**How to: Create a Duplex Contract**

This topic shows the basic steps to create methods that use a duplex (two-way) contract. A duplex contract allows clients and servers to communicate with each other independently so that either can initiate calls to the other. The duplex contract is one of three message patterns available to Windows Communication Foundation (WCF) services. The other two message patterns are one-way and request-reply. A duplex contract consists of two one-way contracts between the client and the server and does not require that the method calls be correlated. Use this kind of contract when your service must query the client for more information or explicitly raise events on the client. For more information about creating a client application for a duplex contract, see [How to: Access Services with a Duplex Contract](http://msdn.microsoft.com/en-us/library/ms731935.aspx). For a working sample, see the [Duplex](http://msdn.microsoft.com/en-us/library/ms752216.aspx) sample.

**To create a duplex contract**

1. Create the interface that makes up the server side of the duplex contract.
2. Apply the [ServiceContractAttribute](http://msdn.microsoft.com/en-us/library/system.servicemodel.servicecontractattribute.aspx) class to the interface.

Visual Basic

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

<ServiceContract(Namespace:="http://Microsoft.ServiceModel.Samples", SessionMode:=SessionMode.Required, \_

CallbackContract:=GetType(ICalculatorDuplexCallback))> \_

Public Interface ICalculatorDuplex

<OperationContract(IsOneWay:=True)> \_

Sub Clear()

<OperationContract(IsOneWay:=True)> \_

Sub AddTo(ByVal n As Double)

<OperationContract(IsOneWay:=True)> \_

Sub SubtractFrom(ByVal n As Double)

<OperationContract(IsOneWay:=True)> \_

Sub MultiplyBy(ByVal n As Double)

<OperationContract(IsOneWay:=True)> \_

Sub DivideBy(ByVal n As Double)

End Interface

C#

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

[ServiceContract(Namespace = "http://Microsoft.ServiceModel.Samples", SessionMode=SessionMode.Required,

CallbackContract=typeof(ICalculatorDuplexCallback))]

public interface ICalculatorDuplex

{

[OperationContract(IsOneWay=true)]

void Clear();

[OperationContract(IsOneWay = true)]

void AddTo(double n);

[OperationContract(IsOneWay = true)]

void SubtractFrom(double n);

[OperationContract(IsOneWay = true)]

void MultiplyBy(double n);

[OperationContract(IsOneWay = true)]

void DivideBy(double n);

}

1. Declare the method signatures in the interface.
2. Apply the [OperationContractAttribute](http://msdn.microsoft.com/en-us/library/system.servicemodel.operationcontractattribute.aspx) class to each method signature that must be part of the public contract.
3. Create the callback interface that defines the set of operations that the service can invoke on the client.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl07_code');" \o "Copy Code)

Public Interface ICalculatorDuplexCallback

<OperationContract(IsOneWay:=True)> \_

Sub Equals(ByVal result As Double)

<OperationContract(IsOneWay:=True)> \_

Sub Equation(ByVal eqn As String)

end interface

C#

public interface ICalculatorDuplexCallback

{

[OperationContract(IsOneWay = true)]

void Equals(double result);

[OperationContract(IsOneWay = true)]

void Equation(string eqn);

}

1. Declare the method signatures in the callback interface.
2. Apply the **OperationContractAttribute** class to each method signature that must be part of the public contract.
3. Link the two interfaces into a duplex contract by setting the [CallbackContract](http://msdn.microsoft.com/en-us/library/system.servicemodel.servicecontractattribute.callbackcontract.aspx) property in the primary interface to the type of the callback interface.

**To call methods on the client**

1. In the service's implementation of the primary contract, declare a variable for the callback interface.
2. Set the variable to the object reference returned by the [GetCallbackChannel](http://msdn.microsoft.com/en-us/library/ms575542.aspx) method of the [OperationContext](http://msdn.microsoft.com/en-us/library/system.servicemodel.operationcontext.aspx) class.

Visual Basic

Dim callback As ICalculatorDuplexCallback

C#

ICalculatorDuplexCallback callback = null;

Visual Basic

callback = OperationContext.Current.GetCallbackChannel(Of ICalculatorDuplexCallback)()

C#

callback = OperationContext.Current.GetCallbackChannel<ICalculatorDuplexCallback>();

1. Call the methods defined by the callback interface.

**Example**

The following code example demonstrates duplex communication. The service’s contract contains service operations for moving forward and backward. The client’s contract contains a service operation for reporting its position.

Visual Basic

' Define a duplex service contract.

' A duplex contract consists of two interfaces.

' The primary interface is used to send messages from client to service.

' The callback interface is used to send messages from service back to client.

' ICalculatorDuplex allows one to perform multiple operations on a running result.

' The result is sent back after each operation on the ICalculatorCallback interface.

<ServiceContract(Namespace:="http://Microsoft.ServiceModel.Samples", SessionMode:=SessionMode.Required, \_

CallbackContract:=GetType(ICalculatorDuplexCallback))> \_

Public Interface ICalculatorDuplex

<OperationContract(IsOneWay:=True)> \_

Sub Clear()

<OperationContract(IsOneWay:=True)> \_

Sub AddTo(ByVal n As Double)

<OperationContract(IsOneWay:=True)> \_

Sub SubtractFrom(ByVal n As Double)

<OperationContract(IsOneWay:=True)> \_

Sub MultiplyBy(ByVal n As Double)

<OperationContract(IsOneWay:=True)> \_

Sub DivideBy(ByVal n As Double)

End Interface

' The callback interface is used to send messages from service back to client.

