# Add a WSDL Interface to an APLNext Web Service

## Overview

One of the ways to provide virtually any application system access to an APL+Win-based web service is to implement a WSDL interface layer on the server-side. A WSDL interface layer provides an industry-standard web service definition which a web service client can use to access the resources provided by the APL+Win-based web service.

With a WSDL interface layer an authenticated client can 'discover' the web service resources available to that client with precise details about the methods supported by the web service including their arguments and results. The WSDL interface provides an XML-format document to the authenticated client containing this information.

The fundamental principle which makes this possible is that a web service client needs to have no knowledge of how and where a web service is implemented. Using the technology described in this document, any client which can implement industry-standard WSDL tools can effectively and efficiently access an APL+Win-based web service.

## WSDL Interface is an option

It is not always necessary to add a WSDL interface layer to an APL+Win-based web service. Adding a WSDL interface layer is the choice of the web service designer based upon:

- The type of clients who will be granted access to the web service
- The type of arguments and results of the APL+Win functions exposed by the APL+Win-based web service

## WSDL Interface Tools

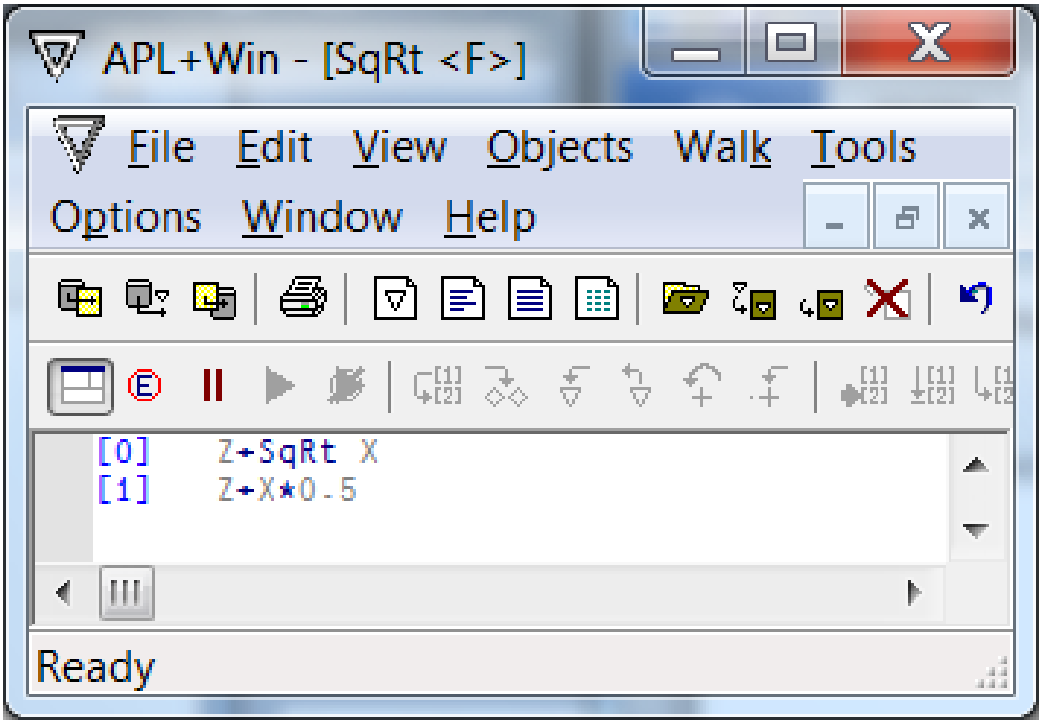
Since APL+Win is software which runs in the Microsoft Windows operating system, a WSDL interface is added to an APL+Win-based web service using Microsoft tools. These tools include:

- Microsoft Internet Information Services (IIS) – This software is included with Windows 7, 8 and all server versions of Microsoft Windows
- Microsoft Windows Communication Foundation (WCF) – Visual Studio includes templates for WCF service application solutions
- Microsoft Visual Studio – This example uses VS2010
- .Net programming language – This example uses C#

## Create the APL+Win workspace and function to be exposed as a web service

Start a developer session of APL+Win and define the 'SqRt' function as follows. In this example an APL+Win function with very simple arguments, results and calculation process is illustrated. In a production environment, the APL+Win function(s) exposed as web service methods may be much more complex. Save the 'SqRt' function in an APL+Win workspace with the name 'WORKSPACE1.w3' in a location which is accessible to the workstation.



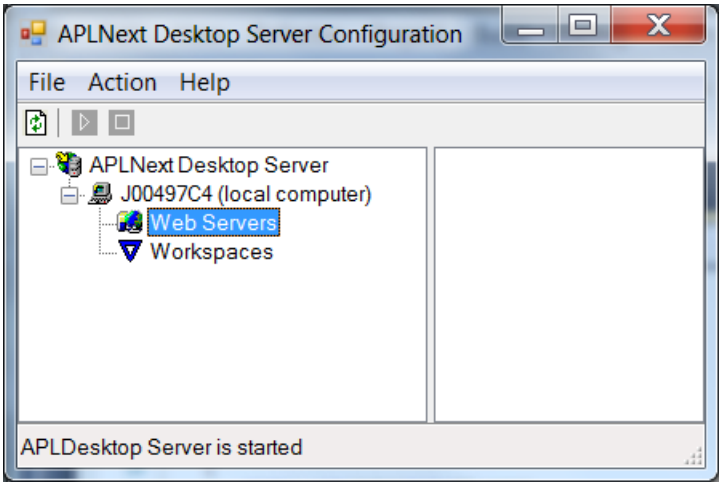

 

## Configure the APL+Win-based web service using APLNext Application Server

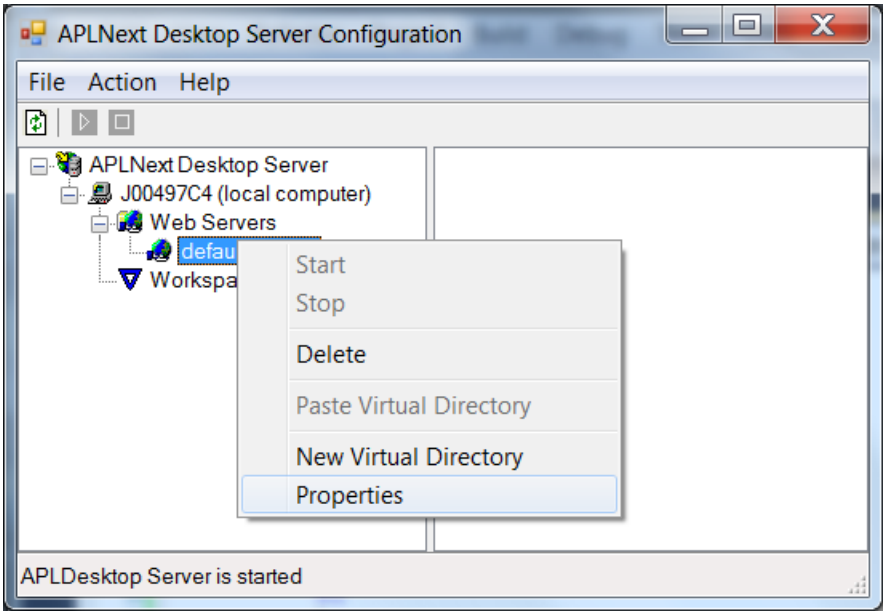
The 'traditional' or 'integrated with IIS' version of APLNext Application Server can be used for this purpose. In this example the APLNext Application Server 'desktop' version is illustrated because it is very convenient when designing, implementing and testing an APL+Win based web service.

After the APLNext Desktop Server software has been installed to the developer workstation, open the 'APLNext Desktop Server Admin' tool and follow the configuration instructions below.

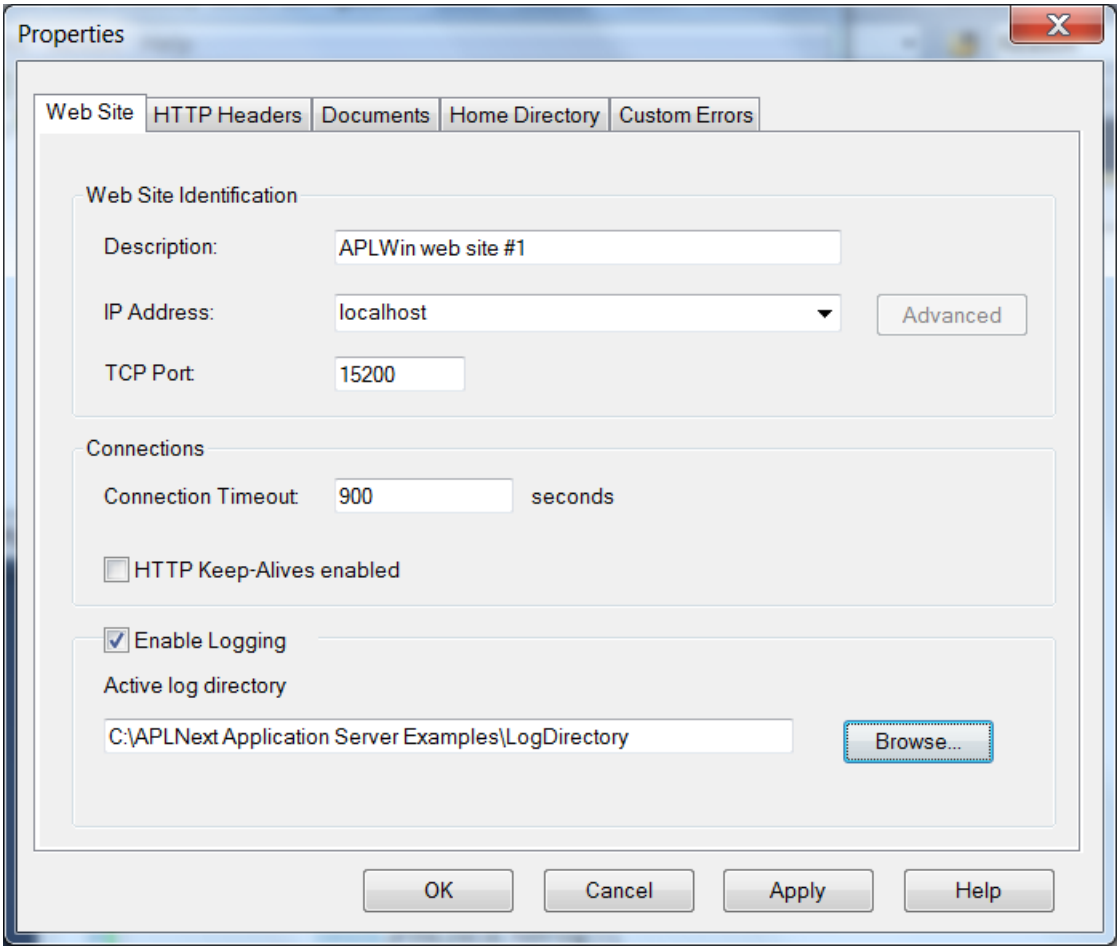
Expand the 'APLNext Desktop Server' tree in the left panel, right click the 'Web Servers' node and select 'New Server':




