
Create and Use Remote Services

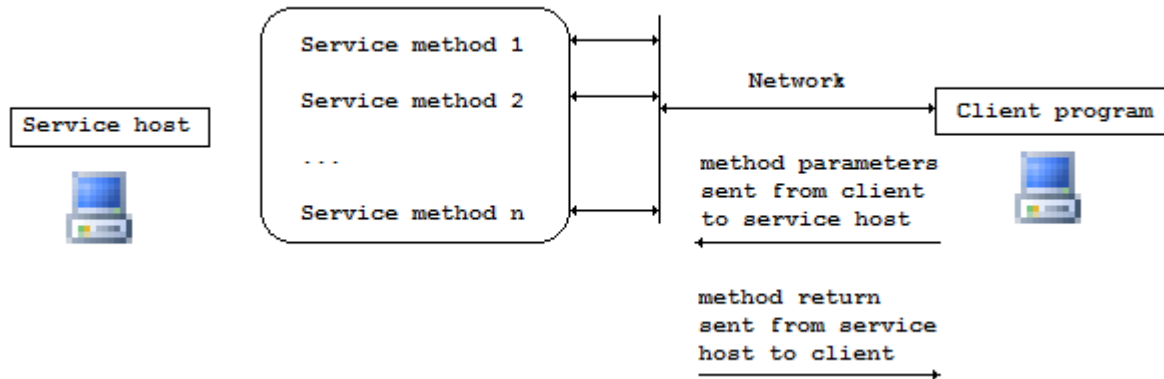
Contents

Introduction	1
Create a Service Host Console	2
Remote Service Console Application.....	2
Remote Service Host	4
Service Console	5
Generate Service Application.....	6
Create Client Applications	7
Create a Windows Form Application.....	7
Add Service Proxy	8
Use the service proxy.....	11
Test	14
Specify communication type at design time	15
Specify communication type at runtime	16
Configurations.....	18
Configuration files	18
Specify service bindings and service ports.....	18
Set Address for Client Program	18
Security Settings	21
Service host settings	21
Client program settings.....	22

Introduction

Networked computers may send messages and commands to each other. A program running in one computer is waiting for other computers to send messages and commands to it. The program is called a server program or a service host program. The service host program exposes methods to be executed by other computers. We may call such methods service methods. A program running in another computer

may execute service methods; we call such a program a client program and the computer running the client program a client computer.



Your task is to design the service methods and client programs. Methods parameters are data/messages sent from client computers. Method return value is the data/message sent from the service host computer to client computers.

For information on how to create methods, see <http://www.limnor.com/support/Limnor%20Studio%20-%20User%20Guide%20-%20Part%20IV.pdf> and <http://www.limnor.com/support/Limnor%20Studio%20-%20User%20Guide%20-%20Part%20V.pdf>

Windows Communication Foundation (WCF) makes it easy to create remote service host programs and client programs. Limnor Studio further simplifies the programming so that you may focus on implementing your business logics. This document shows how to create service host program and client program in Limnor Studio.

Create a Service Host Console

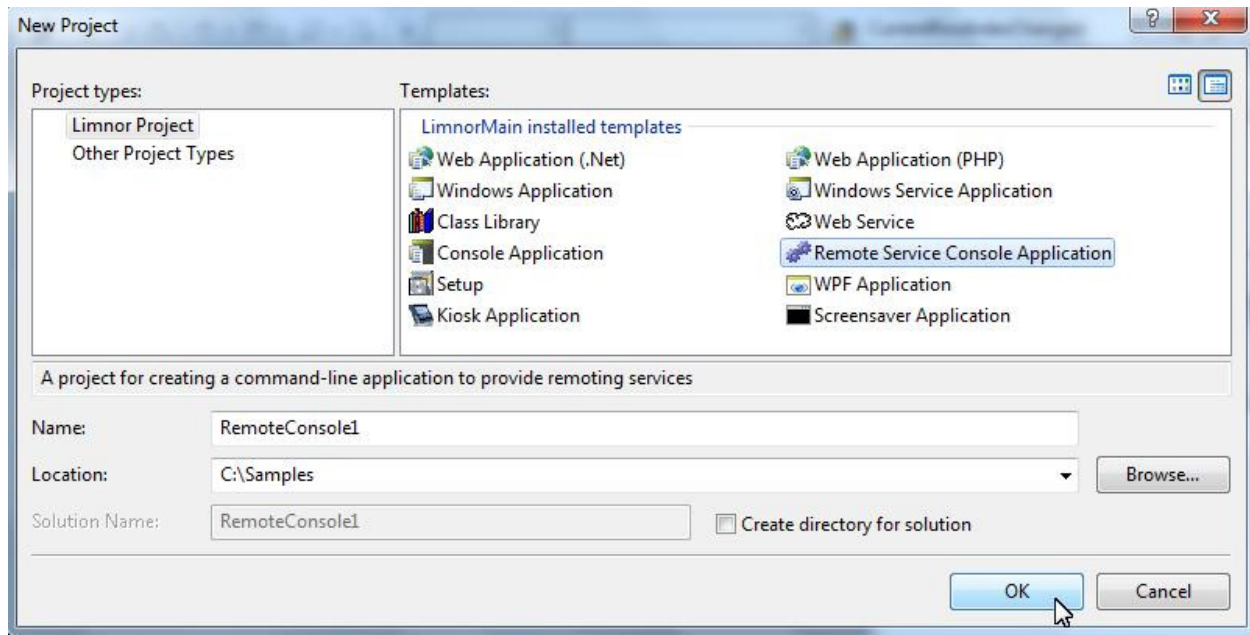
If we do not need a graphic user interface for our service host application then we may simply create a remote-service-console application.

Remote Service Console Application

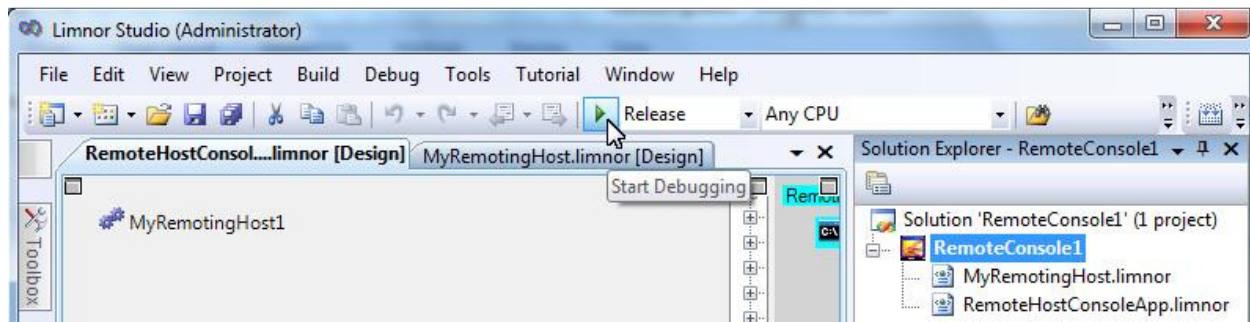
Create a new project:



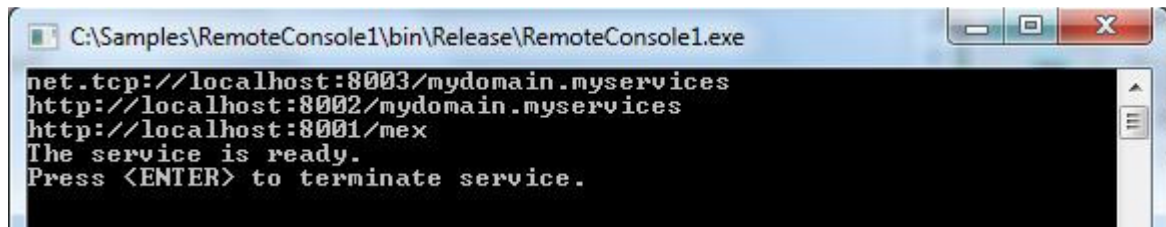
Select “Remote Service Console Application”:



A new console application is created. Click the Run button to try it:



A console window appears:

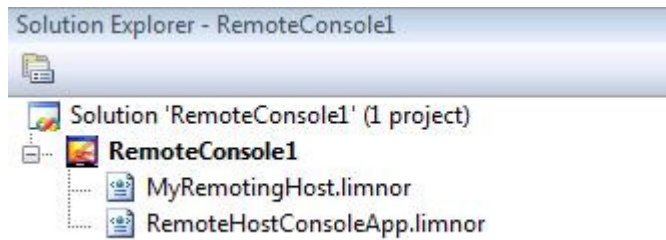


The console hosts a remote service for client programs to call.

Below we explain how this project is formed.

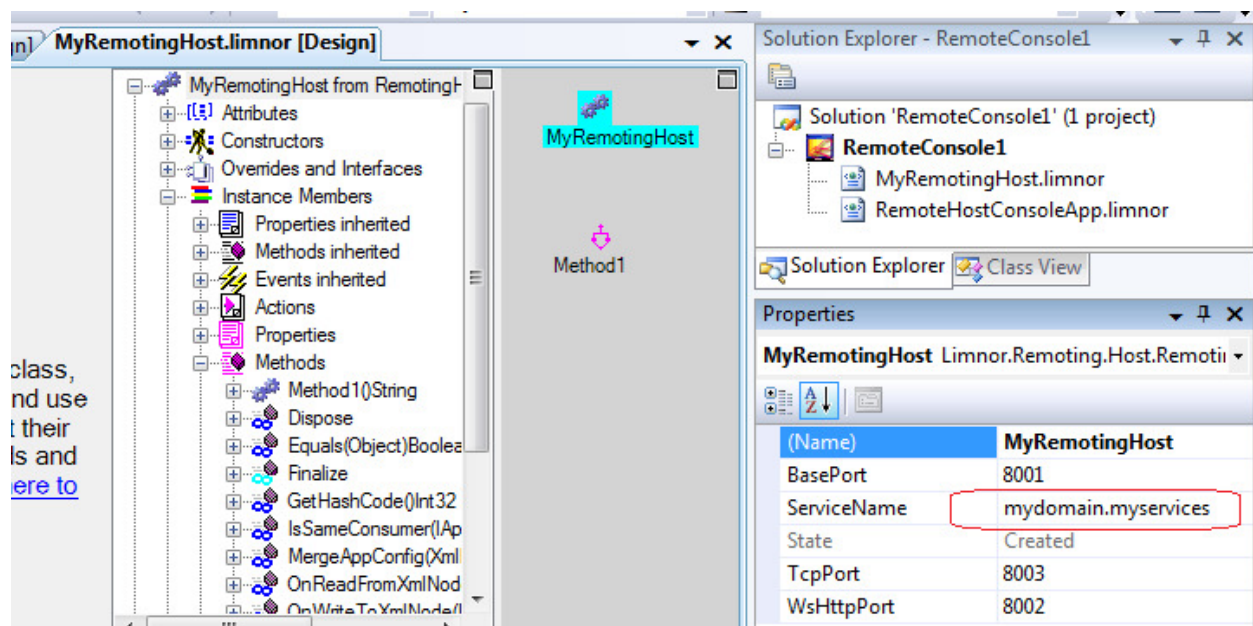
In this console project RemoteHostConsoleApp.limnor is the console application.

