**Part 4 - WCF service implementing multiple service contracts**

**n this video, we will discuss**  
**1.** How a single **WCF service** can implement **multiple service contracts** and then  
**2.** Expose each service contract using a different endpoint  
  
Let's understand this with an example.

**Creating a WCF service that implements multiple contracts**  
**1.** Create a new class library project with name = **CompanyService**  
**2.** Delete **Class1.cs** file that is auto-generated  
**3.** Add a new **WCF** service with name = **CompanyService.** This should generate **CompanyService.cs** and **ICompanyService.cs** files.  
**4.** Copy and paste the following code in **ICompanyService.cs** file  
using System.ServiceModel;  
namespace CompanyService  
{  
    [ServiceContract]  
    public interface IMyCompanyPublicService  
    {  
        [OperationContract]  
        string GetPublicInformation();  
    }  
  
    [ServiceContract]  
    public interface IMyCompanyConfidentialService  
    {  
        [OperationContract]  
        string GetCofidentialInformation();  
    }  
}  
**5.** Copy and paste the following code in **CompanyService.cs** file  
namespace CompanyService  
{  
    public class CompanyService : IMyCompanyPublicService, IMyCompanyConfidentialService  
    {  
        public string GetPublicInformation()  
        {  
            return "This is public information and available over HTTP to all general public outside the FireWall";  
        }  
  
        public string GetCofidentialInformation()  
        {  
            return "This is confidential information and only available over TCP behind the company FireWall";  
        }  
    }  
}

**Hosting the WCF service using a console application.**  
**1.** Right click on **CompanyService** solution in Solution Explorer and add a new Console Application project with name = **CompanyServiceHost**

**2.** Add a reference to **System.ServiceModel** assembly and **CompanyService** project  
**3.** Right click on **CompanyServiceHost** project and add **Application Configuration File.** This should add **App.config** file to the project. Copy and paste the following XML.   
<?xml version="1.0" encoding="utf-8" ?>  
<configuration>  
  <system.serviceModel>  
    <services>  
      <service name="CompanyService.CompanyService" behaviorConfiguration="mexBehaviour">  
        <endpoint address="CompanyService" binding="basicHttpBinding" contract="CompanyService.IMyCompanyPublicService"></endpoint>  
        <endpoint address="CompanyService" binding="netTcpBinding" contract="CompanyService.IMyCompanyConfidentialService"></endpoint>  
        <host>  
          <baseAddresses>  
            <add baseAddress="http://localhost:8080/"/>  
            <add baseAddress="net.tcp://localhost:8090/"/>  
          </baseAddresses>  
        </host>  
      </service>  
    </services>  
    <behaviors>  
      <serviceBehaviors>  
        <behavior name="mexBehaviour">  
          <serviceMetadata httpGetEnabled="true"/>  
        </behavior>  
      </serviceBehaviors>  
    </behaviors>  
  </system.serviceModel>  
</configuration>  
**4.** Copy and paste the following code in **Program.cs** file  
using System;  
namespace CompanyServiceHost  
{  
    class Program  
    {  
        static void Main()  
        {  
            using(System.ServiceModel.ServiceHost host = new   
                System.ServiceModel.ServiceHost(typeof(CompanyService.CompanyService)))  
            {  
                host.Open();  
                Console.WriteLine("Host started @ " + DateTime.Now.ToString());  
                Console.ReadLine();  
            }  
        }  
    }  
}  
  
Build the solution. Set **CompanyServiceHost** as startup project and run it by pressing CTRL + F5 keys.  
  
**Now lte's build a web application that is going to consume the WCF service.**

**1.** Create a new **asp.net empty web application** and name it **CompanyClient**  
**2.** Right click on **References** folder and select **Add Service Reference** option. In the address textbox type **http://localhost:8080/** and click on **GO** button. In the namespace textbox type **CompanyService** and click **OK.** This should generate a proxy class to communicate with the service.  
**3.** Add a new webform. Copy and paste the following HTML in WebForm1.aspx  
<div style="font-family:Arial">  
    <table style="border:1px solid black">  
        <tr>  
            <td>  
                <asp:Button ID="Button1" runat="server" Text="Get Public Information"   
                    onclick="Button1\_Click" Width="300px" />  
            </td>  
            <td>  
                <asp:Label ID="Label1" runat="server" Font-Bold="true"></asp:Label>              
            </td>  
        </tr>  
        <tr>  
            <td>  
                <asp:Button ID="Button2" runat="server" Text="Get Confidential Information"   
                    onclick="Button2\_Click" Width="300px" />  
            </td>  
            <td>  
                <asp:Label ID="Label2" runat="server" Font-Bold="true"></asp:Label>              
            </td>  
        </tr>  
    </table>  
</div>  
**4.** Copy and paste the following code in WebForm1.aspx.cs file  
protected void Button1\_Click(object sender, EventArgs e)  
{  
    CompanyService.MyCompanyPublicServiceClient client1 = new  
        CompanyService.MyCompanyPublicServiceClient("BasicHttpBinding\_IMyCompanyPublicService");  
    Label1.Text = client1.GetPublicInformation();  
}  
  
protected void Button2\_Click(object sender, EventArgs e)  
{  
    CompanyService.MyCompanyConfidentialServiceClient client2 = new   
        CompanyService.MyCompanyConfidentialServiceClient("NetTcpBinding\_IMyCompanyConfidentialService");  
    Label2.Text = client2.GetCofidentialInformation();  
}  
  
**Here is the requirement**  
**1.** **MyCompanyPublicService** should be available to everyone behind and outside the firewall  
**2.** **MyCompanyConfidentialService** should be available only with in the company behind the firewall.

But with the client web application we are able to create proxy classes of both the services and invoke both the methods. That is because in our case we are having both the client and the service running on the same machine. When it comes to clients outside the firewall, they will be able to create proxy classes of both the services, but when they attempt to invoke the MyCompanyConfidentialService method they would get an exception.  
  
**Is it possible for a WCF service to implement multiple service contracts?**  
Yes, we make the service class implement multiple service interfaces, and then expose each service using a different endpoint.