**Example of using Custom service by using a Basic port**

We give a example using custom services. In this case user can give two numbers and what he like to do with this numbers. There is a choice for add, substract, divide and multiplicate the two numbers. This process is done in a Visual Studio console, but the logic is done in a Ax.

In this documentation it's decribed how to do this. To realize this we need a

In AX:

1) Class

2) Service

3) Service group

In Visual studio:

4) Set up service reference of the project

5) customize program.cs file (C#)

1) Class.

We add the method 'formula' . Here the logic for the service is done. The attribute should be defined in the first line:

[AifCollectionTypeAttribute('\_array', Types::Real), SysEntryPointAttribute(**true**)]

As we use an array as type, we need to define it as parameter in the AifCollectionTypeAttribute, so that we can use the method as a service operator for the service we gonna create in the next step.

Here is the complete code of the method:

[AifCollectionTypeAttribute('\_array', Types::Real), SysEntryPointAttribute(**true**)]

**public** **real** formula( **str** \_type,

Array \_array)

{

#define.Add("A")

#define.Substract("S")

#define.Multiplicate("M")

#define.Divide("D")

**int** i= **1**;

**real** result;

**try**

{

**for** (i = **1**; i <= \_array.lastIndex(); i++)

{

**switch** (\_type)

{

**case** #add :

result += \_array.value(i);

**break**;

**case** #Substract :

result = i == **1** ? \_array.value(i) : result - \_array.value(i);

**break**;

**case** #Multiplicate :

result = i == **1** ? \_array.value(i) : result \* \_array.value(i);

**break**;

**case** #Divide :

result = i == **1** ? \_array.value(i) : result / minone(\_array.value(i));

**break**;

**default** :

**throw** Exception::Error;

}

}

**return** result;

}

**catch** (Exception::Error)

{

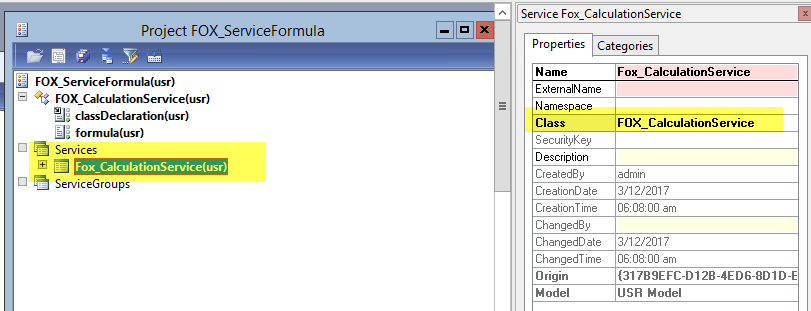
**throw** error("Option not exists");

}

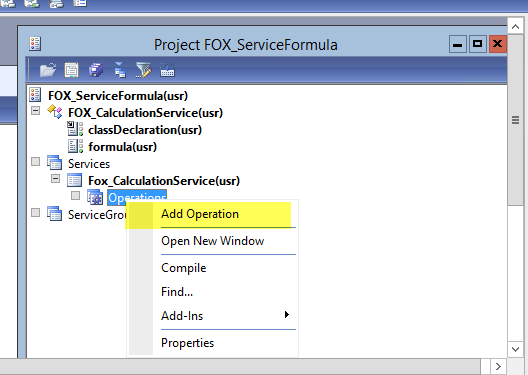
}

2. Service

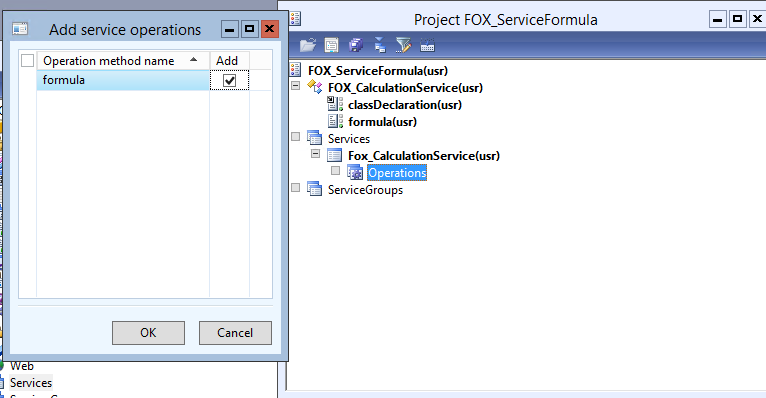
Now we going create a service. in the property menu, we connect the class to the service.