MyRemotingHost.limnor is the remote service host:

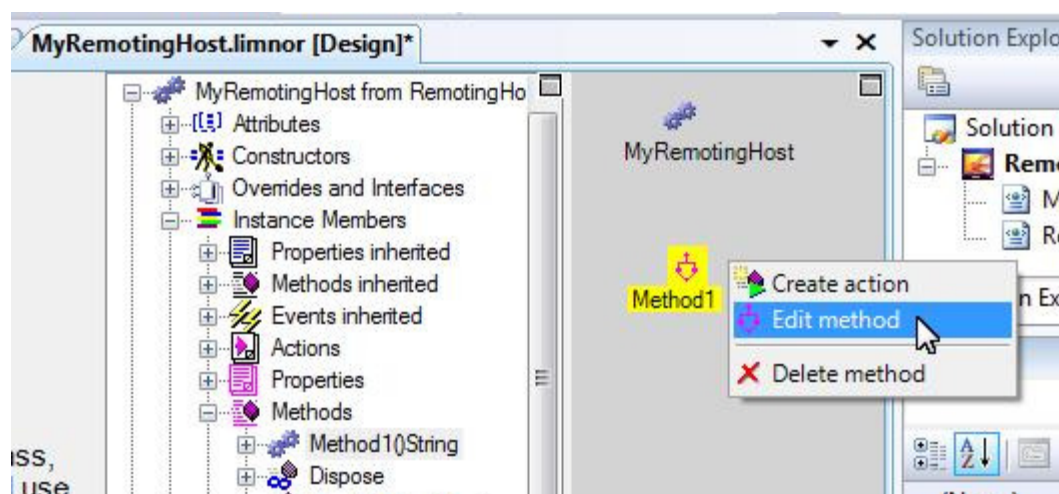


Remote Service Host

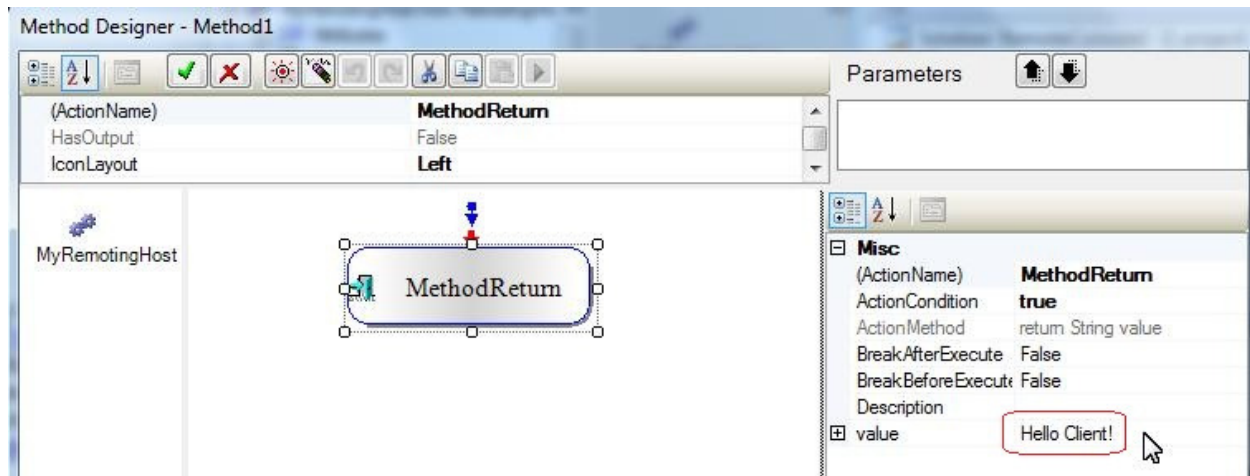
The default service name for the remote service host is myDomain.myservices. You may change it:



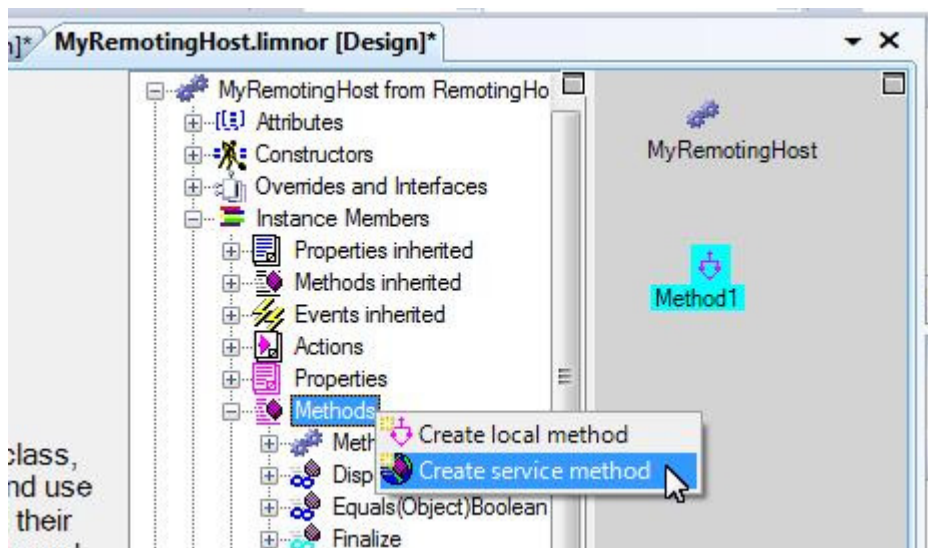
Note the method Method1. This is a sample service method. To edit or review this method, right-click it and choose "Edit method":



You can see that this method only has one action which makes the method return a text “Hello Client!”:



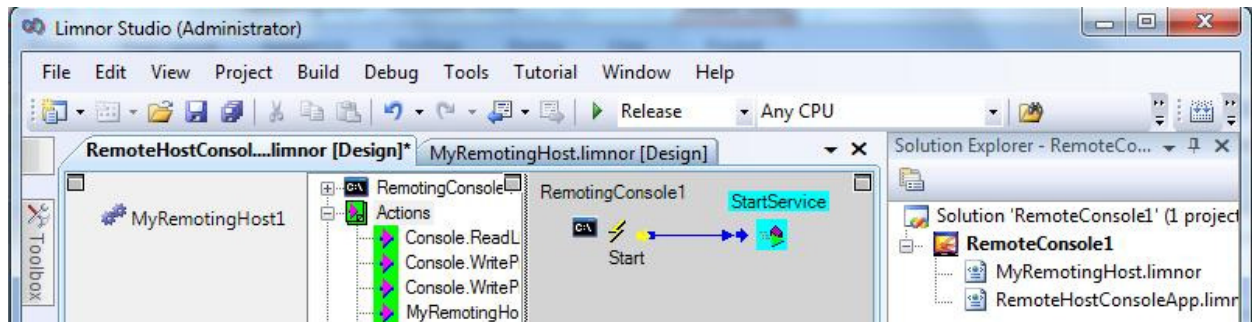
You may edit this method or add new service methods to the service host:



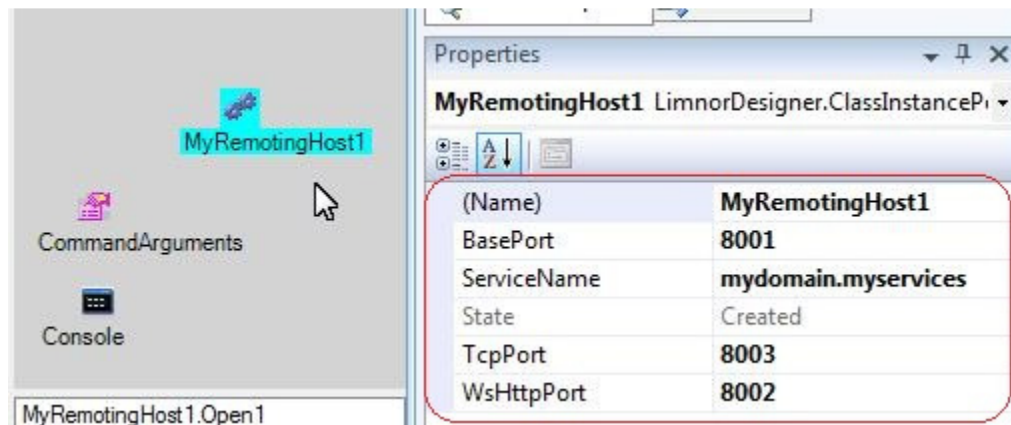
Creating service methods to implement desired business logics is the major task of developing a service host program.

Service Console Application

This sample console application contains an instance of the class `MyRemotingHost`, which is a remote service host. The console application executes an `Open` action of the remote service host to start waiting for client calls; shows ports used and shows some prompts in the console; waits for the user to press <Enter> to quit; and then executes a `Close` action of the remote service host. All these actions are in an event handler method named `StartService`:



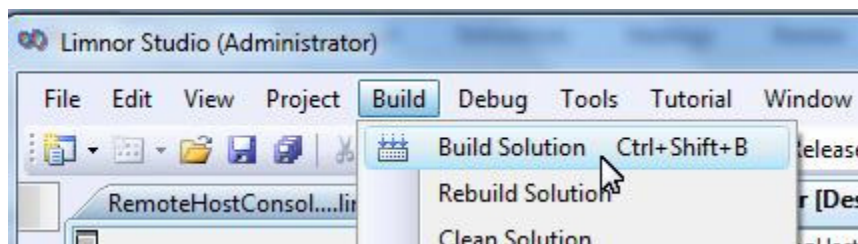
You may modify the properties of the remote service object:



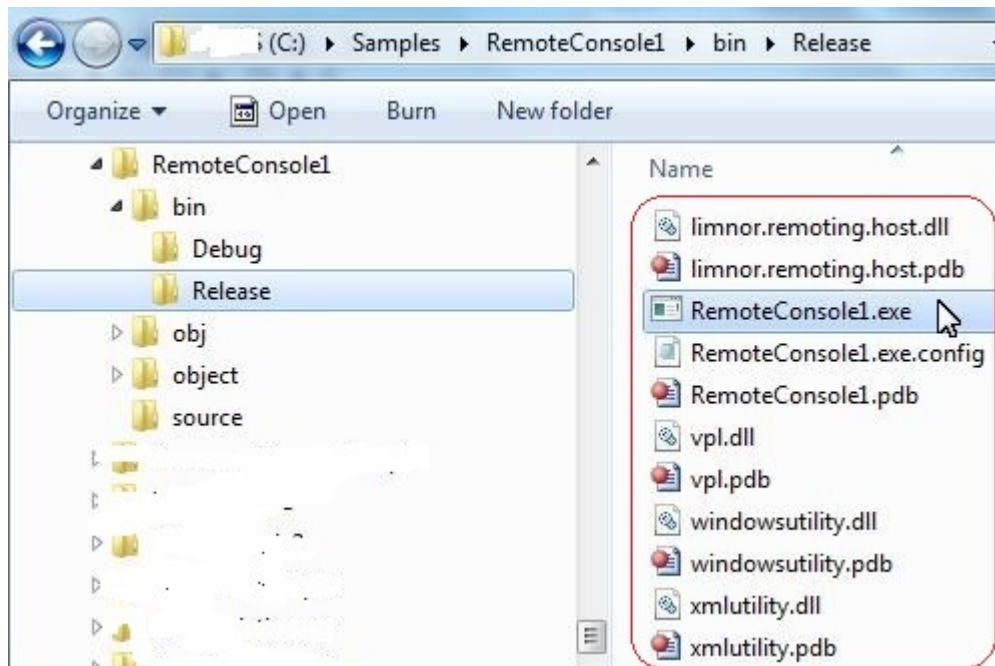
By default the service object uses two ports. One port is for TCP-HTTP binding; another port is for WS-HTTP binding. Such settings can be modified via configuration file for the service host: new ports can be added; default ports can be removed. If you sell your service host program to your customers then your customers will do the configurations according to their computer environments. For more information on bindings, see [http://msdn.microsoft.com/en-us/library/ms730879\(v=VS.90\).aspx](http://msdn.microsoft.com/en-us/library/ms730879(v=VS.90).aspx)

Generate Service Application

Compile the project:

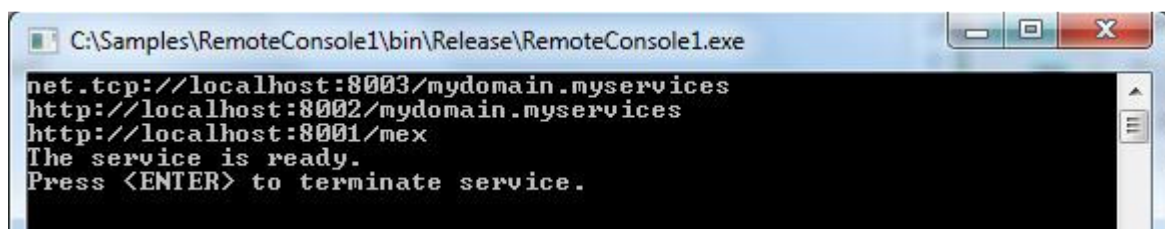


The service application, RemoteCpnsole1.exe, is created.



The *.pdb files are for debugging purpose. You do not need to distribute *.pdb files.

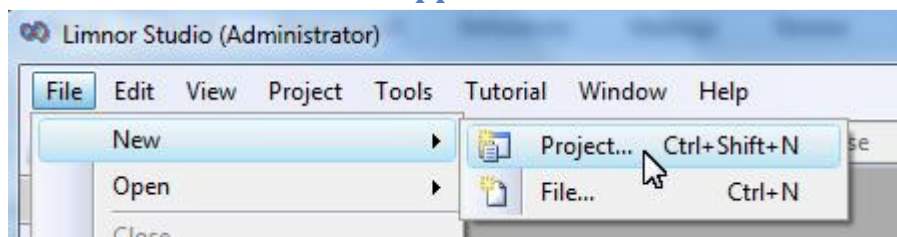
Double-click RemoteConsole1.exe to start the service application. The console starts and waits for the user to close it. While it is waiting, the client computers may execute the service method, Method1, provided by this service application.

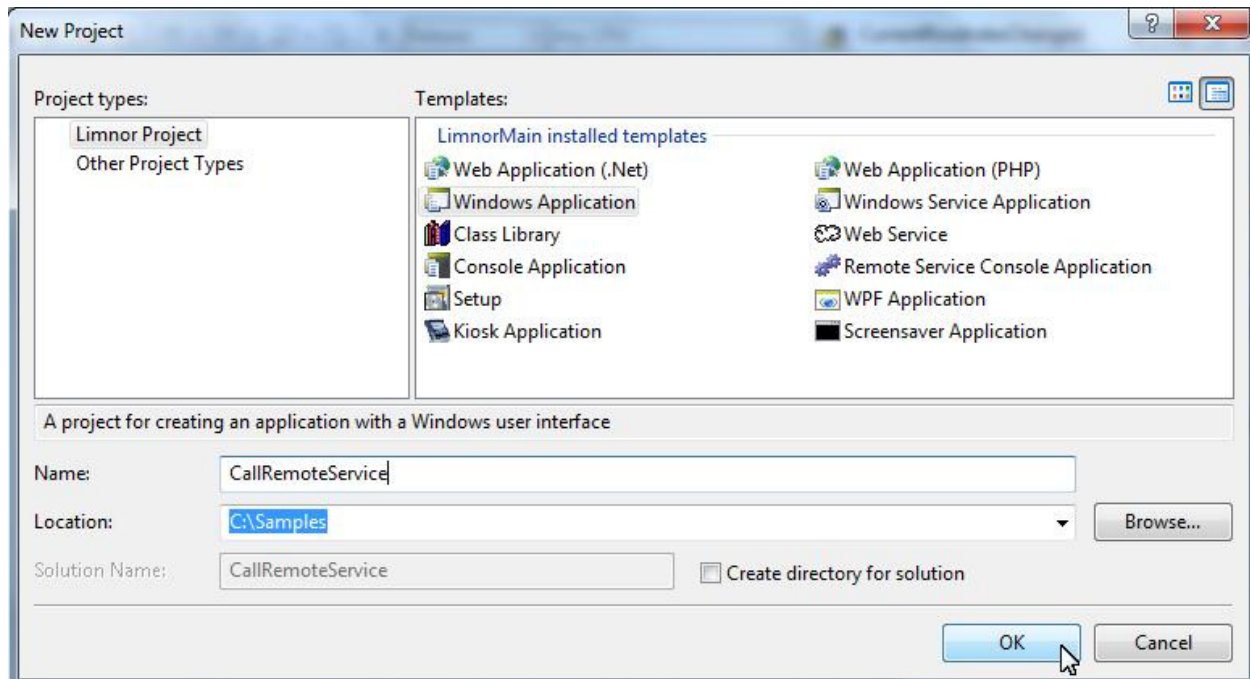


Create Client Applications

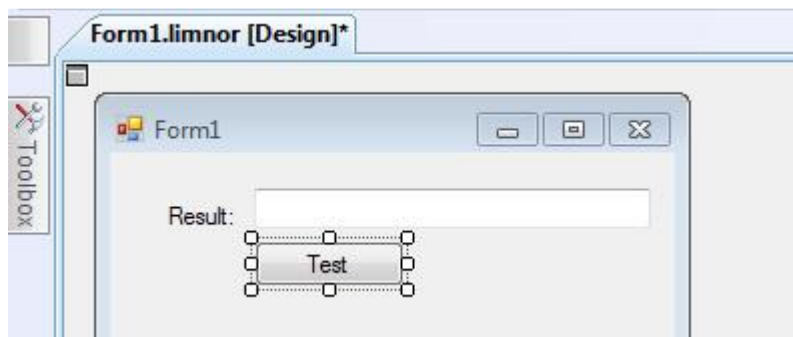
We may now create a client application to execute the service methods. We'll use a Windows Form application as an example.

Create a Windows Form Application





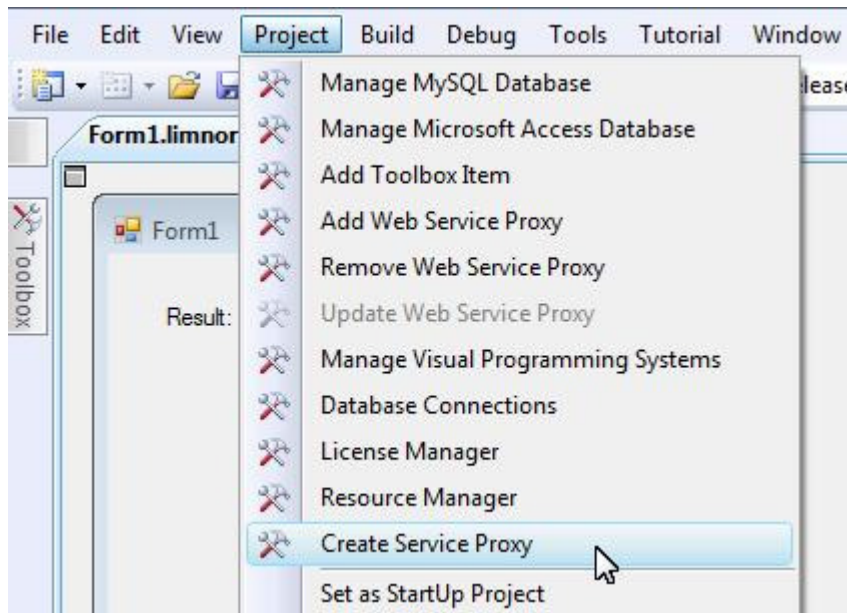
We use a text box to show the return value of the service method; use a button to execute the service method:



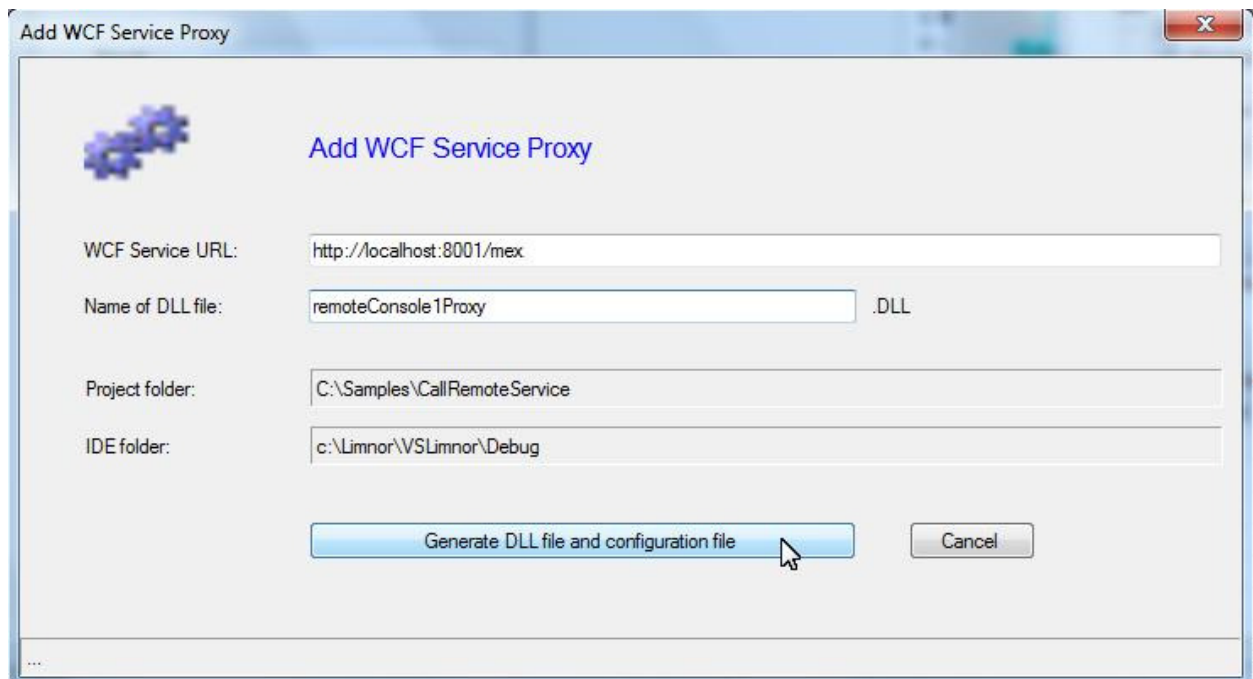
Add Service Proxy

The client application must detect the service methods a service host provides.