Right client the 'defaultwebsite' node and select 'Properties':

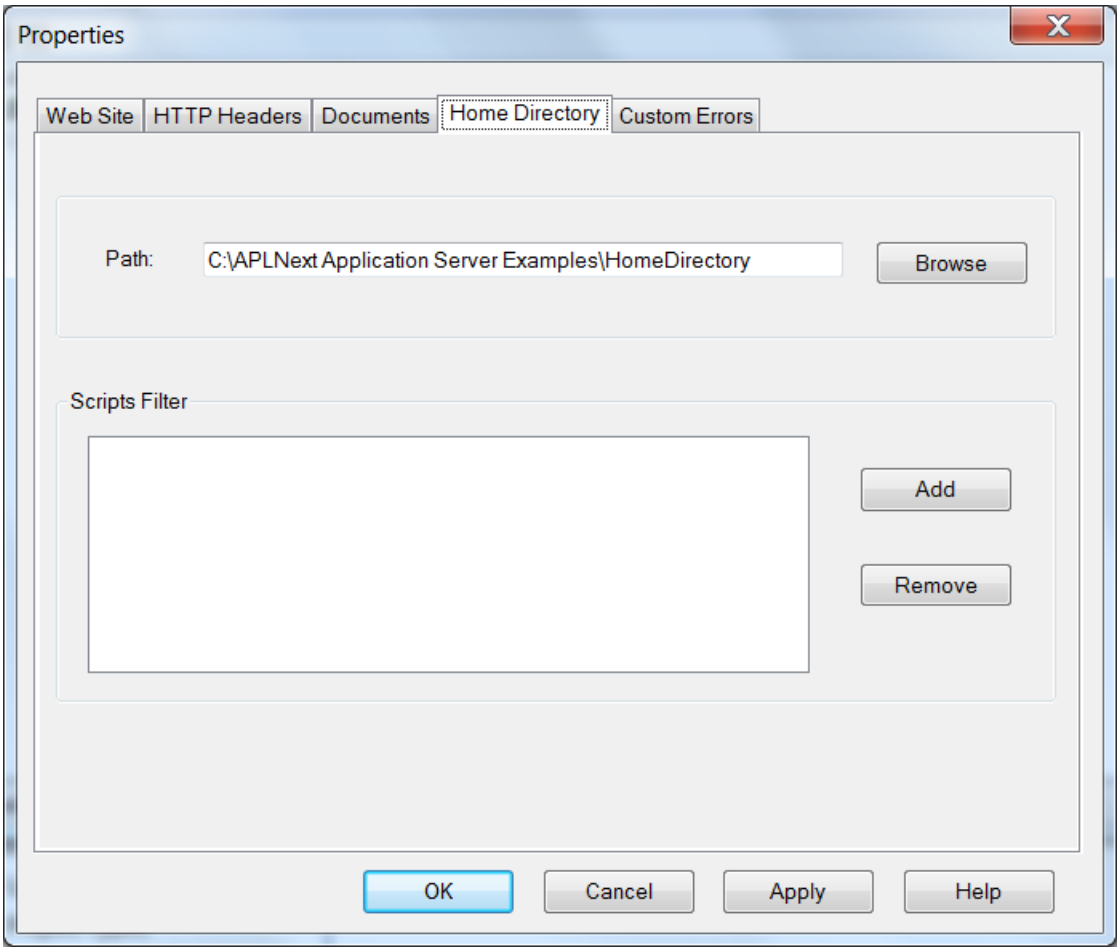


Complete the 'Web Site' tab of the 'properties' as follows. Note that the address of the 'Active log directory' should be customized to the workstation. The 'IP Address' field value is set to 'localhost' because the APL+Win-based web service will be tested on the workstation before it is deployed to a production server.

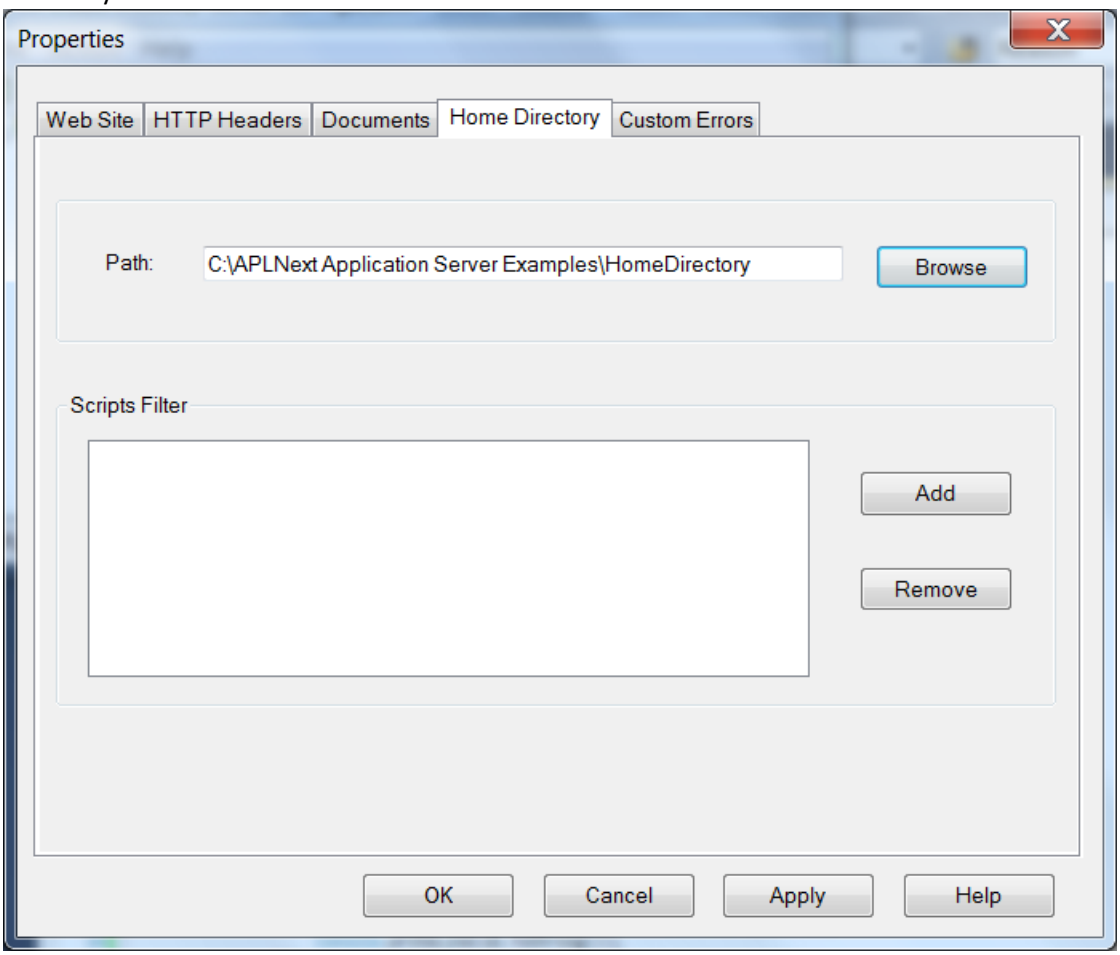


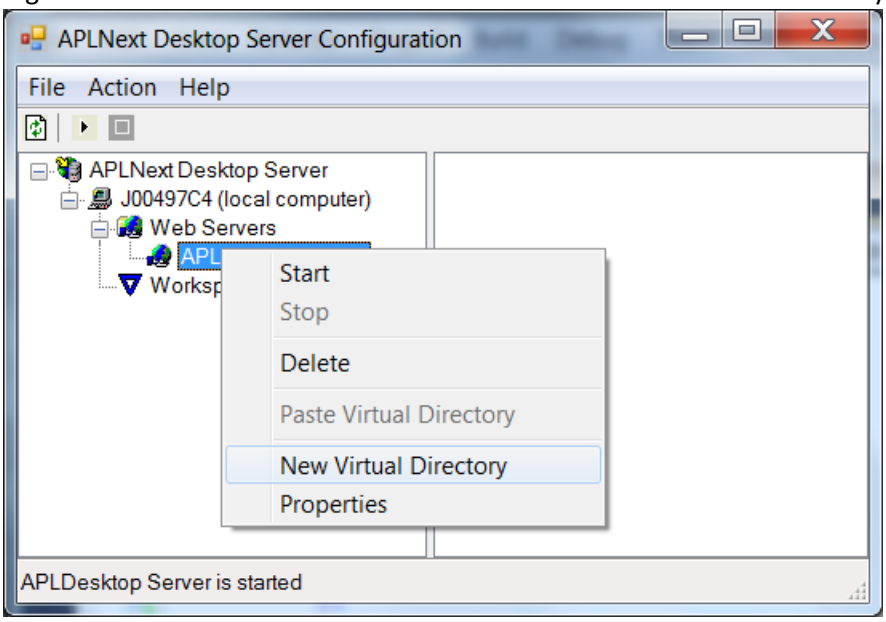
Complete the ‘Web Site’ tab of the ‘properties’ as follows.  Note that the ‘Path’ of the ‘Home Directory’ should be customized to the workstation.



