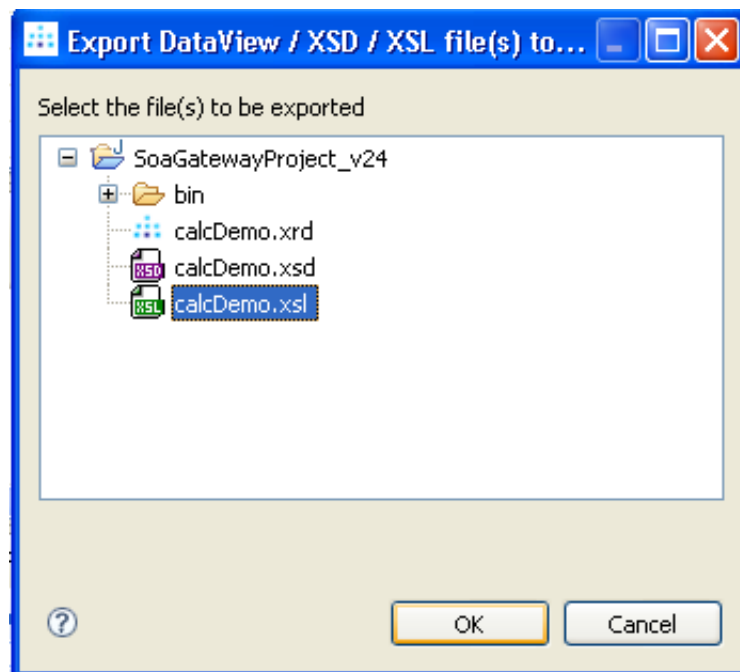
Congratulations! You have now accessed a DLL using the SOA Gateway

- If you look back at the original C program, you can see how the name=value pairs match up to the parameters of the calc function call. Try changing the operation and operands for different mathematical operations.

You can also use a XSL Stylesheet which can be used to change the appearance of the output XML. It is available here. As with the XRD previously, import this file into the project, and export it to the server.



And then ...



- Now when you refresh your browser, the stylesheet will be applied and the output should look like this