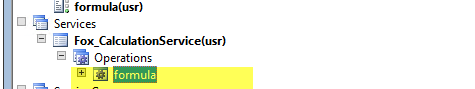
Add the operation, which is the method we created.



Click 'OK'



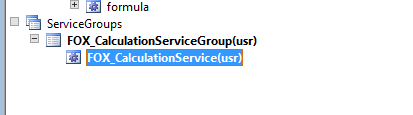
No the service operation is added to the service.



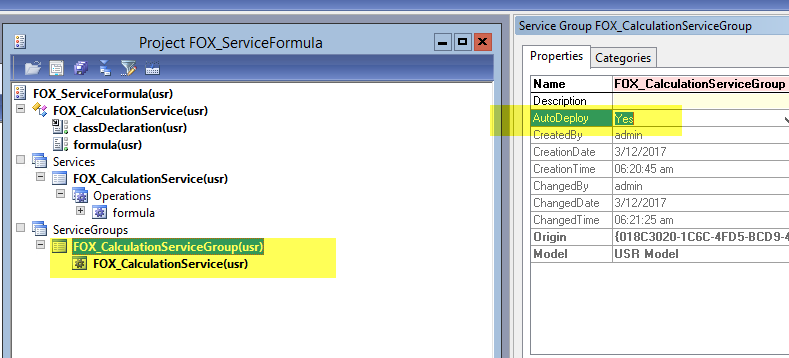
3. Service group.

Create a new servicegroup and drag and draw the service created at point 2) into to servicegroup.

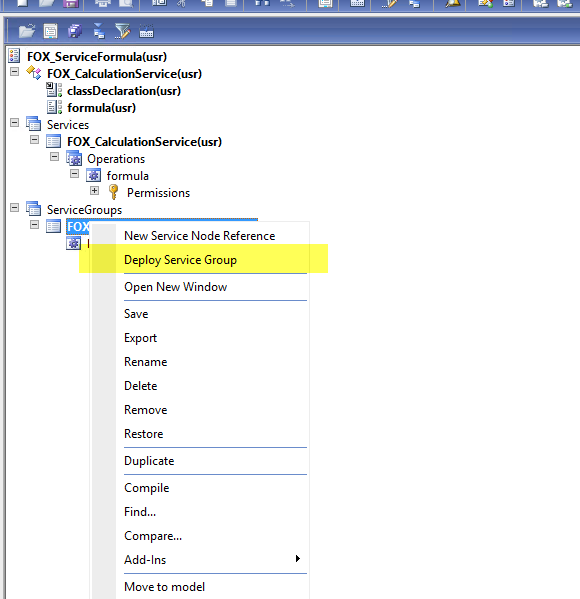
Service group looks like this:



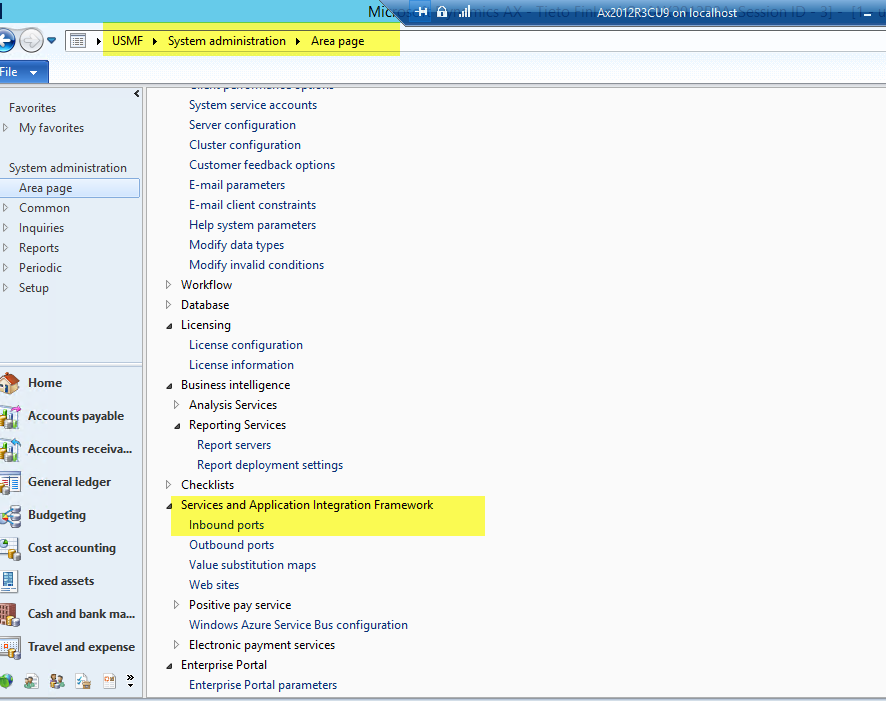
Set the property AutoDeploy to Yes.



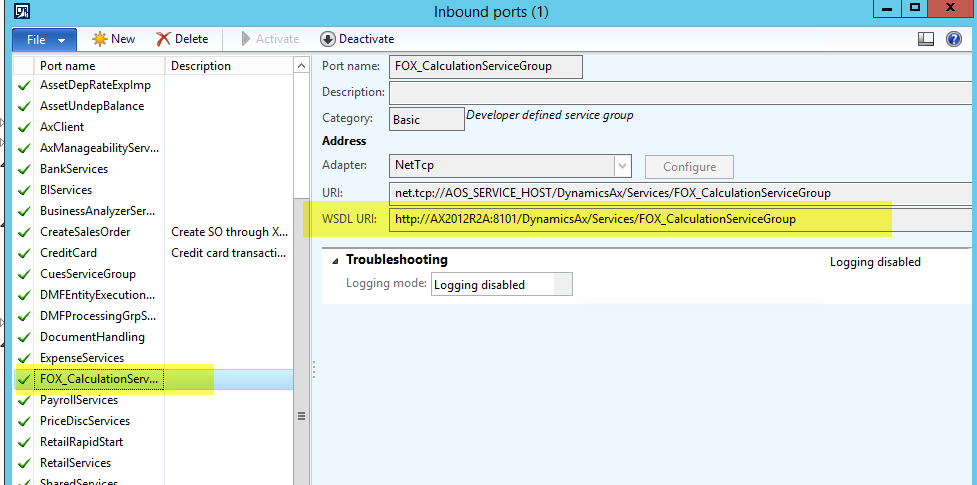
Deploy the service group.



Now we go to the System administration | setup | Services and Application Integration Framework | Inbound ports

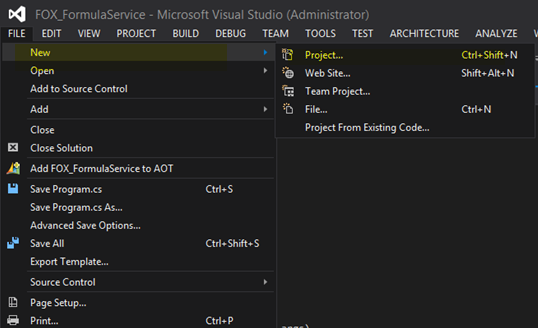


Here we see the service:

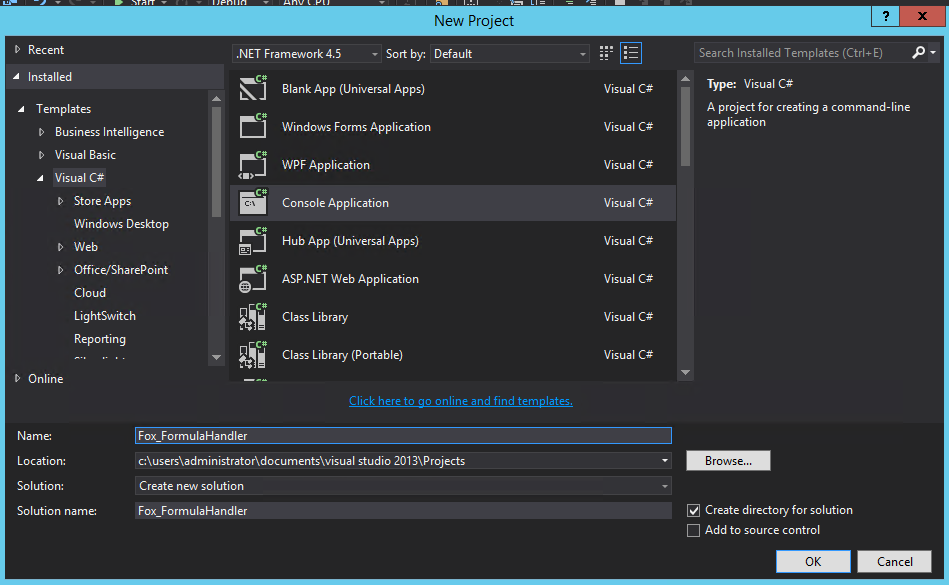


We need to remember the WSDL URI, as we need it for the ServiceReference in Visual Studio.

Now we have finished the Dynamics Ax part, now we continue in Visual studio.In visual studio create a new project via File | project

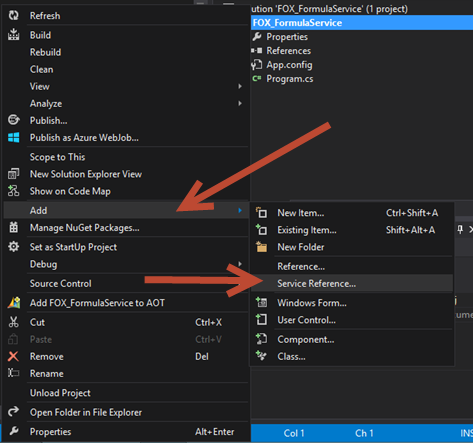


We create a new project, choose the Console Application template and give it a name.

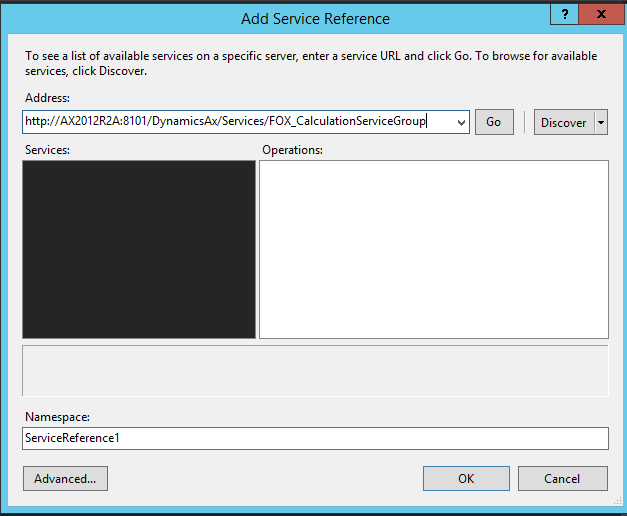


Now we going to add the Service reference. Service reference connect our service group we created in Ax with Visual Studio.

From the project name do a right mouse click. In the menu select Add | Service Reference