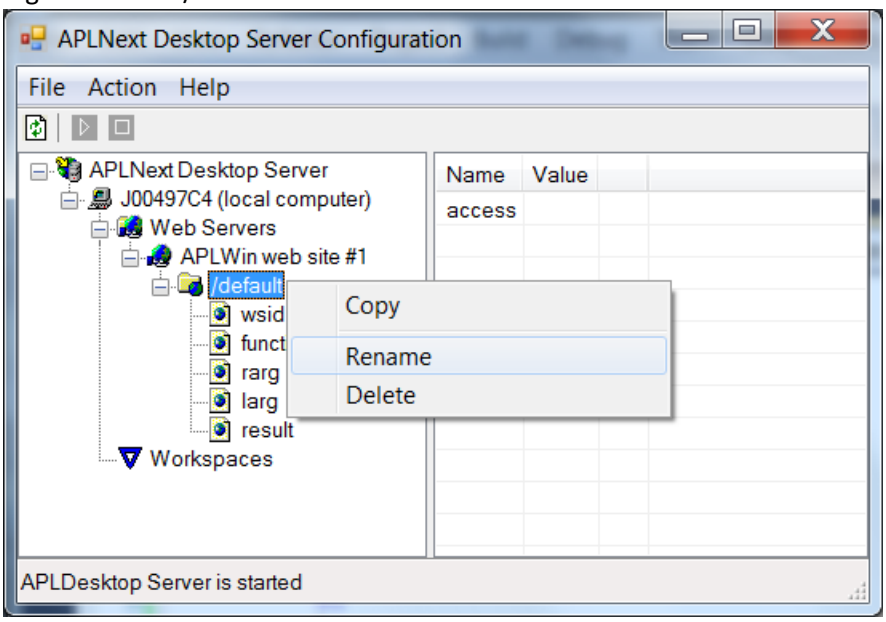
Complete the ‘Home Directory’ tab of the ‘properties’ as follows. Note that the ‘Path’ of the ‘Home Directory’ should be customized to the workstation.



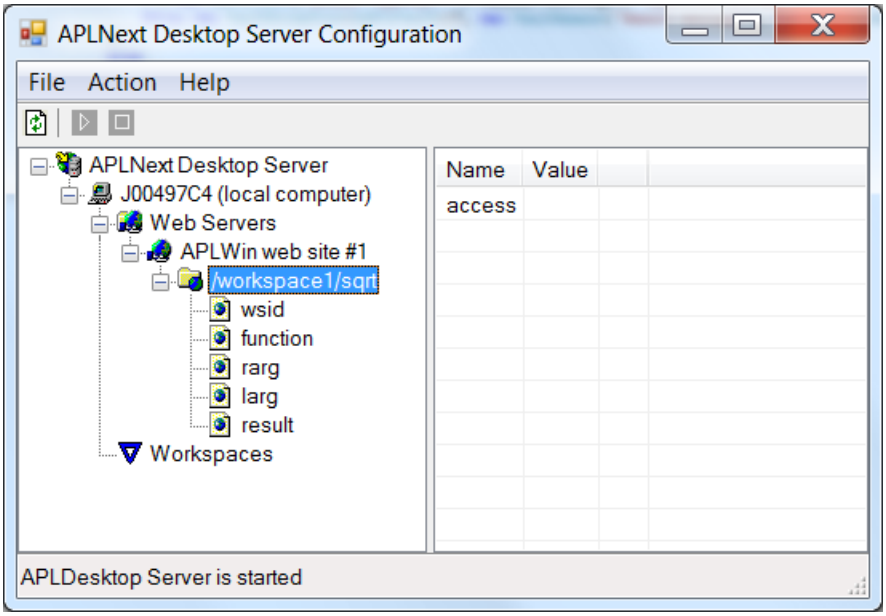
Right click the ‘APLWin web site #1 node and select ‘New Virtual Directory’.



Right click the '/default' node and select 'Rename'.

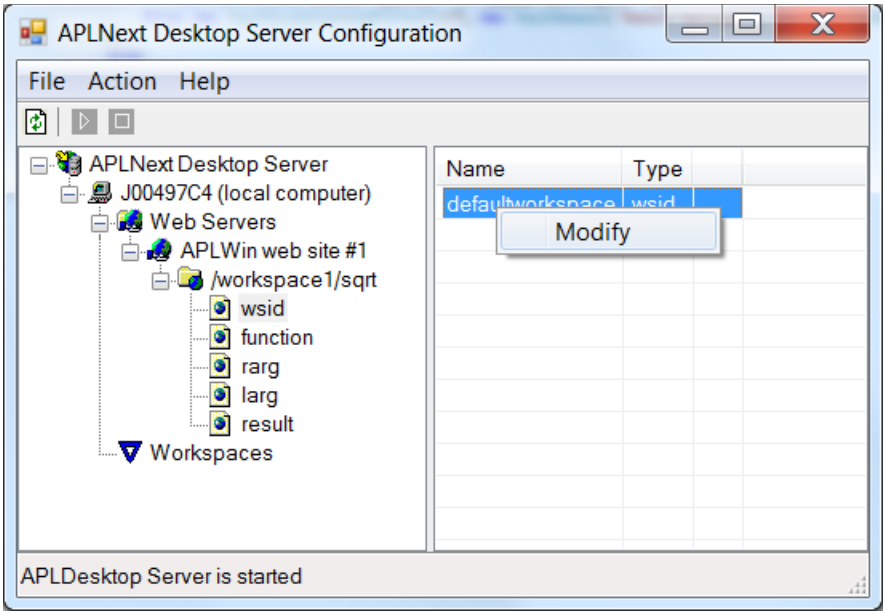



Rename the '/default' virtual path to '/workspace1/sqrt'.

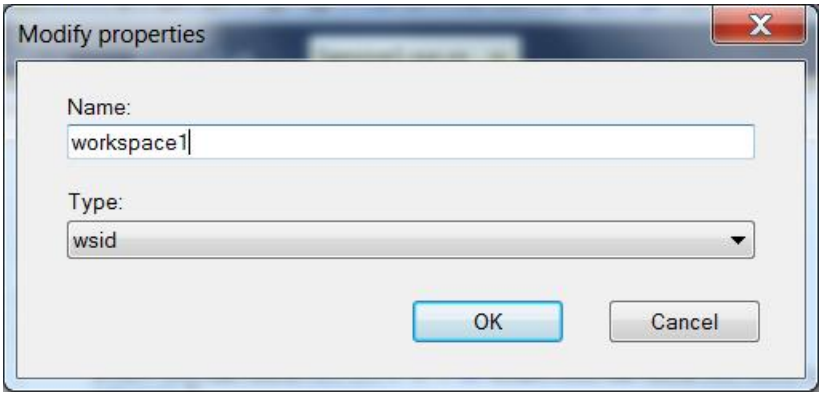


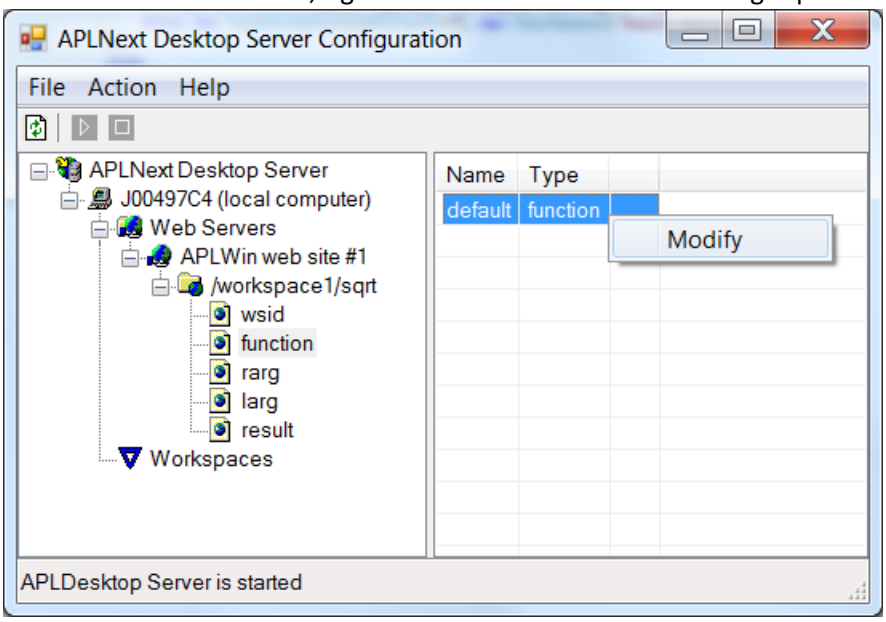
Select the ‘wsid’ node and right click the ‘defaultworkspace’ item in the right pane and select ‘Modify’.



