**How to: Create a Service with a Contract Interface**

The preferred way to create a Windows Communication Foundation (WCF) contract is by using an interface. This contract specifies the collection and structure of messages required to access the operations the service offers. This interface defines the input and output types by applying the [ServiceContractAttribute](http://msdn.microsoft.com/en-us/library/system.servicemodel.servicecontractattribute.aspx) class to the interface and the [OperationContractAttribute](http://msdn.microsoft.com/en-us/library/system.servicemodel.operationcontractattribute.aspx) class to the methods that you want to expose.

For more information about service contracts, see [Designing Service Contracts](http://msdn.microsoft.com/en-us/library/ms733070.aspx).

**Creating a WCF contract with an interface**

1. Create a new interface using Visual Basic, C#, or any other common language runtime language.
2. Apply the **ServiceContractAttribute** class to the interface.
3. Define the methods in the interface.
4. Apply the **OperationContractAttribute** class to each method that must be exposed as part of the public WCF contract.

**Example**

The following code example shows an interface that defines a service contract.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl04_code');" \o "Copy Code)

<ServiceContract()> \_

Public Interface ICalculator

<OperationContract()> \_

Function Add(ByVal n1 As Double, ByVal n2 As Double) As Double

<OperationContract()> \_

Function Subtract(ByVal n1 As Double, ByVal n2 As Double) As Double

<OperationContract()> \_

Function Multiply(ByVal n1 As Double, ByVal n2 As Double) As Double

<OperationContract()> \_

Function Divide(ByVal n1 As Double, ByVal n2 As Double) As Double

End Interface

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl05_code');" \o "Copy Code)

using System.ServiceModel;

[ServiceContract]

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 methods that have the **OperationContractAttribute** class applied use a request-reply message pattern by default. For more information about this message pattern, see [How to: Create a Request-Reply Contract](http://msdn.microsoft.com/en-us/library/ms732381.aspx). You can also create and use other message patterns by setting properties of the attribute. For more examples, see [How to: Create a One-Way Contract](http://msdn.microsoft.com/en-us/library/ms733035.aspx) and [How to: Create a Duplex Contract](http://msdn.microsoft.com/en-us/library/ms731184.aspx).