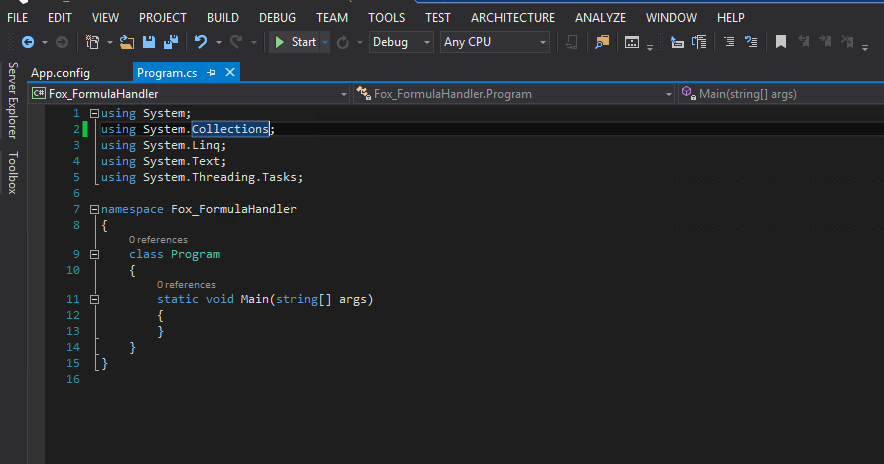


Now we paste the WSDL URI from the Ax Service group here. Click OK and wait until Visual Studio is done.

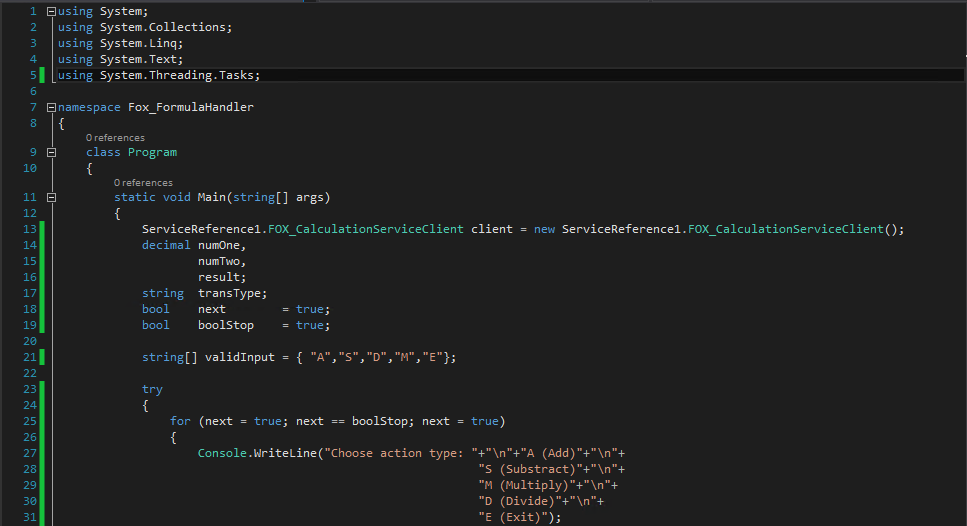


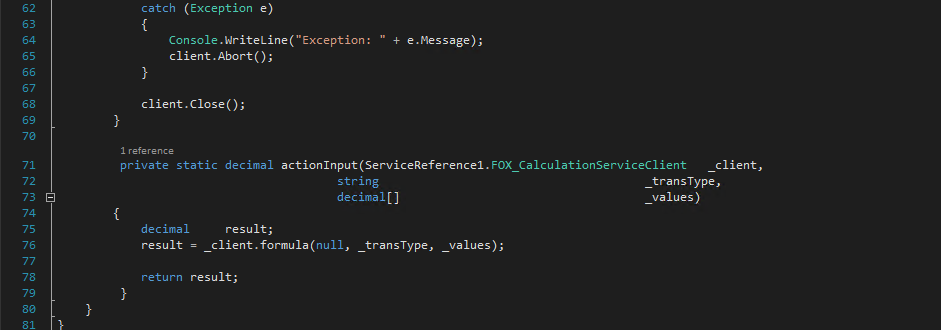
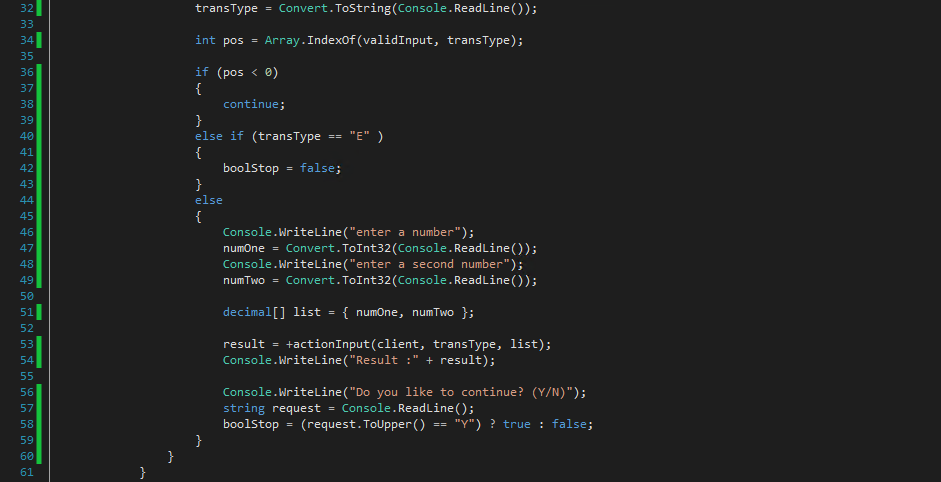
Now we need make some changes in Program.cs with C#.