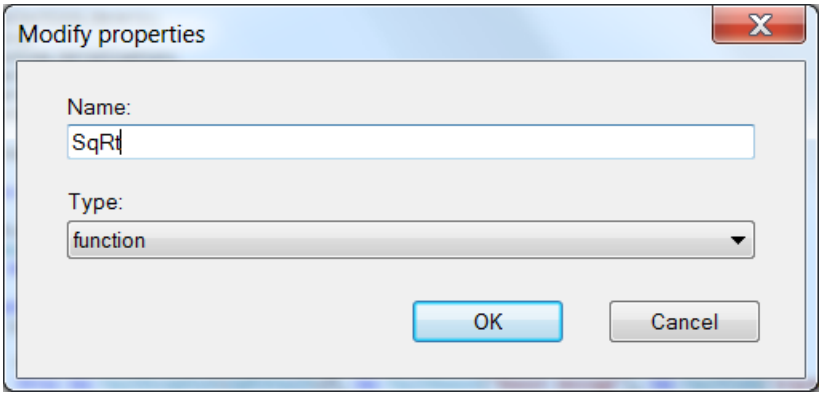

Change the ‘Name’ to ‘workspace1’ and click ‘OK’.



Select the ‘function’ node, right click the ‘default’ item in the right pane and select ‘Modify’.

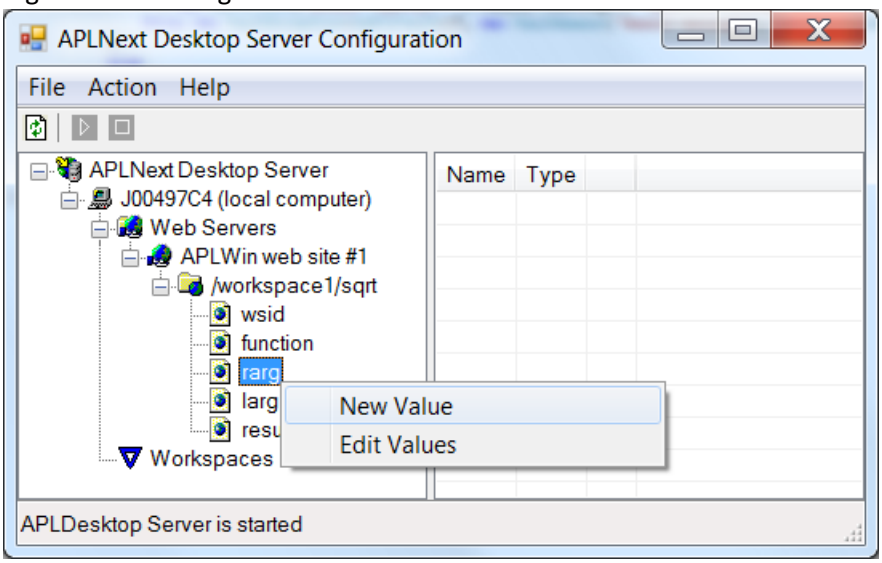



Change the function name to ‘SqRt’ and click ‘OK’.

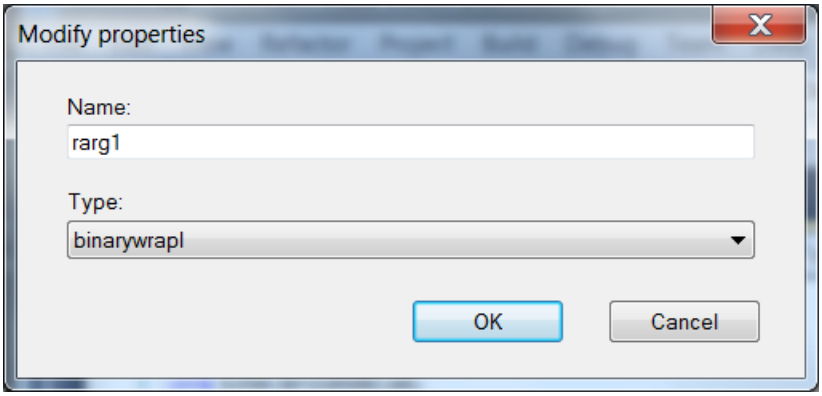


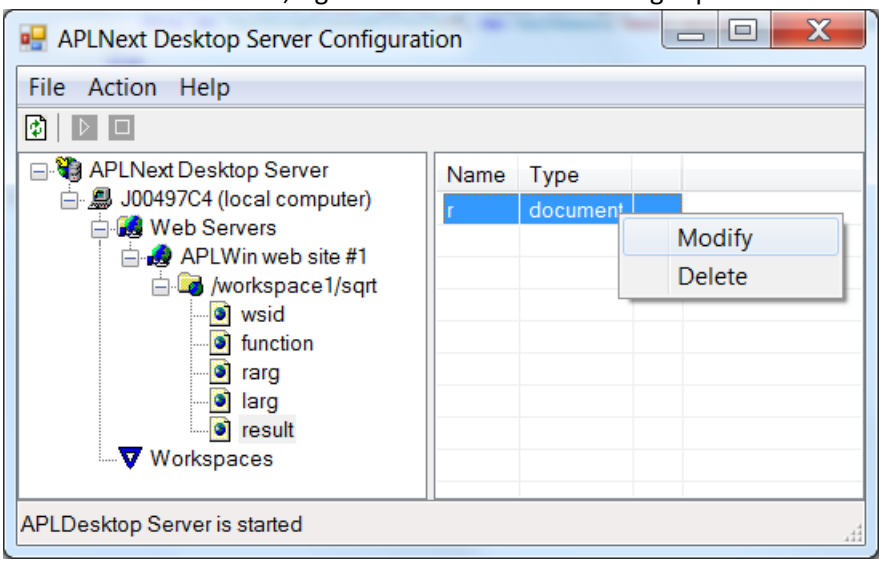
Right click the ‘rarg’ node and select ‘New Value’.



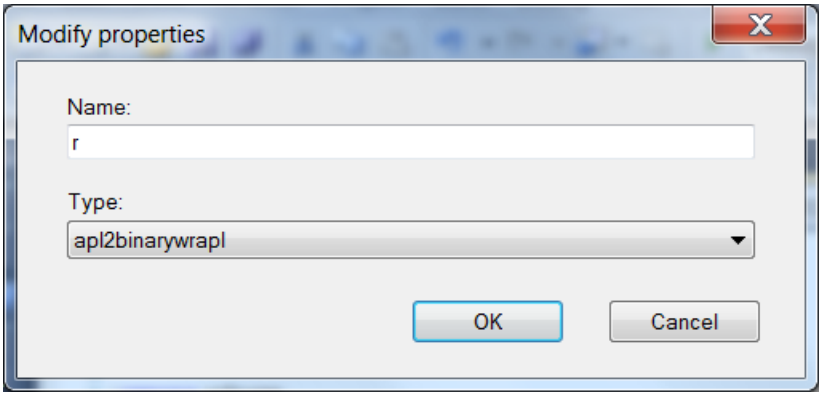

Complete the ‘Modify Properties’ dialog as follows and click ‘OK’. The ‘binarywrapl’ ‘type’ option is selected here because the WSDL Interface will use the APLNext WebTransfer ‘SendObject()’ method to communicate with the APL+Win-based web service. If the ‘SendSoapObject()’ were used, the ‘soapwrapl’ ‘type’ option would be selected here.



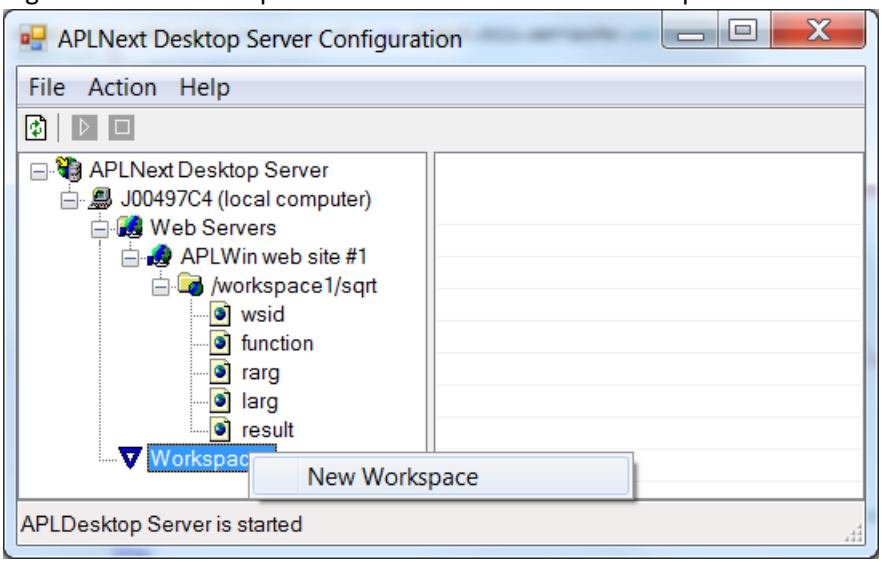
Select the ‘Result’ node, right click the ‘r’ item in the right pane and select ‘Modify’.