Select "Project" menu; choose "Create Service Proxy":

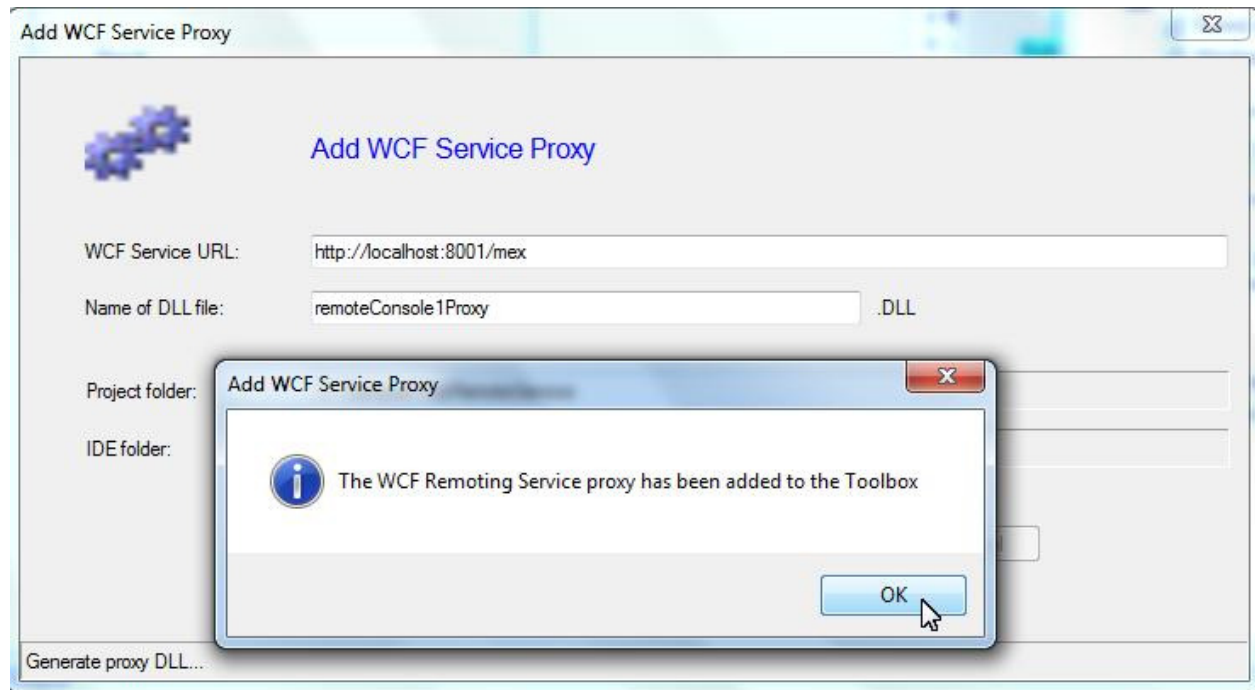


Set the WCF Service URL to the value used by the service host console application; give a DLL file name

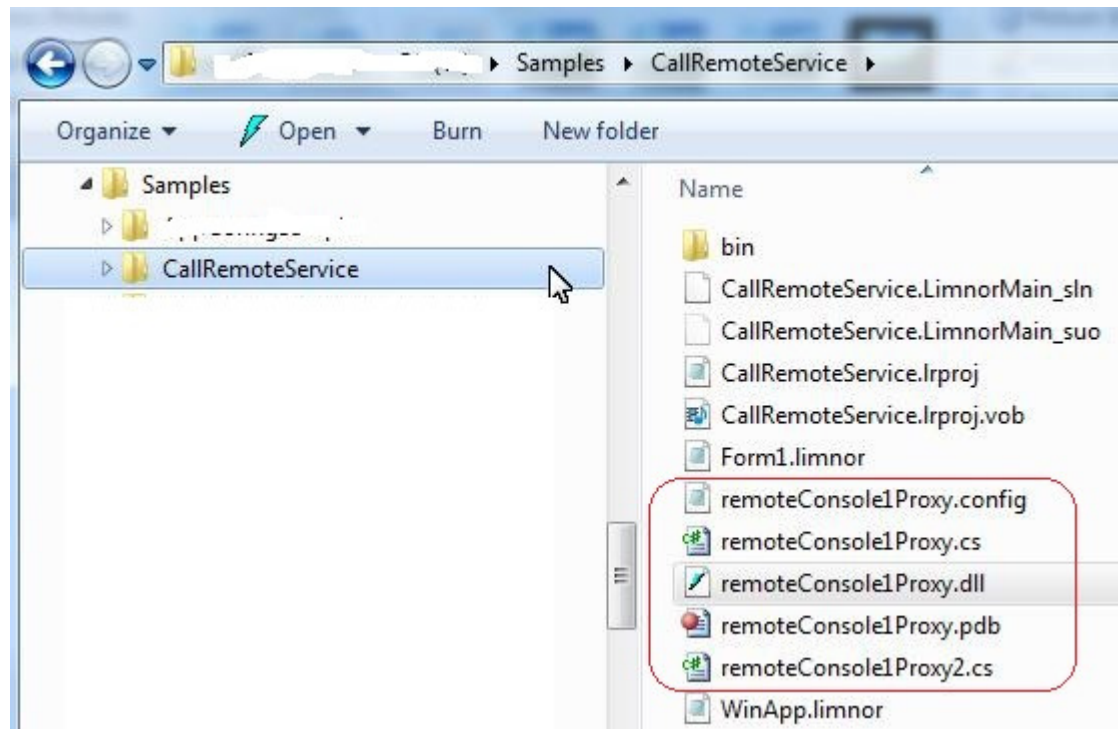


Click button “Generate DLL file and configuration file” to detect the service methods and create the service proxy. Before clicking the button, **make sure that the service console application is running.**

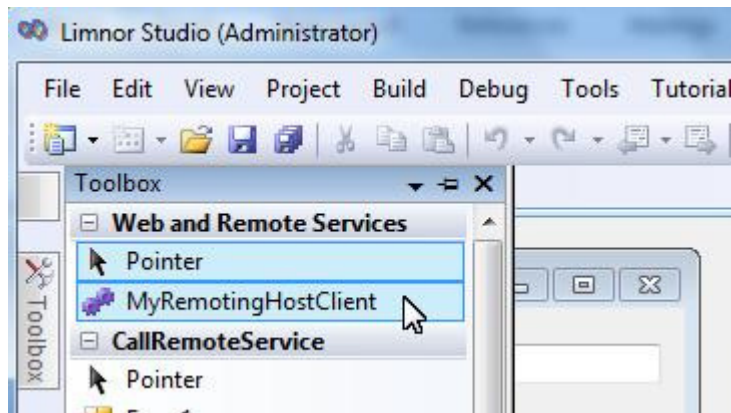
Once it is done, a message box appears:



The service proxy is generated:

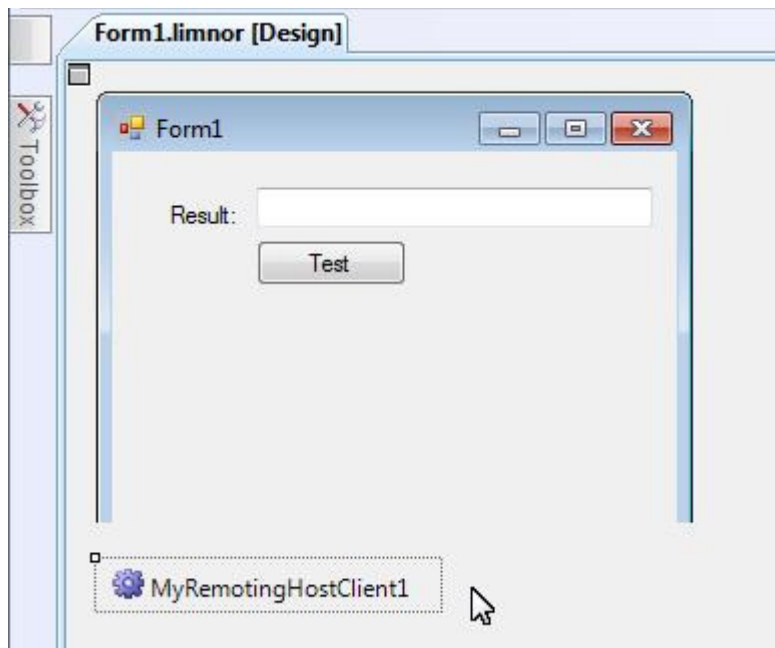


We can see it in the Toolbox:



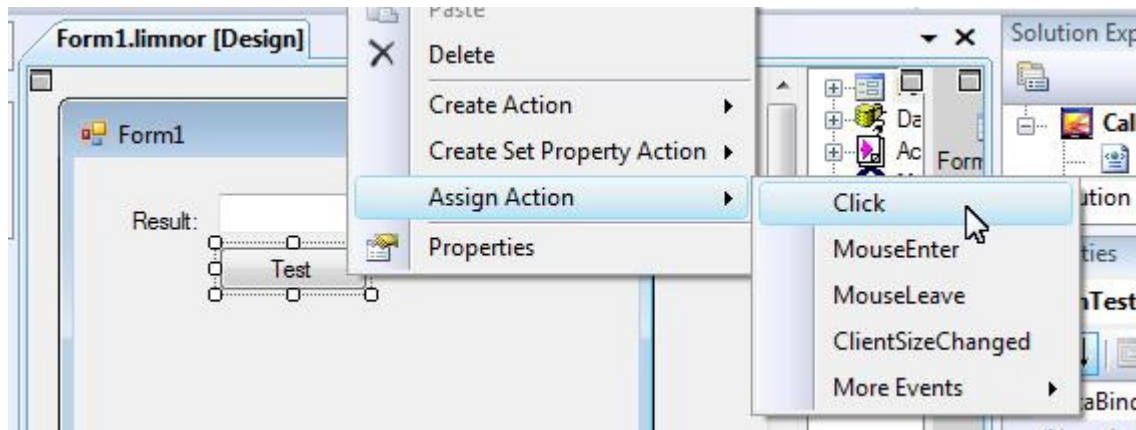
Use the service proxy

We may drop the proxy from the Toolbox to the form:

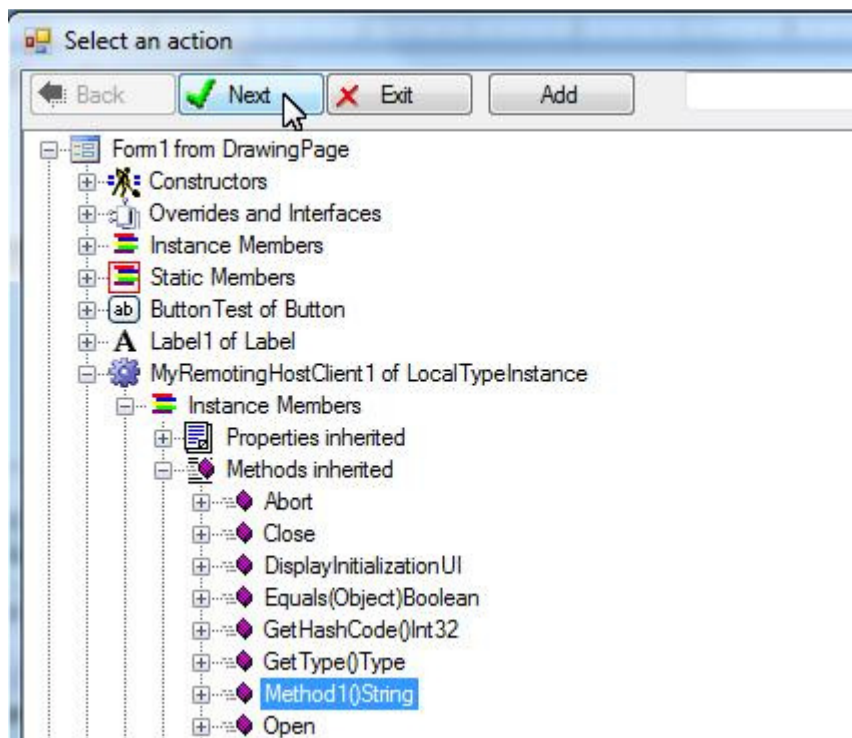


In this sample, we want to click the Test button to execute service method, Method1.

Right-click the button; choose "Assign Action"; choose "Click" event:

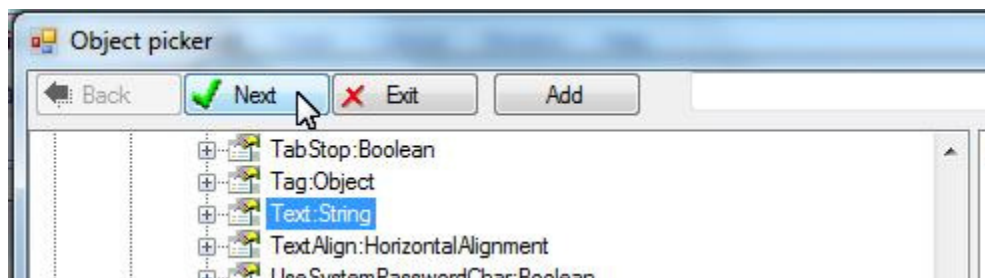
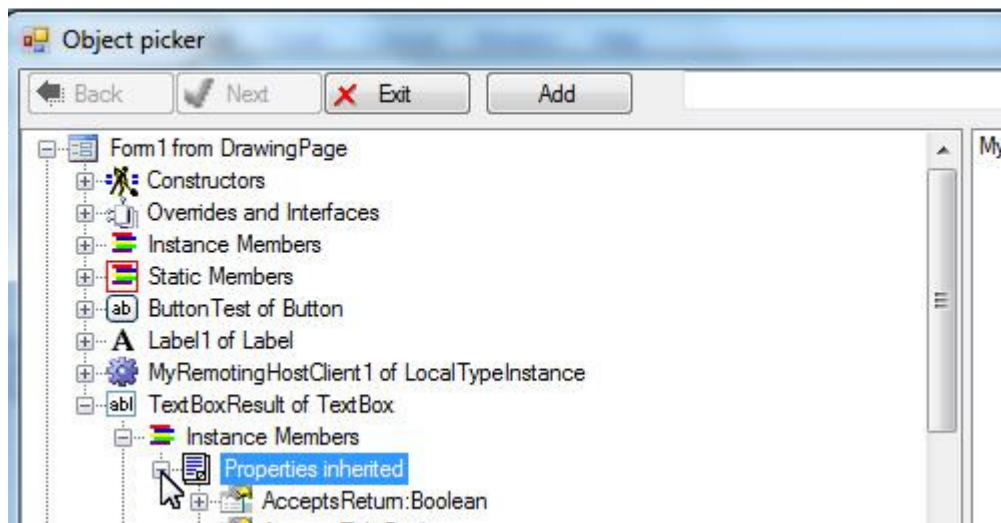
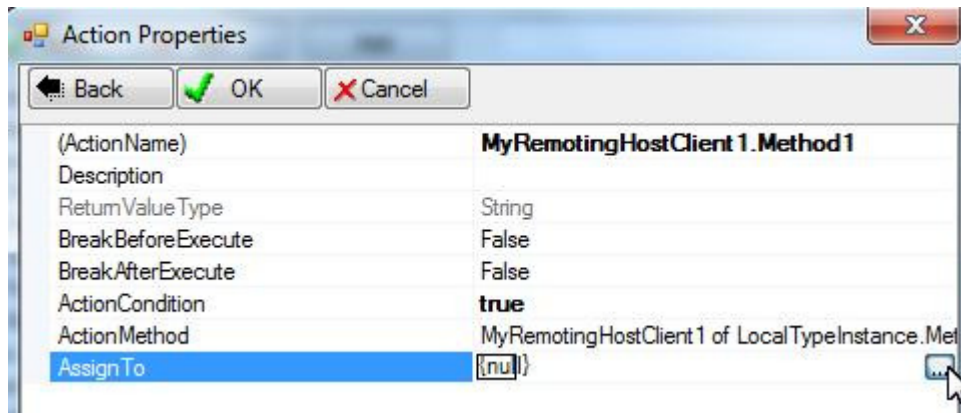


Select the service method, Method1, click “Next”:

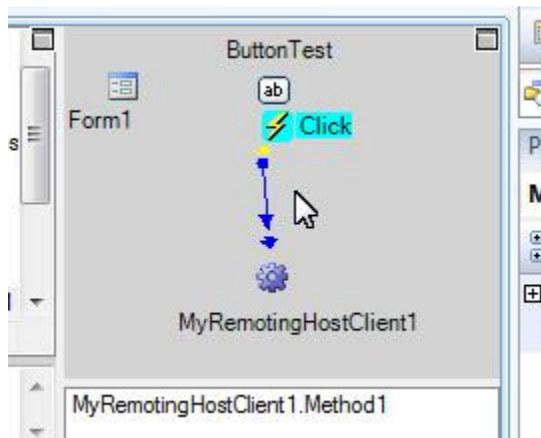
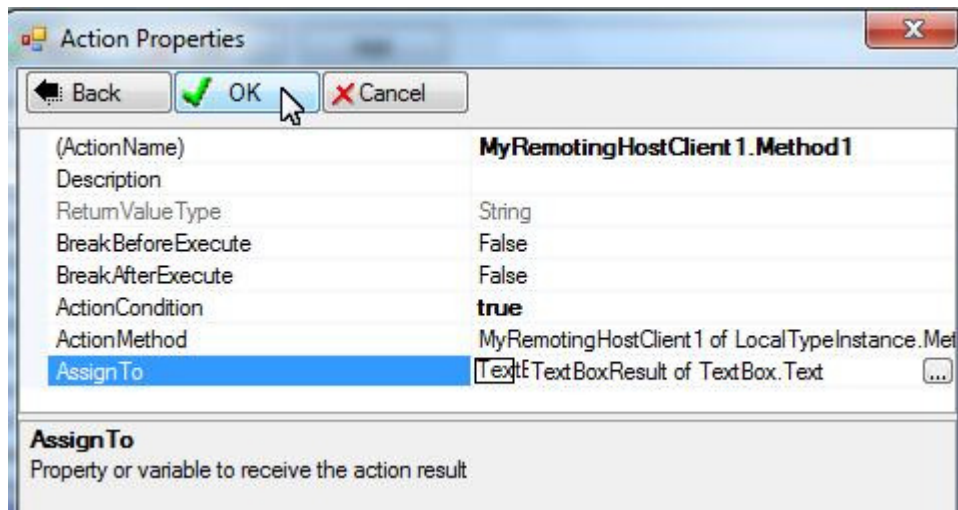


This method does not use parameters.

The action will return method result. We set its AssignTo to the Text property of the text box to display it in the text box:

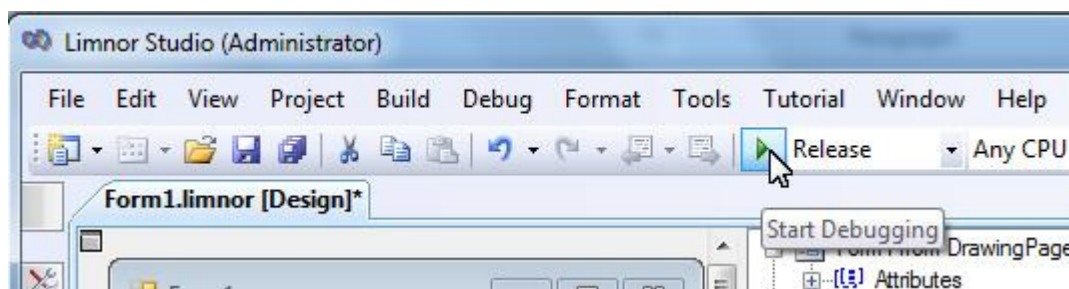


Click OK to finish creating the action and assigning it to the button:

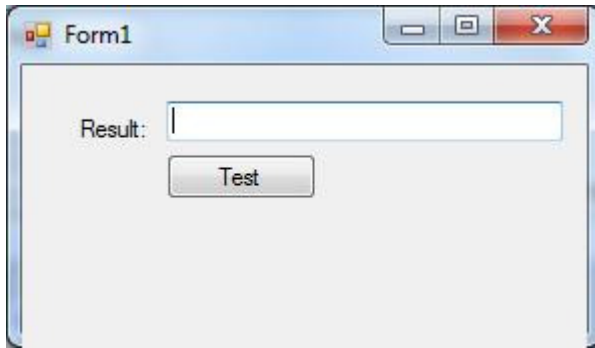


Test

We may test this client application now.