' The Equals operation will return the current result after each operation.

' The Equation opertion will return the complete equation after Clear() is called.

Public Interface ICalculatorDuplexCallback

<OperationContract(IsOneWay:=True)> \_

Sub Equals(ByVal result As Double)

<OperationContract(IsOneWay:=True)> \_

Sub Equation(ByVal eqn As String)

end interface

' Service class which implements a duplex service contract.

' Use an InstanceContextMode of PerSession to store the result

' An instance of the service will be bound to each duplex session

<ServiceBehavior(InstanceContextMode:=InstanceContextMode.PerSession)> \_

Public Class CalculatorService

Implements ICalculatorDuplex

Dim result As Double

Dim equation As String

Dim callback As ICalculatorDuplexCallback

Public Sub New()

result = 0D

equation = result.ToString()

callback = OperationContext.Current.GetCallbackChannel(Of ICalculatorDuplexCallback)()

End Sub

Public Sub AddTo(ByVal n As Double) Implements ICalculatorDuplex.AddTo

result += n

equation += " + " + n.ToString()

callback.Equals(result)

End Sub

Public Sub Clear() Implements ICalculatorDuplex.Clear

callback.Equation(equation + " = " + result.ToString())

result = 0D

equation = result.ToString()

End Sub

Public Sub DivideBy(ByVal n As Double) Implements ICalculatorDuplex.DivideBy

result /= n

equation += " / " + n.ToString()

callback.Equals(result)

End Sub

Public Sub MultiplyBy(ByVal n As Double) Implements ICalculatorDuplex.MultiplyBy

result \*= n

equation += " \* " + n.ToString()

callback.Equals(result)

End Sub

Public Sub SubtractFrom(ByVal n As Double) Implements ICalculatorDuplex.SubtractFrom

result -= n

equation += " - " + n.ToString()

callback.Equals(result)

End Sub

End Class

C#

// Define a duplex service contract.

// A duplex contract consists of two interfaces.

// The primary interface is used to send messages from client to service.

// The callback interface is used to send messages from service back to client.

// ICalculatorDuplex allows one to perform multiple operations on a running result.

// The result is sent back after each operation on the ICalculatorCallback interface.

[ServiceContract(Namespace = "http://Microsoft.ServiceModel.Samples", SessionMode=SessionMode.Required,

CallbackContract=typeof(ICalculatorDuplexCallback))]

public interface ICalculatorDuplex

{

[OperationContract(IsOneWay=true)]

void Clear();

[OperationContract(IsOneWay = true)]

void AddTo(double n);

[OperationContract(IsOneWay = true)]

void SubtractFrom(double n);

[OperationContract(IsOneWay = true)]

void MultiplyBy(double n);

[OperationContract(IsOneWay = true)]

void DivideBy(double n);

}

// The callback interface is used to send messages from service back to client.

// The Equals operation will return the current result after each operation.

// The Equation opertion will return the complete equation after Clear() is called.

public interface ICalculatorDuplexCallback

{

[OperationContract(IsOneWay = true)]

void Equals(double result);

[OperationContract(IsOneWay = true)]

void Equation(string eqn);

}

// Service class which implements a duplex service contract.

// Use an InstanceContextMode of PerSession to store the result

// An instance of the service will be bound to each duplex session

[ServiceBehavior(InstanceContextMode = InstanceContextMode.PerSession)]

public class CalculatorService : ICalculatorDuplex

{

double result;

string equation;

ICalculatorDuplexCallback callback = null;

public CalculatorService()

{

result = 0.0D;

equation = result.ToString();

callback = OperationContext.Current.GetCallbackChannel<ICalculatorDuplexCallback>();

}

public void Clear()

{

callback.Equation(equation + " = " + result.ToString());

result = 0.0D;

equation = result.ToString();

}

public void AddTo(double n)

{

result += n;

equation += " + " + n.ToString();

callback.Equals(result);

}

public void SubtractFrom(double n)

{

result -= n;

equation += " - " + n.ToString();

callback.Equals(result);

}

public void MultiplyBy(double n)

{

result \*= n;

equation += " \* " + n.ToString();

callback.Equals(result);

}

public void DivideBy(double n)

{

result /= n;

equation += " / " + n.ToString();

callback.Equals(result);

}

}

* Applying the **ServiceContractAttribute** and **OperationContractAttribute** attributes allows the automatic generation of service contract definitions in the Web Services Description Language (WSDL).
* Use the [ServiceModel Metadata Utility Tool (Svcutil.exe)](http://msdn.microsoft.com/en-us/library/aa347733.aspx) to retrieve the WSDL document and (optional) code and configuration for a client.
* Endpoints exposing duplex services must be secured. When a service receives a duplex message, it looks at the ReplyTo in that incoming message to determine where to send the reply. If the channel is not secured, then an untrusted client could send a malicious message with a target machine's ReplyTo, leading to a denial of service of the target machine. With regular request-reply messages, this is not an issue, because the ReplyTo is ignored and the response is sent on the channel the original message came in on.