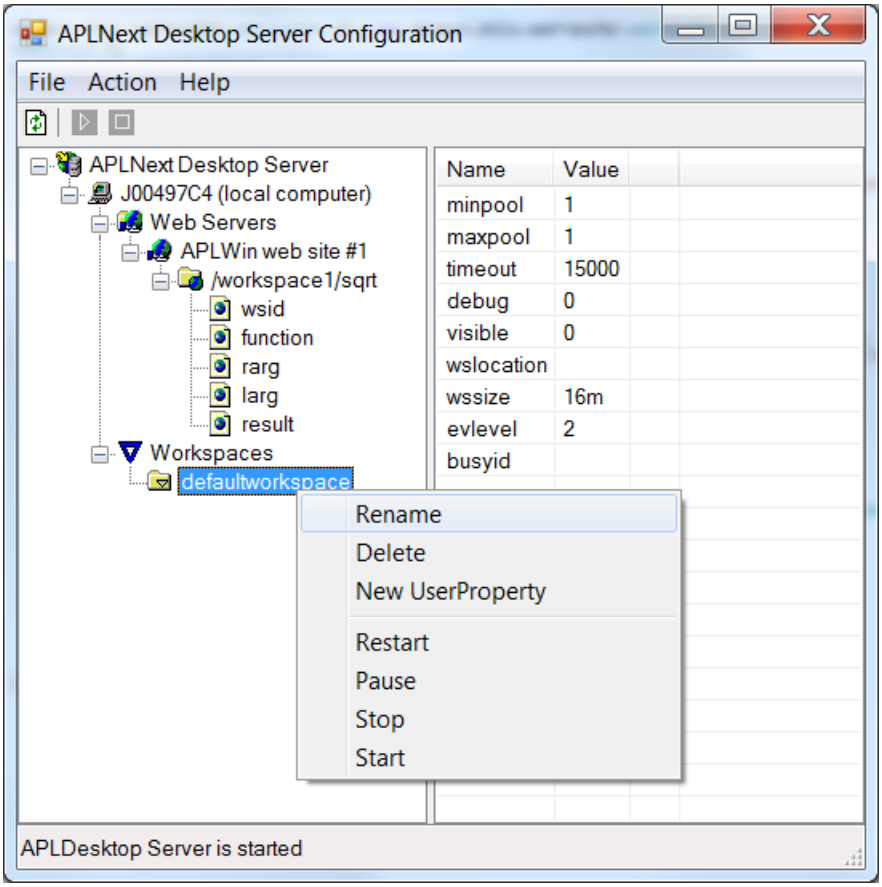

Complete the ‘Modify Properties’ dialog as follows and click ‘OK’. The ‘apl2binarywrapl’ ‘type’ option is selected here because the WSDL Interface will use the APLNext WebTransfer ‘SendObject()’ method to communicate with the APL+Win-based web service. If the ‘SendSoapObject()’ were used, the ‘apl2soapwrapl’ ‘type’ option would be selected here.



Right click the ‘Workspaces’ node and select ‘New Workspace’.

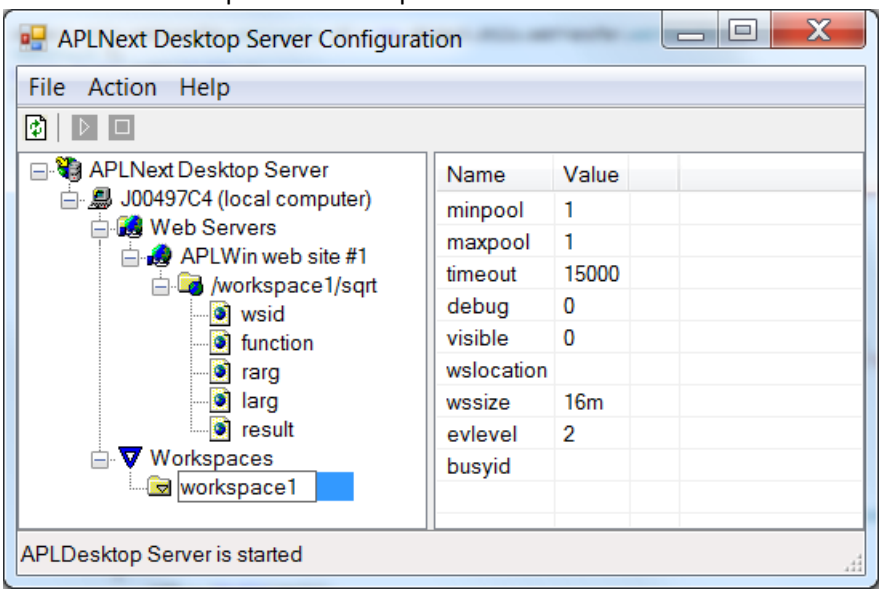



Right click the ‘defaultworkspace’ node and select ‘Rename’.

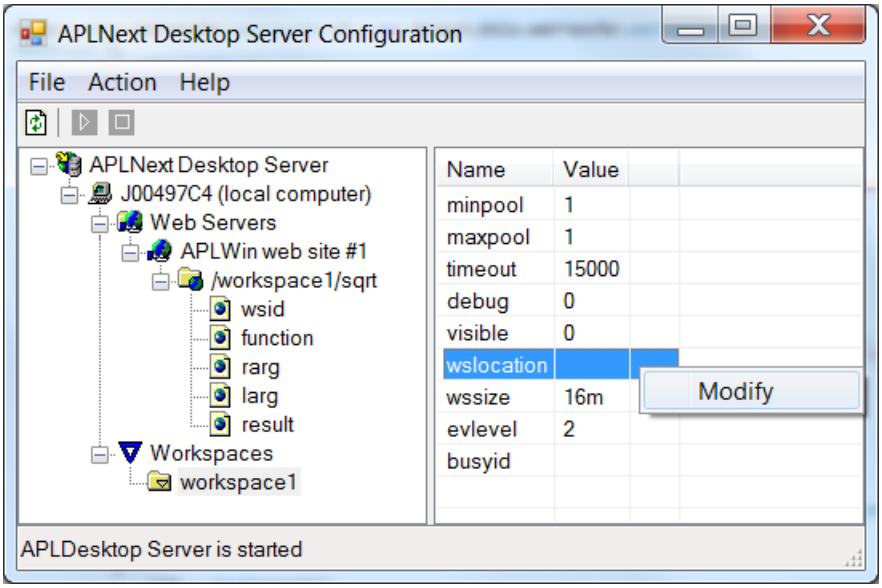


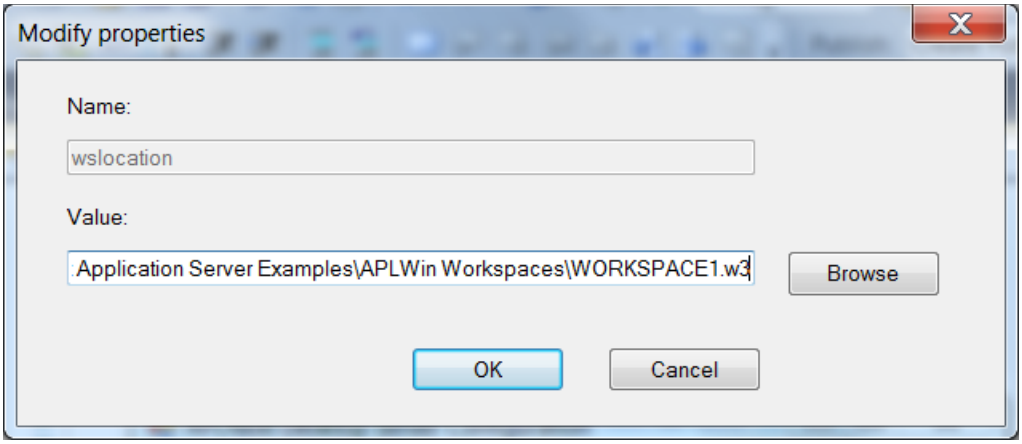
Rename the workspace to ‘workspace1’.




Right click the ‘wslocation’ item in the right pane and select ‘Modify’.



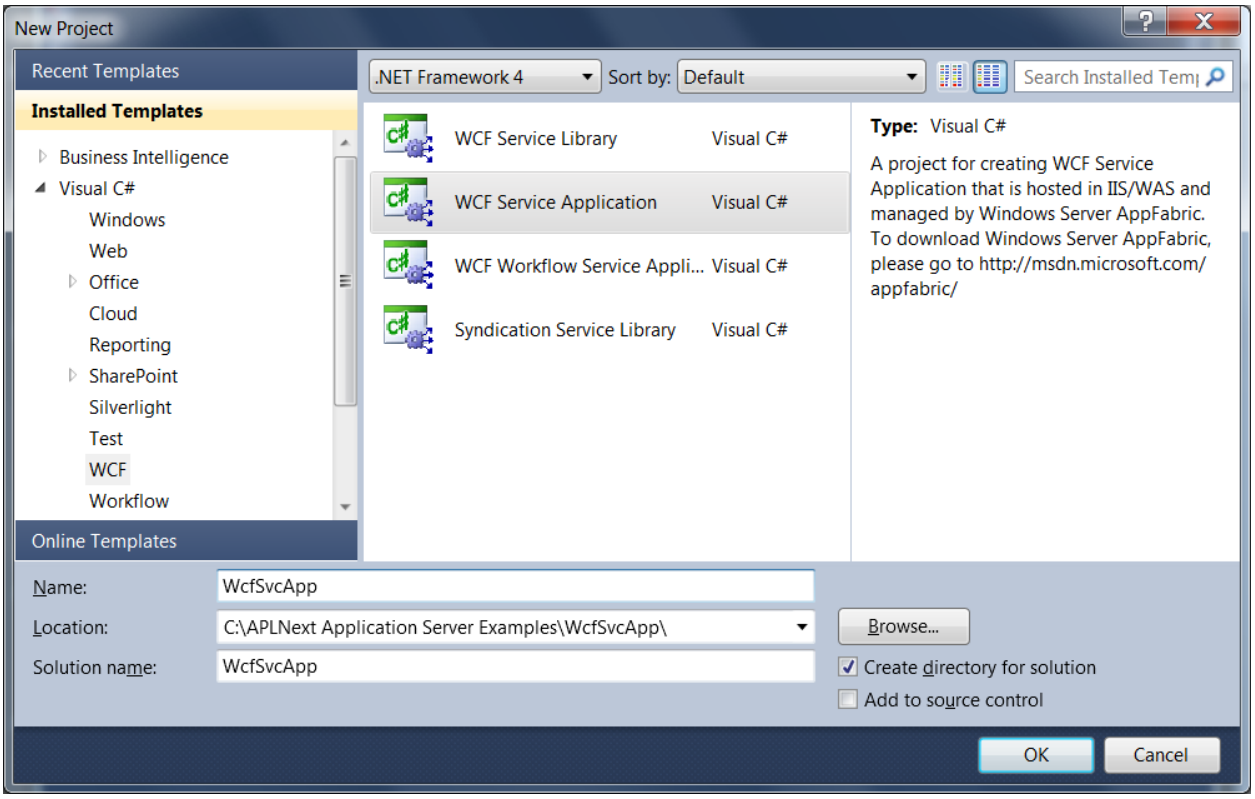
Set the ‘Value’ of the ‘wslocation’ using the ‘Browse’ button to the location of the ‘WORKSPACE1.w3’ APL+Win workspace previously created and click ‘OK’. Note that the ‘Value’ of the workspace should be customized to the workstation.