The form appears

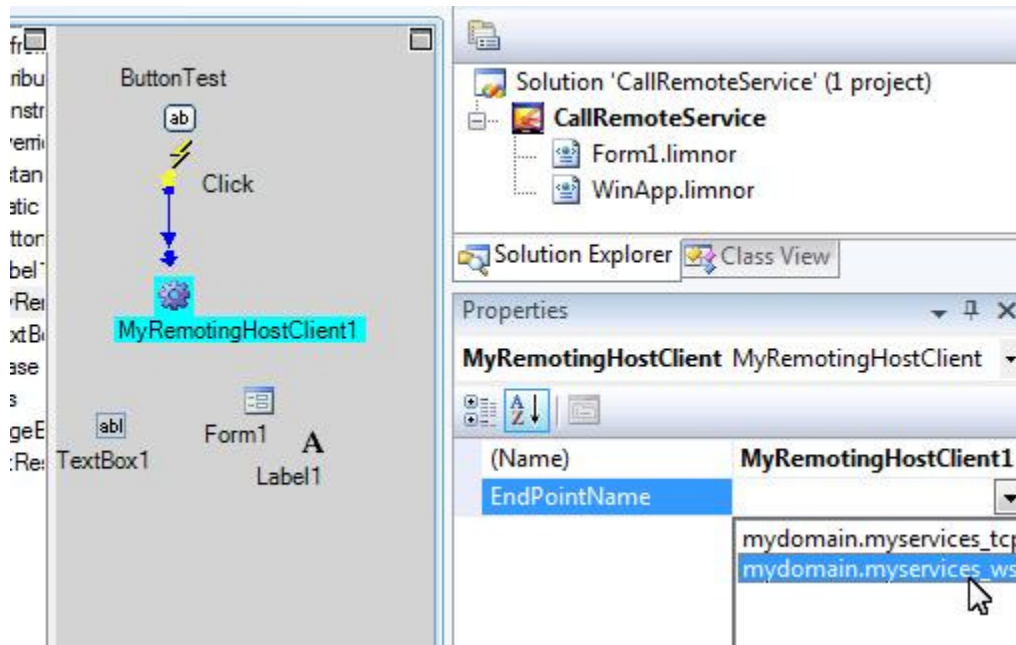


Click the Test button. The service method result is displayed in the Text box.



Specify communication type at design time

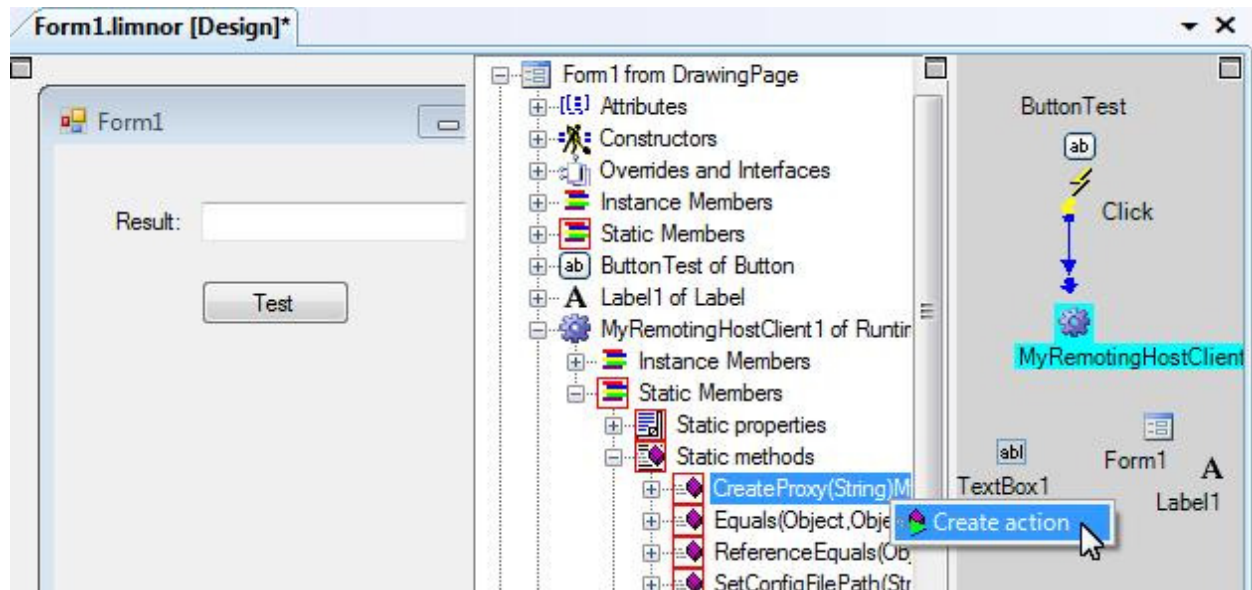
In this sample, the remote service provides two types of communication: TcpHttpBinding and WsHttpBinding. If we do not tell the system the binding to be used then the compiler will pick one. We may specify which binding to use by setting the EndPointName property of the proxy object:



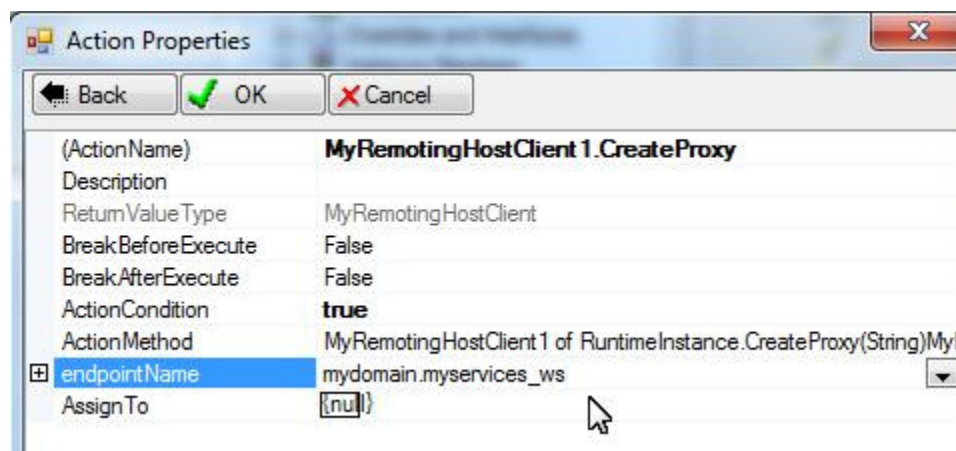
Specify communication type at runtime

The binding can also be changed at runtime by creating CreateProxy actions. Below is an example of creating such an action.

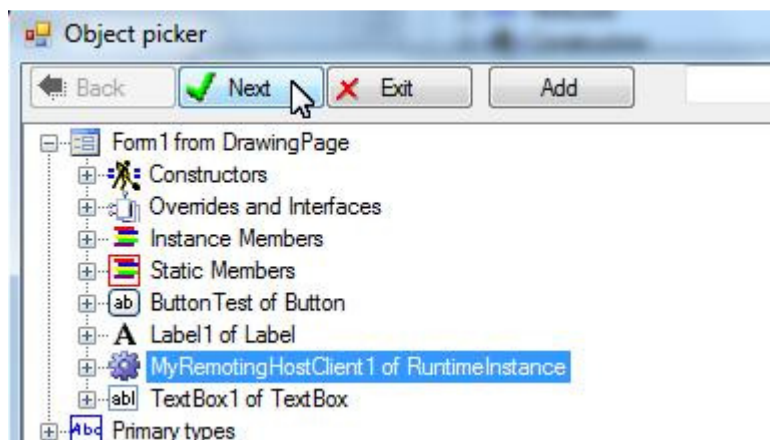
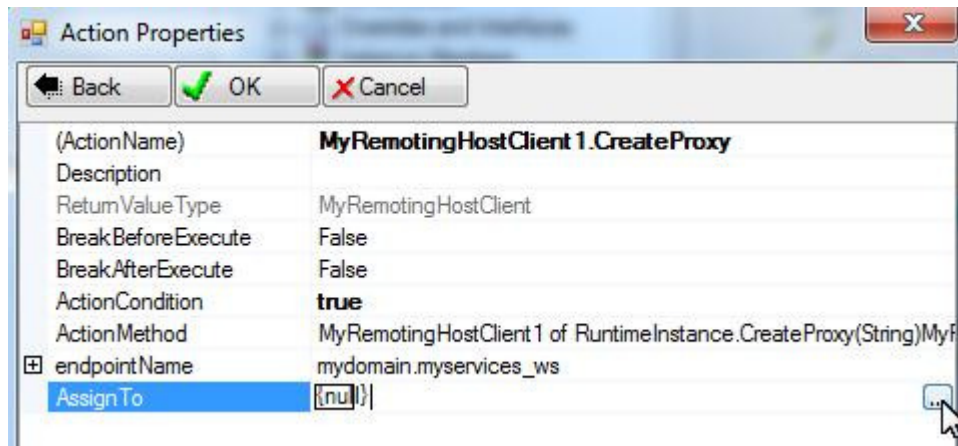
Right-click CreateProxy, choose “Create action”:



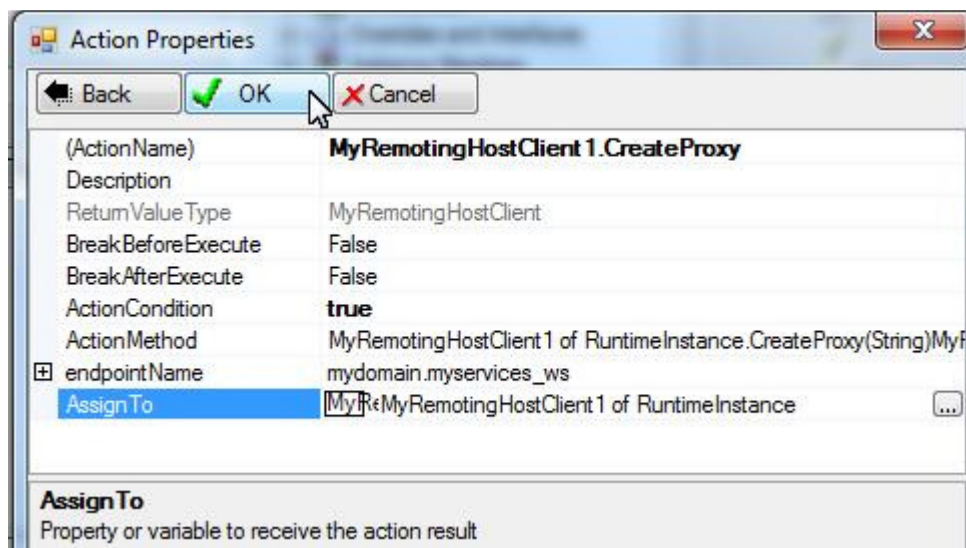
Specify a desired binding name for the “endpointName” parameter:



Set “AssignTo” to the proxy object in the form:



This is the action to set binding at runtime.