By default it looks like this:



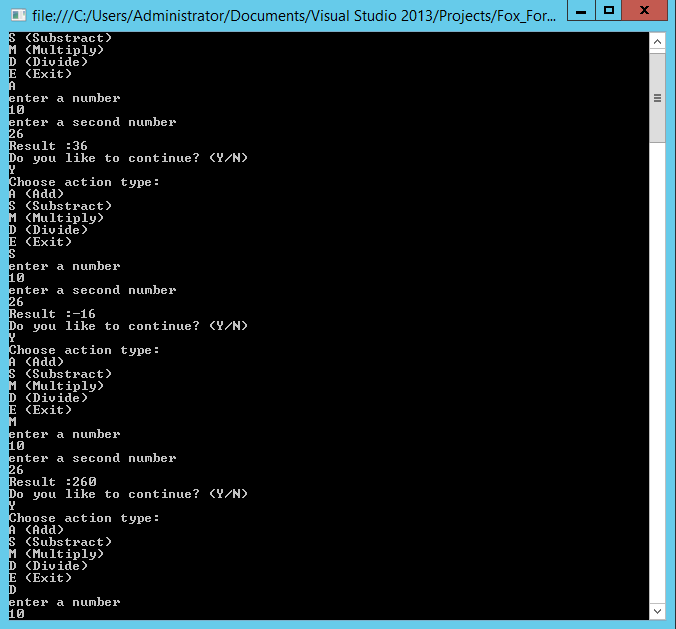
After modificfation code in the program.cs looks like this:







When we build and run the program in Visual Studio, it look like this:



Appendix: C# code in program.cs file

using System;

using System.Collections;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Fox\_FormulaHandler

{

class Program

{

static void Main(string[] args)

{

ServiceReference1.FOX\_CalculationServiceClient client = new ServiceReference1.FOX\_CalculationServiceClient();

decimal numOne,

numTwo,

result;

string transType;

bool next = true;

bool boolStop = true;

string[] validInput = { "A","S","D","M","E"};

try

{

for (next = true; next == boolStop; next = true)

{

Console.WriteLine("Choose action type: "+"\n"+"A (Add)"+"\n"+

"S (Substract)"+"\n"+

"M (Multiply)"+"\n"+

"D (Divide)"+"\n"+

"E (Exit)");

transType = Convert.ToString(Console.ReadLine());

int pos = Array.IndexOf(validInput, transType);

if (pos < 0)

{

continue;

}

else if (transType == "E" )

{

boolStop = false;

}

else

{

Console.WriteLine("enter a number");

numOne = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter a second number");

numTwo = Convert.ToInt32(Console.ReadLine());

decimal[] list = { numOne, numTwo };

result = +actionInput(client, transType, list);

Console.WriteLine("Result :" + result);

Console.WriteLine("Do you like to continue? (Y/N)");

string request = Console.ReadLine();

boolStop = (request.ToUpper() == "Y") ? true : false;

}

}

}

catch (Exception e)

{

Console.WriteLine("Exception: " + e.Message);

client.Abort();

}

client.Close();

}

private static decimal actionInput(ServiceReference1.FOX\_CalculationServiceClient \_client,

string \_transType,

decimal[] \_values)

{

decimal result;

result = \_client.formula(null, \_transType, \_values);

return result;

}

}

}