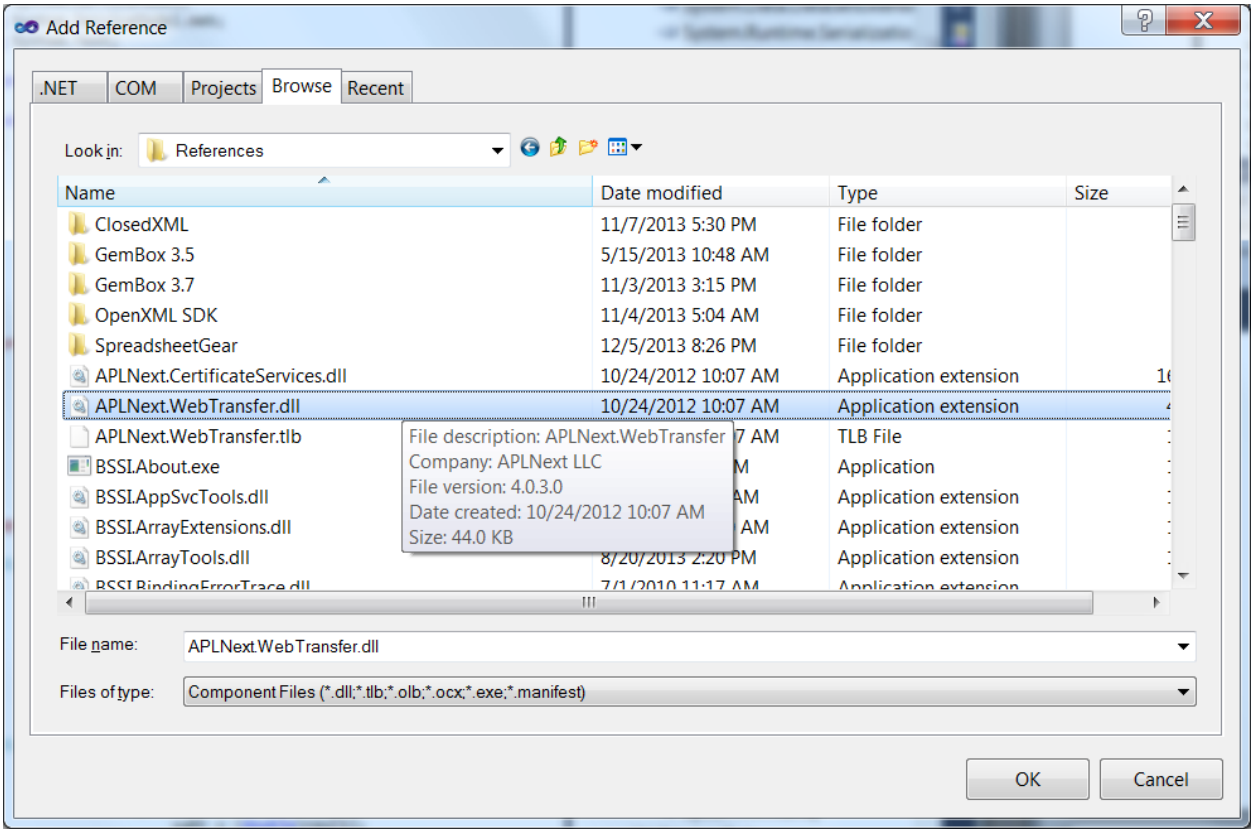

 

## Create a Visual Studio WCF Service Application Solution

Open Visual Studio and use the 'File > New Project > Visual C# > WCF' menu item to select the 'WCF Service Application' project template. Name the project 'WcfSvcApp' and select a suitable location for the Visual Studio solution on the workstation.



From the Visual Studio ‘Solution Explorer’ right click the ‘References’ node of the ‘WcfSvcApp’ project and select ‘Add Reference’. In the ‘Add Reference’ dialog select the ‘Browse’ tab, select the ‘APLNext.WebTransfer.dll’ component and click ‘OK’. This component is included with the APLNext Application Server license. It provides .Net or ActiveX access to an APL+Win-based web service.



Replace the content of the 'IService1.cs' file with the following source code. The '[OperationContract]' method has been named 'SqRt' to correspond to the APL+Win function name. The 'SqRTFault' class has been defined so that exceptions can be properly provided to clients of the web service. Two custom error information strings 'AppInfo1' and 'AppInfo2' have been created to provide additional details to the client if such an exception occurs.

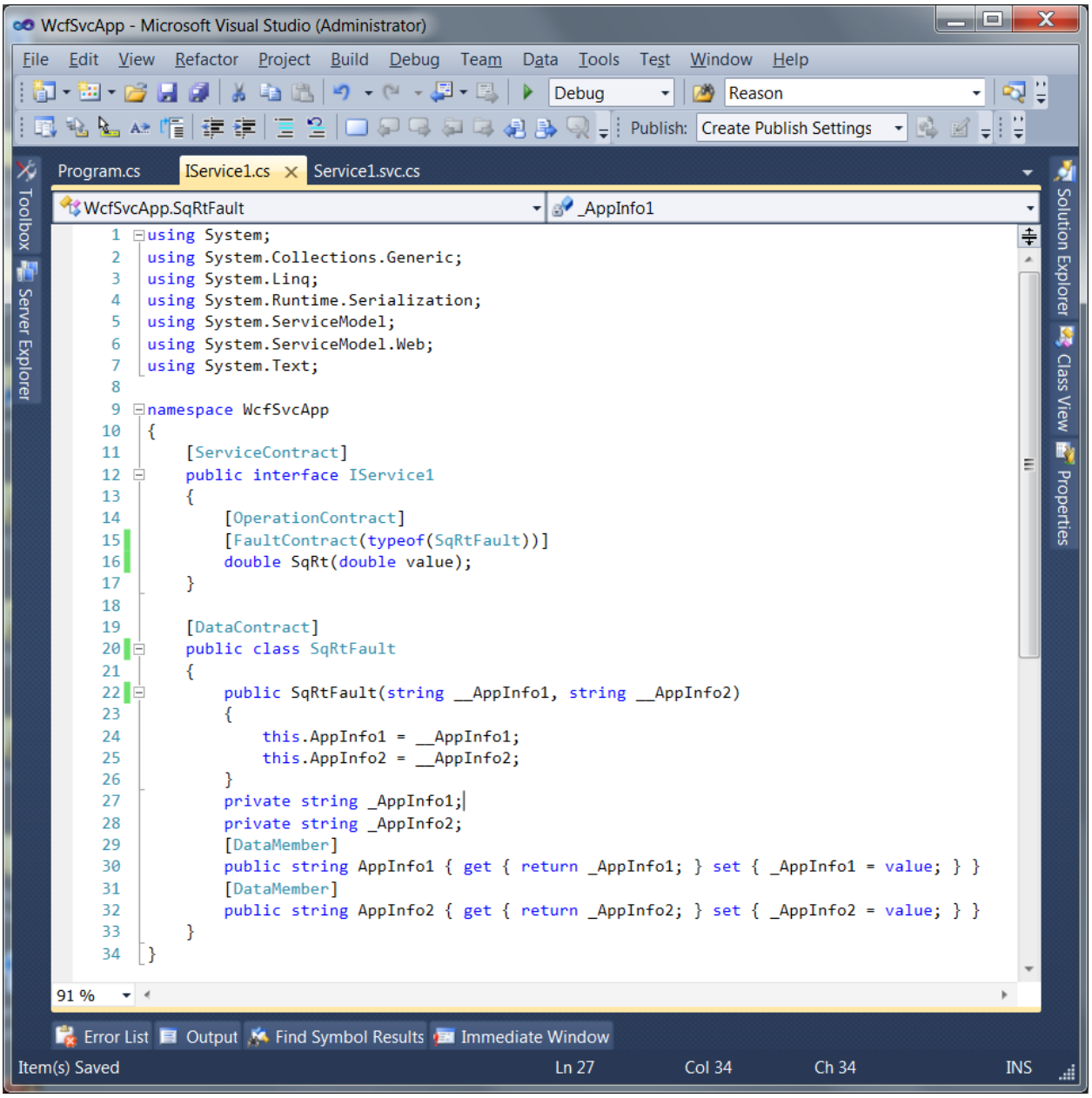
This file defines the interface between the client and the web service.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

namespace WcfSvcApp
{
    [ServiceContract]
    public interface IService1
    {
        [OperationContract]
        [FaultContract(typeof(SqRtFault))]
        double SqRt(double value);
    }