Execute the above action will specify the desired binding.

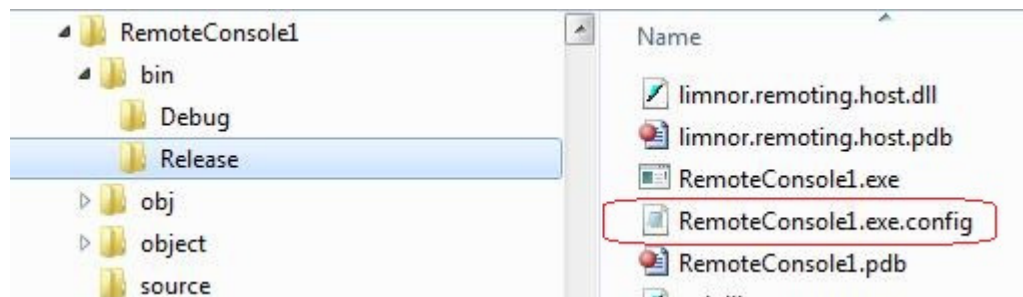
Configurations

Configuration files

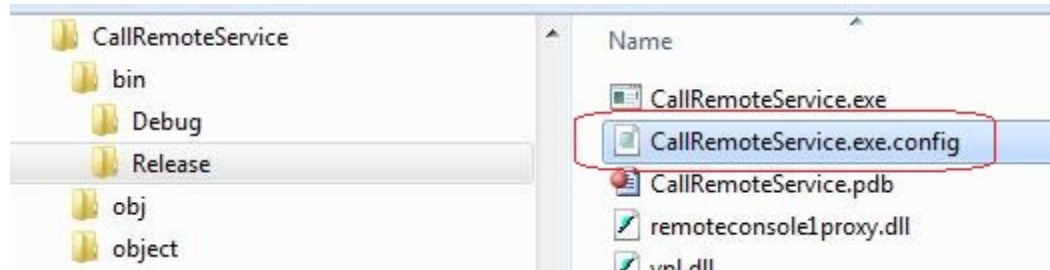
Both the service host program and the client program can be adjusted via configuration files.

A configuration file is in the same folder as where the executable file is. The configuration file name is formed by the executable file name followed by “.config”.

For example, our service host sample program is RemoteConsole1.exe; its configuration file is RemoteConsole1.exe.config:



Our sample client program is CallRemoteService.exe; its configuration file is CallRemoteService.exe.config:



Specify service bindings and service ports

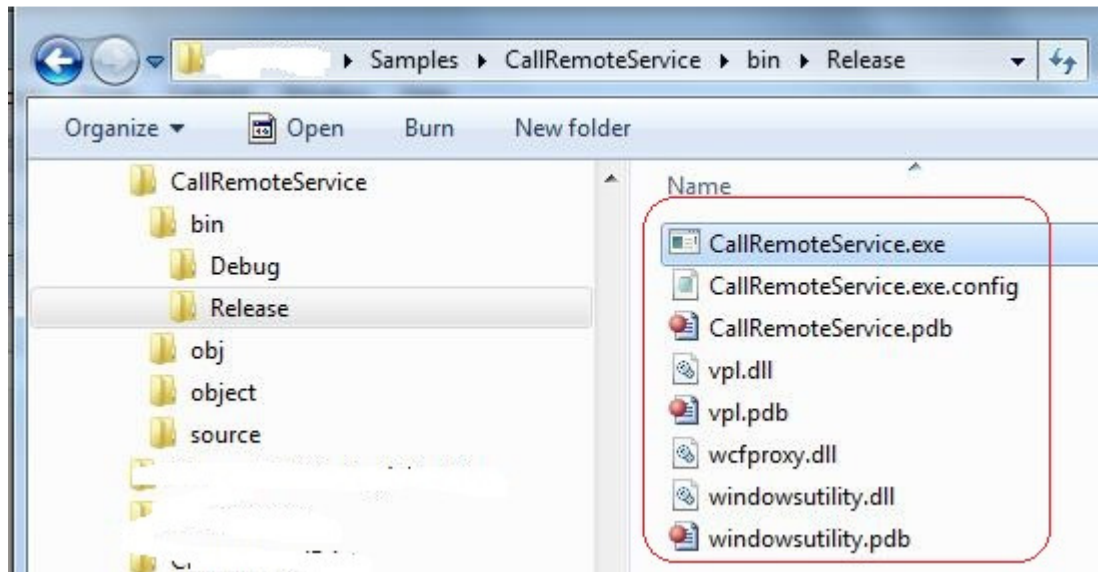
The above service host sample uses two service bindings. Open file RemoteConsole1.exe.config, you can see the <binding> notes and the corresponding <endpoint> nodes. The “bindingConfiguration” attribute of a <endpoint> node specify the “name” attribute of its corresponding <binding> note. Service port is specified in the <endpoint> node.

For bindings provided by the .Net Framework, see [http://msdn.microsoft.com/en-us/library/ms730879\(v=VS.90\).aspx](http://msdn.microsoft.com/en-us/library/ms730879(v=VS.90).aspx)

Set Address for Client Program

In the above test the remote service console application and the client application are running on the same computer. Usually the two applications are running at different computers.

We may copy the client program files to another computer which is networked to the service host computer.

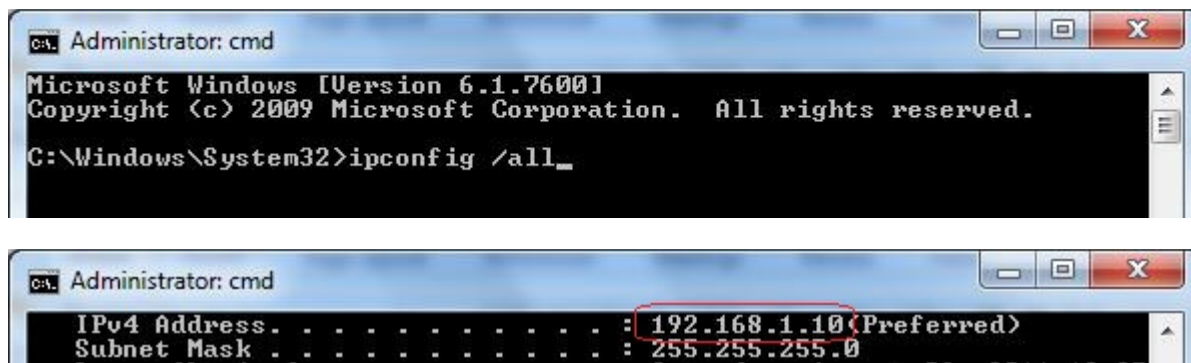


We must tell the client program where the service host computer is. We do it via the configuration file for the client program. In our example it is the CallRemoteService.exe.config.

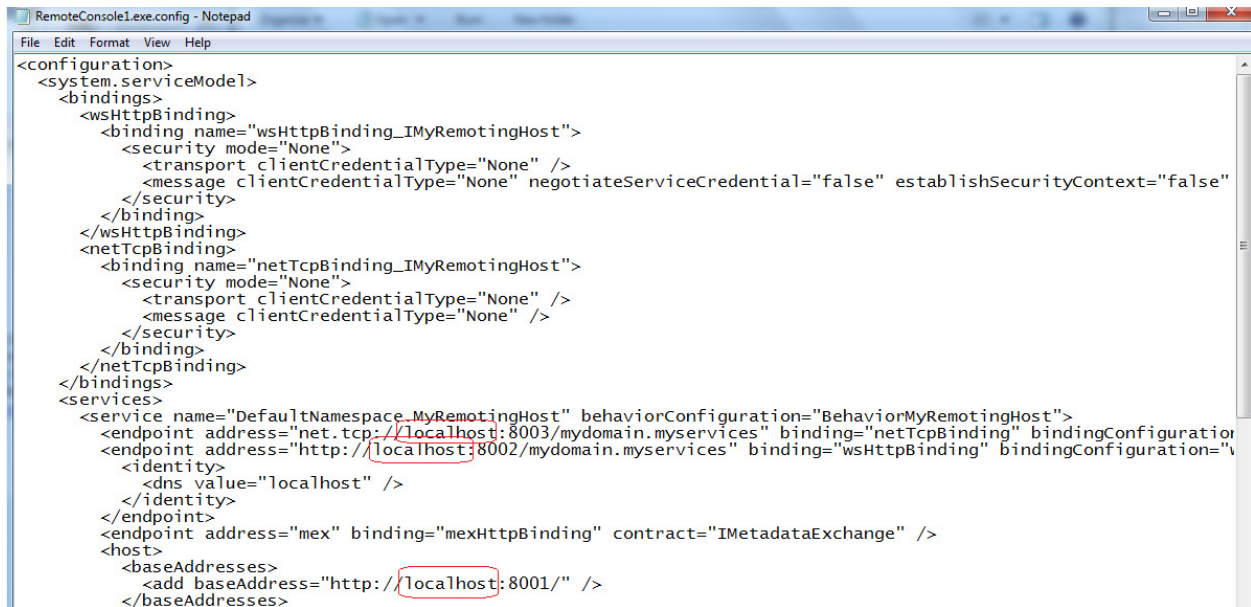
First, we need to find out the IP address of the service host computer. We can use command

`Ipconfig /all`

to find out a computer's ip address:



Now we go to the client computer. Open CallRemoteService.exe.config with an XML editor or the Notepad. Locate the "endpoint" and "address" in the file:

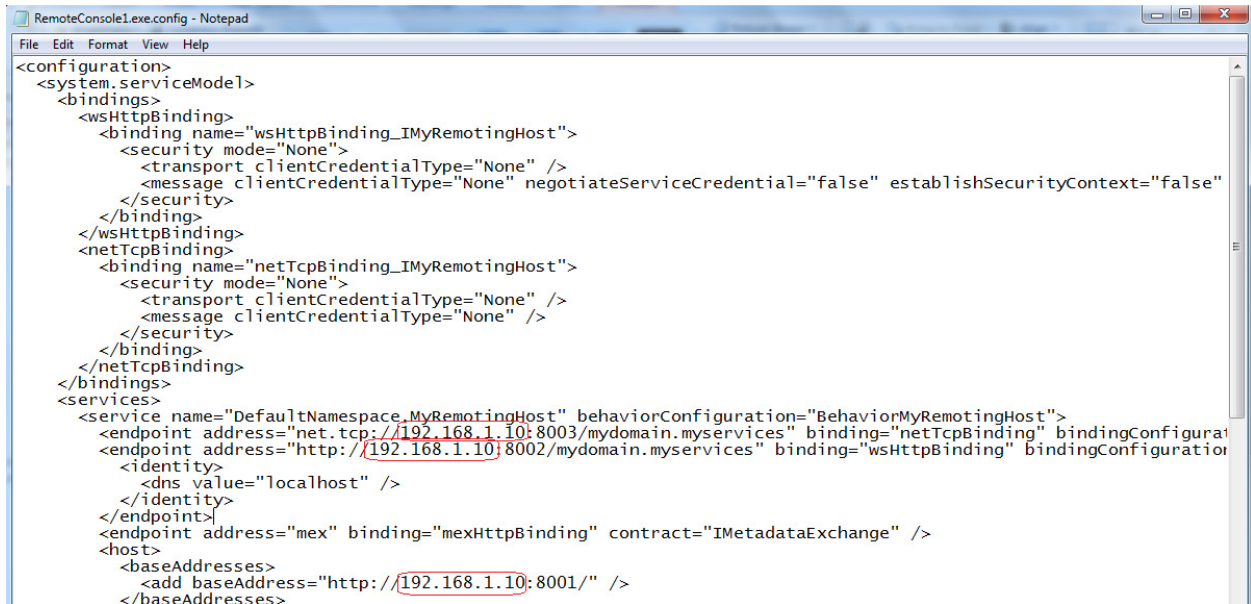


```

<configuration>
  <system.serviceModel>
    <bindings>
      <wsHttpBinding>
        <binding name="wsHttpBinding_IMyRemotingHost">
          <security mode="None">
            <transport clientCredentialType="None" />
            <message clientCredentialType="None" negotiateServiceCredential="false" establishSecurityContext="false" />
          </security>
        </binding>
      </wsHttpBinding>
      <netTcpBinding>
        <binding name="netTcpBinding_IMyRemotingHost">
          <security mode="None">
            <transport clientCredentialType="None" />
            <message clientCredentialType="None" />
          </security>
        </binding>
      </netTcpBinding>
    </bindings>
    <services>
      <service name="DefaultNamespace.MyRemotingHost" behaviorConfiguration="BehaviorMyRemotingHost">
        <endpoint address="net.tcp://localhost:8003/mydomain.myservices" binding="netTcpBinding" bindingConfiguration="netTcpBinding_IMyRemotingHost" />
        <endpoint address="http://localhost:8002/mydomain.myservices" binding="wsHttpBinding" bindingConfiguration="wsHttpBinding_IMyRemotingHost" />
        <identity>
          <dns value="localhost" />
        </identity>
        </endpoint>
        <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
      </service>
    </services>
    <host>
      <baseAddresses>
        <add baseAddress="http://localhost:8001/" />
      </baseAddresses>
    </host>
  </system.serviceModel>
</configuration>

```

Change “localhost” to the IP address of the service host computer:



```

<configuration>
  <system.serviceModel>
    <bindings>
      <wsHttpBinding>
        <binding name="wsHttpBinding_IMyRemotingHost">
          <security mode="None">
            <transport clientCredentialType="None" />
            <message clientCredentialType="None" negotiateServiceCredential="false" establishSecurityContext="false" />
          </security>
        </binding>
      </wsHttpBinding>
      <netTcpBinding>
        <binding name="netTcpBinding_IMyRemotingHost">
          <security mode="None">
            <transport clientCredentialType="None" />
            <message clientCredentialType="None" />
          </security>
        </binding>
      </netTcpBinding>
    </bindings>
    <services>
      <service name="DefaultNamespace.MyRemotingHost" behaviorConfiguration="BehaviorMyRemotingHost">
        <endpoint address="net.tcp://192.168.1.10:8003/mydomain.myservices" binding="netTcpBinding" bindingConfiguration="netTcpBinding_IMyRemotingHost" />
        <endpoint address="http://192.168.1.10:8002/mydomain.myservices" binding="wsHttpBinding" bindingConfiguration="wsHttpBinding_IMyRemotingHost" />
        <identity>
          <dns value="localhost" />
        </identity>
        </endpoint>
        <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
      </service>
    </services>
    <host>
      <baseAddresses>
        <add baseAddress="http://192.168.1.10:8001/" />
      </baseAddresses>
    </host>
  </system.serviceModel>
</configuration>

```

Now we can run CallRemoteService.exe and execute Method1 provided by the service host from computer 192.168.1.10.

If you get a connection error when running the client program then you may try to turn off the firewall on both computers. Once you confirm that turning off the firewalls makes it work you may adjust firewall settings on both computers to allow the communications between the client program and the service host program.

- ✓ In the client computer, add the client program to the allowed program list of the firewall.

- ✓ In the service host computer, add the service host program to the allowed program list of the firewall.
- ✓ In the service host computer, add one inbound rule and one outbound rule to allow the ports the programs are using. See <http://msdn.microsoft.com/en-us/library/ms751530.aspx>

Security Settings

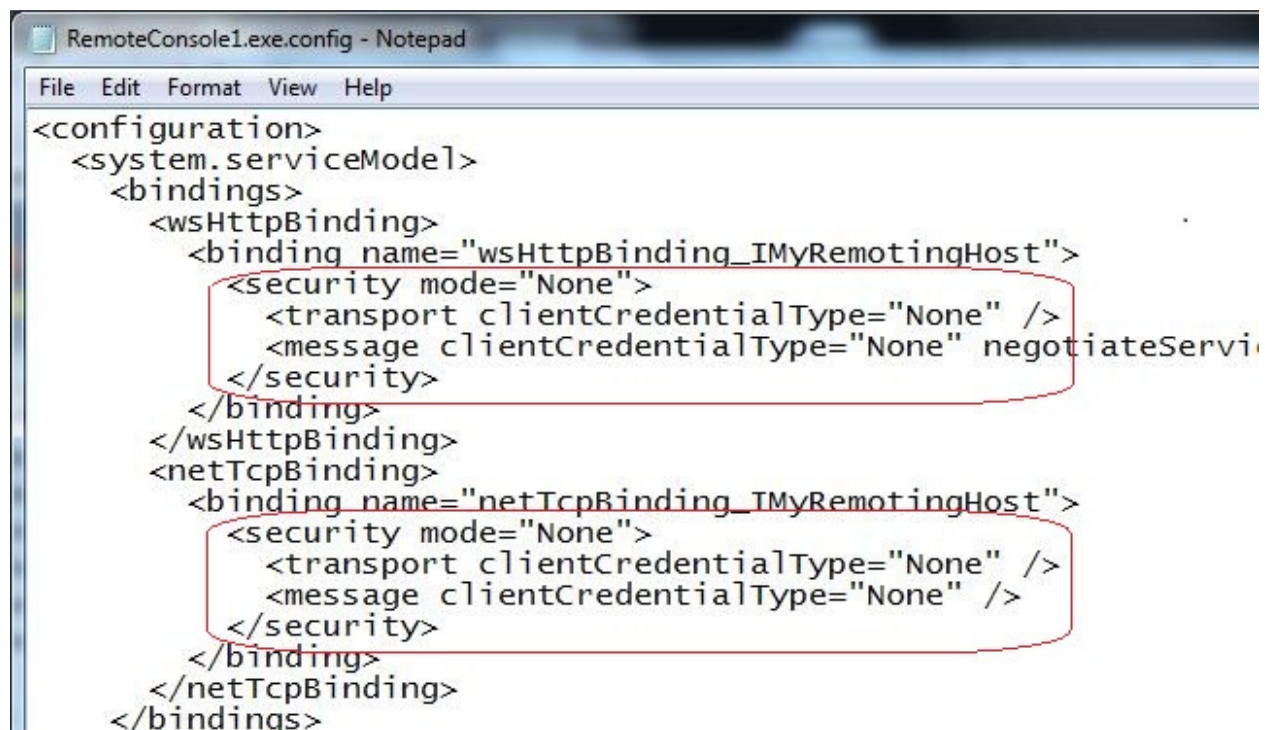
In the above samples the security is turned off. If you want to restrict remote services to authenticated clients only then the configuration files for both service host and client programs have to be modified to specify security settings. The settings have to be appropriate to the types of networks connecting the computers. For more information see <http://msdn.microsoft.com/en-us/library/ff650862.aspx>

<http://msdn.microsoft.com/en-us/library/ff405740.aspx>

Service host settings

The service host may restrict its services to only authenticated clients. Such restriction is set in the service host configuration file. In our example it is the file RemoteConsole1.exe.config

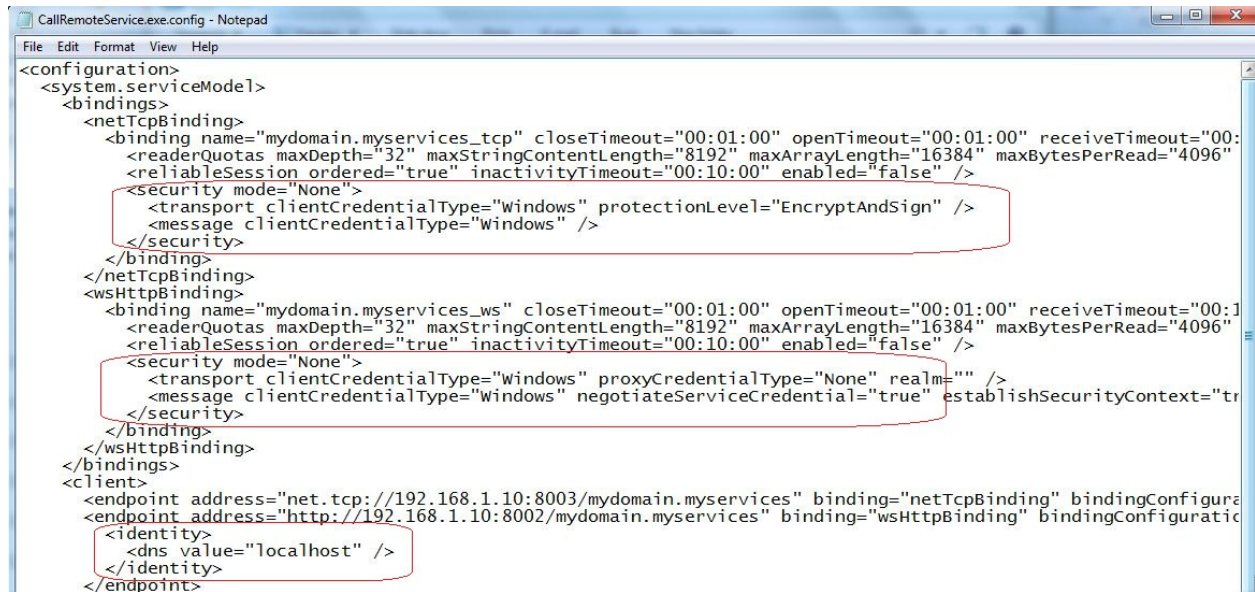
Use an XML editor or the Notepad to open RemoteConsole1.exe.config. Locate the security settings:



For detail information on setting the <security> node, see <http://msdn.microsoft.com/en-us/library/ms731362.aspx>

Client program settings

If the service host restricts the client access then the configuration files for the client programs must be modified accordingly.



For more information, see [http://msdn.microsoft.com/en-us/library/ms731094\(v=VS.90\).aspx](http://msdn.microsoft.com/en-us/library/ms731094(v=VS.90).aspx)