# How to use duplex and TCP binding in WCF in UWP app

## Introduction

This sample demonstrates how to create a duplex and TCP binding WCF and how to use it in UWP app.

## Building the sample

This sample should be run in Microsoft Visual Studio 2015 version and Windows 10.

Before you build the project, make sure you have installed Microsoft.NETCore.UniversalWindowsPlatform package in the project. The following steps can help you to install it:

* Open the solution CSUWPDuplex.sln.
* Right click the CSUWPDuplex project and select [Manage NuGet Packages...].
* Search Microsoft.NETCore.UniversalWindowsPlatform from the Browse tab page. Find the right package and then install it.

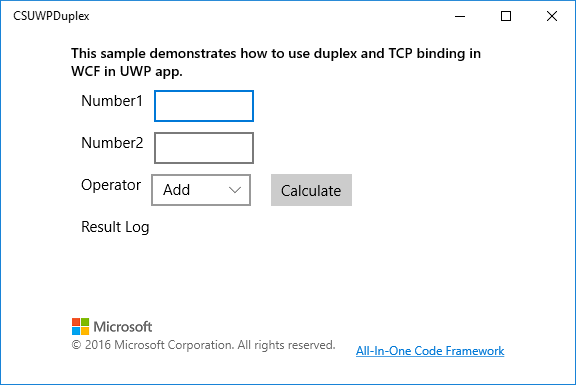
Please notice that Microsoft.NETCore.UniversalWindowsPlatform package version should be 5.2.2 or above.

## Running the sample

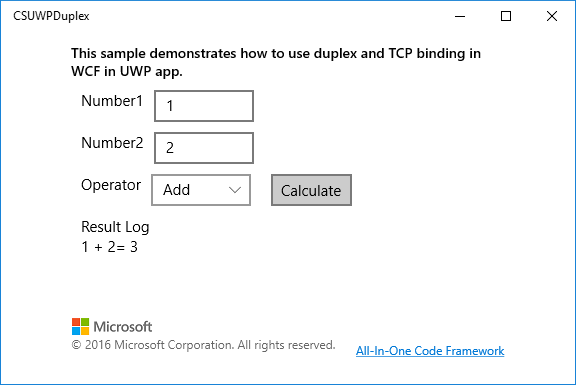
Please make sure that Local Machine which is in the target device menu from the Standard toolbar has been selected. Then do one of the following:

* Click the Start Debugging button on the toolbar.
* Click Start Debugging in the Debug menu.
* Press F5.

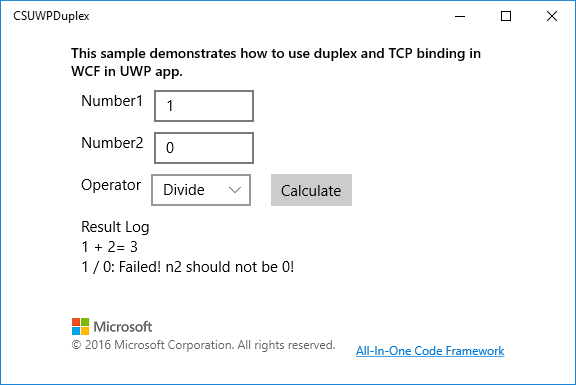
After the service is started, you will see the app window.



Input Number1 and Number2, then click **Calculate** button.



Select another operator and then click **Calculate** button.



**Using the code**

Service Part Calculator class:

[ServiceBehavior(InstanceContextMode = InstanceContextMode.Single, ConcurrencyMode = ConcurrencyMode.Multiple, UseSynchronizationContext = false)]

public class Calculator : ICalculator

{

public ICalculatorCallback CurrentCallback

{

get

{

return OperationContext.Current.

GetCallbackChannel<ICalculatorCallback>();

}

}

public void Add(double n1, double n2)

{

CalculateData data = new CalculateData { Number1 = n1, Number2 = n2, Operator ="+", Result = n1 + n2, Message = string.Empty };

CurrentCallback.ReceiveCalculateResult(data);

}

public void Divide(double n1, double n2)

{

double result = n2 == 0 ? 0 : n1 / n2;

string message = n2 == 0 ? "Failed! n2 should not be 0!" : string.Empty;

CalculateData data = new CalculateData { Number1 = n1, Number2 = n2, Operator = "/", Result = result, Message = message };

CurrentCallback.ReceiveCalculateResult(data);

}

public void Multiply(double n1, double n2)

{

CalculateData data = new CalculateData { Number1 = n1, Number2 = n2, Operator ="x", Result = n1 \* n2, Message = string.Empty };

CurrentCallback.ReceiveCalculateResult(data);

}

public void Subtract(double n1, double n2)

{

CalculateData data = new CalculateData { Number1 = n1, Number2 = n2, Operator = "-", Result = n1 - n2, Message = string.Empty };

CurrentCallback.ReceiveCalculateResult(data);

}

}

Service part ICalculator interface:

[ServiceContract(CallbackContract = typeof(ICalculatorCallback),

SessionMode = SessionMode.Required)]

public interface ICalculator

{

[OperationContract]

void Add(double n1, double n2);

[OperationContract]

void Subtract(double n1, double n2);

[OperationContract]

void Multiply(double n1, double n2);

[OperationContract]

void Divide(double n1, double n2);

}

Service part ICalculatorCallback interface:

public interface ICalculatorCallback

{

[OperationContract(IsOneWay = true)]

void ReceiveCalculateResult(CalculateData data);

}

Service part DataContract:

public class CalculateData

{

[DataMember]

public double Number1 { get; set; }

[DataMember]

public double Number2 { get; set; }

[DataMember]

public double Result { get; set; }

[DataMember]

public string Operator { get; set; }

[DataMember]

public string Message { get; set; }

}

UWP part Page\_Loaded:

private async void Page\_Loaded(object sender, RoutedEventArgs e)

{

//Add items to the operator list.

this.cbxOperators.Items.Add("Add");

this.cbxOperators.Items.Add("Subtract");

this.cbxOperators.Items.Add("Multiply");

this.cbxOperators.Items.Add("Divide");

this.cbxOperators.SelectedIndex = 0;

//set binding

NetTcpBinding tcpBinding = new NetTcpBinding();

tcpBinding.Security.Mode = SecurityMode.None;

tcpBinding.Security.Transport.ClientCredentialType = TcpClientCredentialType.None;

tcpBinding.Security.Message.ClientCredentialType = MessageCredentialType.None;

//new client

MyCalculatorClient = new CalculatorClient(tcpBinding, new EndpointAddress("net.tcp://localhost:7998/CalculatorService/Calculator/tcp"));

//bind call back event

MyCalculatorClient.ReceiveCalculateResultReceived += MyCalculatorClient\_ReceiveCalculateResultReceived;

await MyCalculatorClient.OpenAsync();

}

UWP part call back event:

private async void MyCalculatorClient\_ReceiveCalculateResultReceived(object sender, ReceiveCalculateResultReceivedEventArgs e)

{

//Schedules the provided callback on the UI thread.

await this.Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.Normal, () =>

{

if (string.IsNullOrEmpty(e.data.Message))

{

this.tbResult.Text += $"{e.data.Number1} {e.data.Operator} {e.data.Number2}= {e.data.Result}\n";

}

else

{

this.tbResult.Text += $"{e.data.Number1} {e.data.Operator} {e.data.Number2}: {e.data.Message}\n";

}

});

}

## More information

[Duplex services](https://msdn.microsoft.com/en-us/library/ms731064(v=vs.110).aspx)

[netTcpBinding](https://msdn.microsoft.com/en-us/library/ms731343(v=vs.110).aspx)