    [DataContract]
    public class SqRtFault
    {
        public SqRtFault(string __AppInfo1, string __AppInfo2)
        {
            this.AppInfo1 = __AppInfo1;
            this.AppInfo2 = __AppInfo2;
        }
        private string _AppInfo1;
        private string _AppInfo2;
        [DataMember]
        public string AppInfo1 { get { return _AppInfo1; } set { _AppInfo1 = value; } }
        [DataMember]
        public string AppInfo2 { get { return _AppInfo2; } set { _AppInfo2 = value; } }
    }
}
```



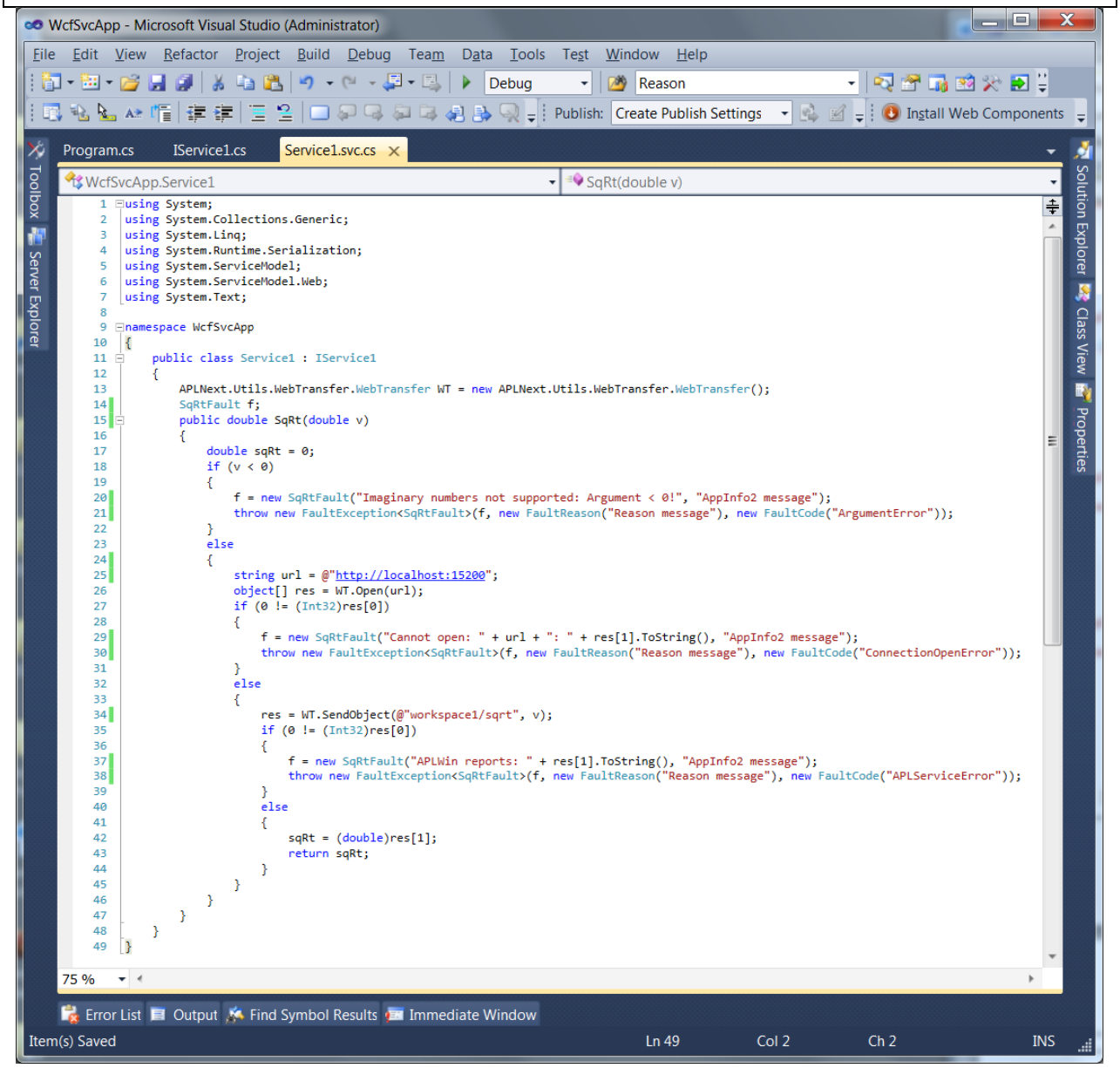
Replace the content of the 'Service1.svc.cs' file with the following source code. This file defines the implementation of the web service methods exposed by the web service interface. The WCF web service interfaces with the APL+Win-based web service via the APLNext WebTransfer class.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

namespace WcfSvcApp
{
    public class Service1 : IService1
    {
        APLNext.Utils.WebTransfer.WebTransfer WT = new
APLNext.Utils.WebTransfer.WebTransfer();
        SqRtFault f;
        public double SqRt(double v)
        {
            double sqRt = 0;
            if (v < 0)
            {
                f = new SqRtFault("Imaginary numbers not supported: Argument < 0!",
"AppInfo2 message");
                throw new FaultException<SqRtFault>(f, new FaultReason("Reason message"),
new FaultCode("ArgumentError"));
            }
            else
            {
                string url = @"http://localhost:15200";
                object[] res = WT.Open(url);
                if (0 != (Int32)res[0])
                {
                    f = new SqRtFault("Cannot open: " + url + ": " + res[1].ToString(),
"AppInfo2 message");
                    throw new FaultException<SqRtFault>(f, new FaultReason("Reason
message"), new FaultCode("ConnectionOpenError"));
                }
                else
                {
                    res = WT.SendObject(@"workspace1/sqrt", v);
                    if (0 != (Int32)res[0])
                    {
                        f = new SqRtFault("APLWin reports: " + res[1].ToString(),
"AppInfo2 message");
                        throw new FaultException<SqRtFault>(f, new FaultReason("Reason
message"), new FaultCode("APLServiceError"));
                    }
                    else
                    {
                        sqRt = (double)res[1];
                        return sqRt;
                    }
                }
            }
        }
```

```
    }
}
```



