ASMX Client with a WCF Service

[Download sample](http://go.microsoft.com/fwlink/?LinkId=87352)

This sample demonstrates how to create a service using Windows Communication Foundation (WCF) and then access the service from a non-WCF client, such as an ASMX client.

|  |
| --- |
| **http://msdn2.microsoft.com/en-us/library/ms751433.note%28en-us,VS.90%29.gifNote:** |
| The setup procedure and build instructions for this sample are located at the end of this topic. |

This sample consists of a client console program (.exe) and a service library (.dll) hosted by Internet Information Services (IIS). The service implements a contract that defines a request-reply communication pattern. The contract is defined by the ICalculator interface, which exposes math operations (Add, Subtract, Multiply, and Divide). The ASMX client makes synchronous requests to a math operation and the service replies with the result.

The service implements an ICalculator contract as defined in the following code.

[ServiceContract(Namespace="http://Microsoft.ServiceModel.Samples"), XmlSerializerFormat]

public interface ICalculator

{

[OperationContract]

double Add(double n1, double n2);

[OperationContract]

double Subtract(double n1, double n2);

[OperationContract]

double Multiply(double n1, double n2);

[OperationContract]

double Divide(double n1, double n2);

}

The [DataContractSerializer](http://msdn2.microsoft.com/en-us/library/ms405768.aspx) and [XmlSerializer](http://msdn2.microsoft.com/en-us/library/swxzdhc0.aspx) map CLR types to an XML representation. The **DataContractSerializer** interprets some XML representations differently than XmlSerializer. Non-WCF proxy generators, such as Wsdl.exe, generate a more usable interface when the XmlSerializer is being used. The [XmlSerializerFormatAttribute](http://msdn2.microsoft.com/en-us/library/ms586952.aspx) is applied to the ICalculator interface, to ensure that the XmlSerializer is used for mapping CLR types to XML. The service implementation calculates and returns the appropriate result.

The service exposes a single endpoint for communicating with the service, defined using a configuration file (Web.config). The endpoint consists of an address, a binding, and a contract. The service exposes the endpoint at the base address provided by the Internet Information Services (IIS) host. The **binding** attribute is set to basicHttpBinding that provides HTTP communications using SOAP 1.1, which is compliant with WS-I BasicProfile 1.1 as shown in the following sample configuration.

<services>

<service

name="Microsoft.ServiceModel.Samples.CalculatorService"

behaviorConfiguration="CalculatorServiceBehavior">

<!-- This endpoint is exposed at the base address provided by the host: http://localhost/servicemodelsamples/service.svc. --> <endpoint address=""

binding="basicHttpBinding"

contract="Microsoft.ServiceModel.Samples.ICalculator" />

</service>

</services>

The ASMX client communicates with the WCF service using a typed proxy that is generated by the Web Services Description Language (WSDL) utility (Wsdl.exe). The typed proxy is contained in the file generatedClient.cs. The WSDL utility retrieves metadata for the specified service and generates a typed proxy for use by a client to communicate. By default, the framework does not expose any metadata. To expose the metadata required to generate the proxy, you must add a [<serviceMetadata> Element](http://msdn2.microsoft.com/en-us/library/ms731317.aspx) and set its **httpGetEnabled** attribute to **True** as shown in the following configuration.

<behaviors>

<serviceBehaviors>

<behavior name="CalculatorServiceBehavior">

<!-- Setting httpGetEnabled to True on the serviceMetadata

behavior exposes the service's wsdl at <base address>?wsdl :

http://localhost/servicemodelsamples/service.svc?wsdl -->

<serviceMetadata httpGetEnabled="True"/>

<serviceDebug includeExceptionDetailInFaults="False" />

</behavior>

</serviceBehaviors>

</behaviors>

Run the following command from a command prompt in the client directory to generate the typed proxy.

wsdl /n:Microsoft.ServiceModel.Samples /o:generatedClient.cs /urlkey:CalculatorServiceAddress http://localhost/servicemodelsamples/service.svc?wsdl

By using the generated typed proxy, the client can access a given service endpoint by configuring the appropriate address. The client uses a configuration file (App.config) to specify the endpoint to communicate with.

<appSettings>

<add key="CalculatorServiceAddress"

value="http://localhost/ServiceModelSamples/service.svc"/>

</appSettings>

The client implementation constructs an instance of the typed proxy to begin communicating with the service.

// Create a client to the CalculatorService.

using (CalculatorService client = new CalculatorService())

{

// Call the Add service operation.

double value1 = 100.00D;

double value2 = 15.99D;

double result = client.Add(value1, value2);

Console.WriteLine("Add({0},{1}) = {2}", value1, value2, result);

// Call the Subtract service operation.

value1 = 145.00D;

value2 = 76.54D;

result = client.Subtract(value1, value2);

Console.WriteLine("Subtract({0},{1}) = {2}", value1, value2, result);

// Call the Multiply service operation.

value1 = 9.00D;

value2 = 81.25D;

result = client.Multiply(value1, value2);

Console.WriteLine("Multiply({0},{1}) = {2}", value1, value2, result);

// Call the Divide service operation.

value1 = 22.00D;

value2 = 7.00D;

result = client.Divide(value1, value2);

Console.WriteLine("Divide({0},{1}) = {2}", value1, value2, result);

}

Console.WriteLine();

Console.WriteLine("Press <ENTER> to terminate client.");

Console.ReadLine();

When you run the sample, the operation requests and responses are displayed in the client console window. Press ENTER in the client window to shut down the client.

Add(100,15.99) = 115.99

Subtract(145,76.54) = 68.46

Multiply(9,81.25) = 731.25

Divide(22,7) = 3.14285714285714

Press <ENTER> to terminate client.

**To set up, build, and run the sample**

1. Ensure that you have performed the [One-Time Set Up Procedure for the Windows Communication Foundation Samples](http://msdn2.microsoft.com/en-us/library/ms751527.aspx).
2. To build the C# or Visual Basic .NET edition of the solution, follow the instructions in [Building the Windows Communication Foundation Samples](http://msdn2.microsoft.com/en-us/library/ms751423.aspx).
3. To run the sample in a single- or cross-machine configuration, follow the instructions in [Running the Windows Communication Foundation Samples](http://msdn2.microsoft.com/en-us/library/ms752240.aspx).