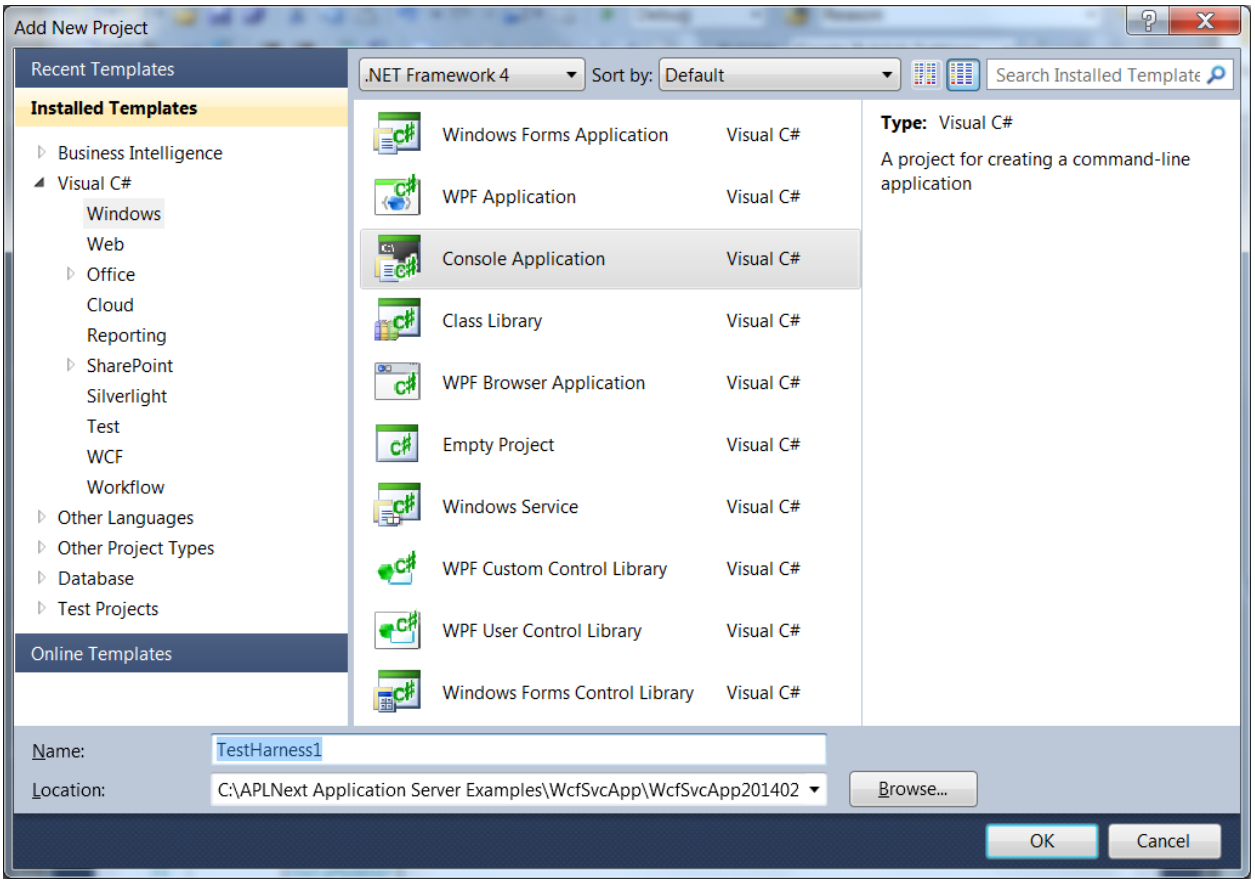

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

namespace WcfSvcApp
{
    public class Service1 : IService1
    {
        APLNext.Utils.WebTransfer.WebTransfer WT = new APLNext.Utils.WebTransfer.WebTransfer();
        SqRtFault f;
        public double SqRt(double v)
        {
            double sqRt = 0;
            if (v < 0)
            {
                f = new SqRtFault("Imaginary numbers not supported: Argument < 0!", "AppInfo2 message");
                throw new FaultException<SqRtFault>(f, new FaultReason("Reason message"), new FaultCode("ArgumentError"));
            }
            else
            {
                string url = @"http://localhost:15200";
                object[] res = WT.Open(url);
                if (0 != (Int32)res[0])
                {
                    f = new SqRtFault("Cannot open: " + url + ": " + res[1].ToString(), "AppInfo2 message");
                    throw new FaultException<SqRtFault>(f, new FaultReason("Reason message"), new FaultCode("ConnectionOpenError"));
                }
                else
                {
                    res = WT.SendObject(@"workspace1/sqrt", v);
                    if (0 != (Int32)res[0])
                    {
                        f = new SqRtFault("APLWin reports: " + res[1].ToString(), "AppInfo2 message");
                        throw new FaultException<SqRtFault>(f, new FaultReason("Reason message"), new FaultCode("APLServiceError"));
                    }
                    else
                    {
                        sqRt = (double)res[1];
                        return sqRt;
                    }
                }
            }
        }
    }
}
```

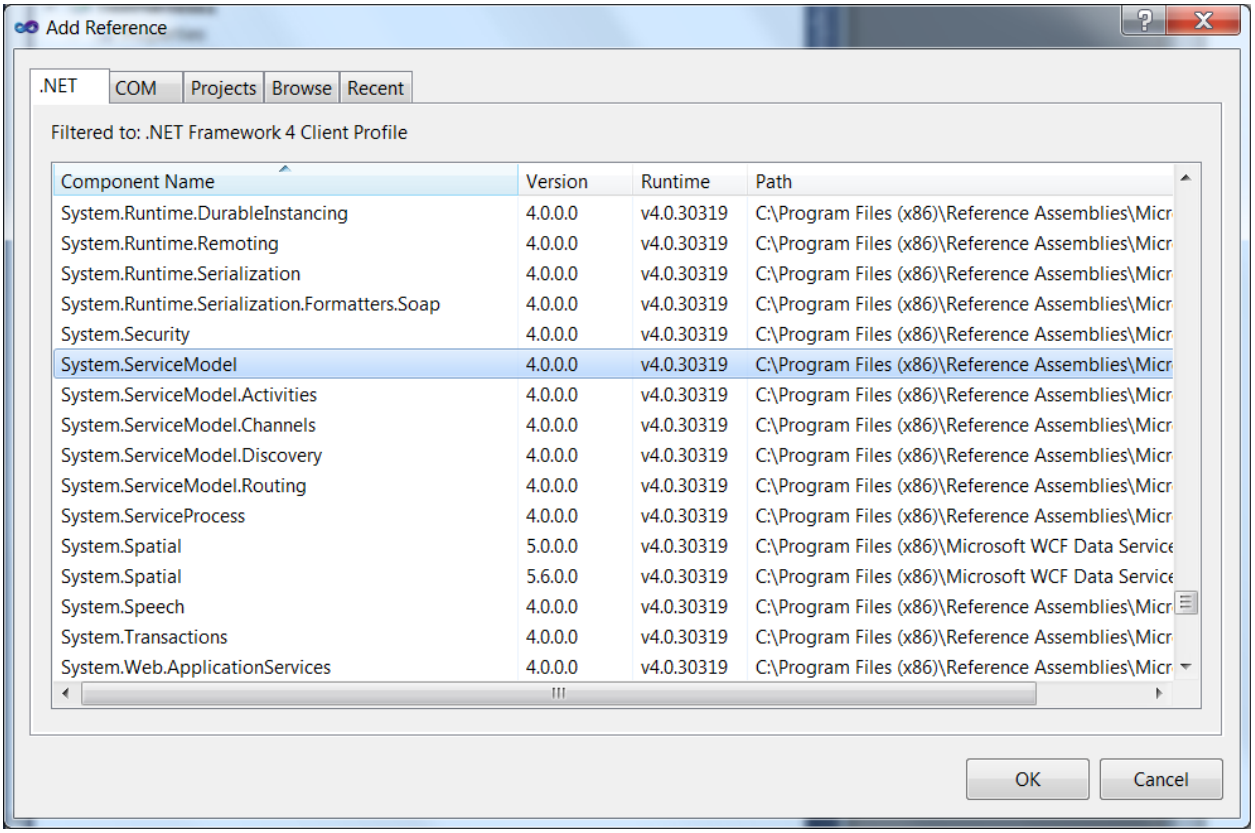
## Create a test client for the WSDL-enabled, APL+Win-based web service

In this example a C# test client is defined which resides on the same workstation as the server. This is a convenient arrangement for designing, developing and testing a WSDL-enabled, APL+Win-based web service. In a production environment the APLNext Application Server and the WCF service would be deployed to a product web server and the service client would be deployed on suitable clients who need the features of the web service, such as client workstations, mobile devices (phones, tablets, laptops) or other servers.

Use the Visual Studio 'File > Add > New Project > Windows' menu item to select the 'Console Application' project template. Name the new project 'TestHarness1'.

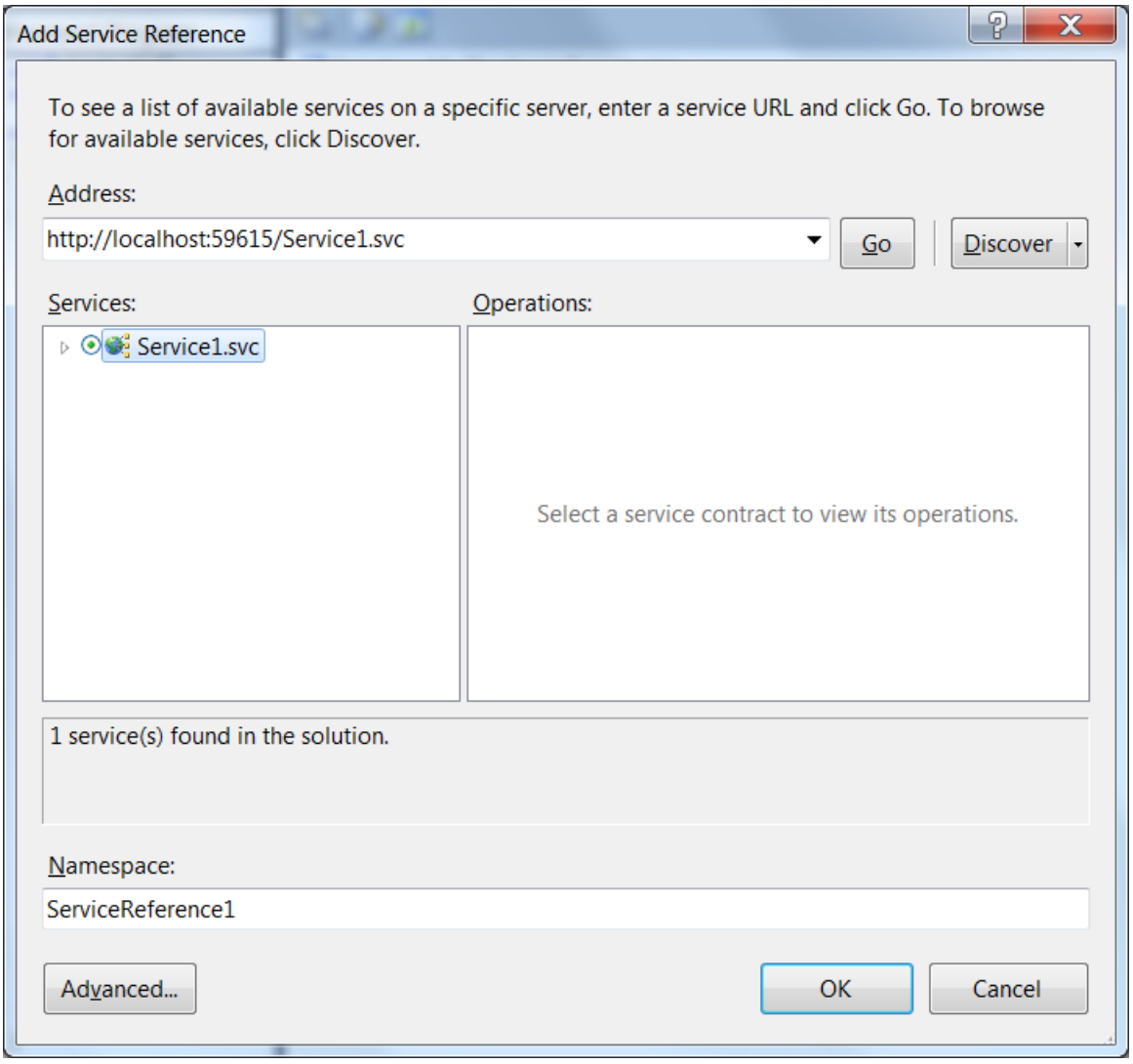


From the Visual Studio ‘Solution Explorer’ right click the ‘References’ node of the ‘TestHarness1’ project and select ‘Add Reference’. From the ‘Add Reference’ dialog select the ‘System.ServiceModel’ component and click ‘OK’.



From the Visual Studio 'Solution Explorer' right click the 'References' node of the 'TestHarness1' project and select 'Add Service Reference'. On the 'Add Service Reference' dialog, click the 'Discover' button, select the 'Service1.svc' and click the 'OK' button. During the testing phase of this project the service url is 'localhost:59615/Service1.svc', but in a production environment the service url will be the production server's IP address.

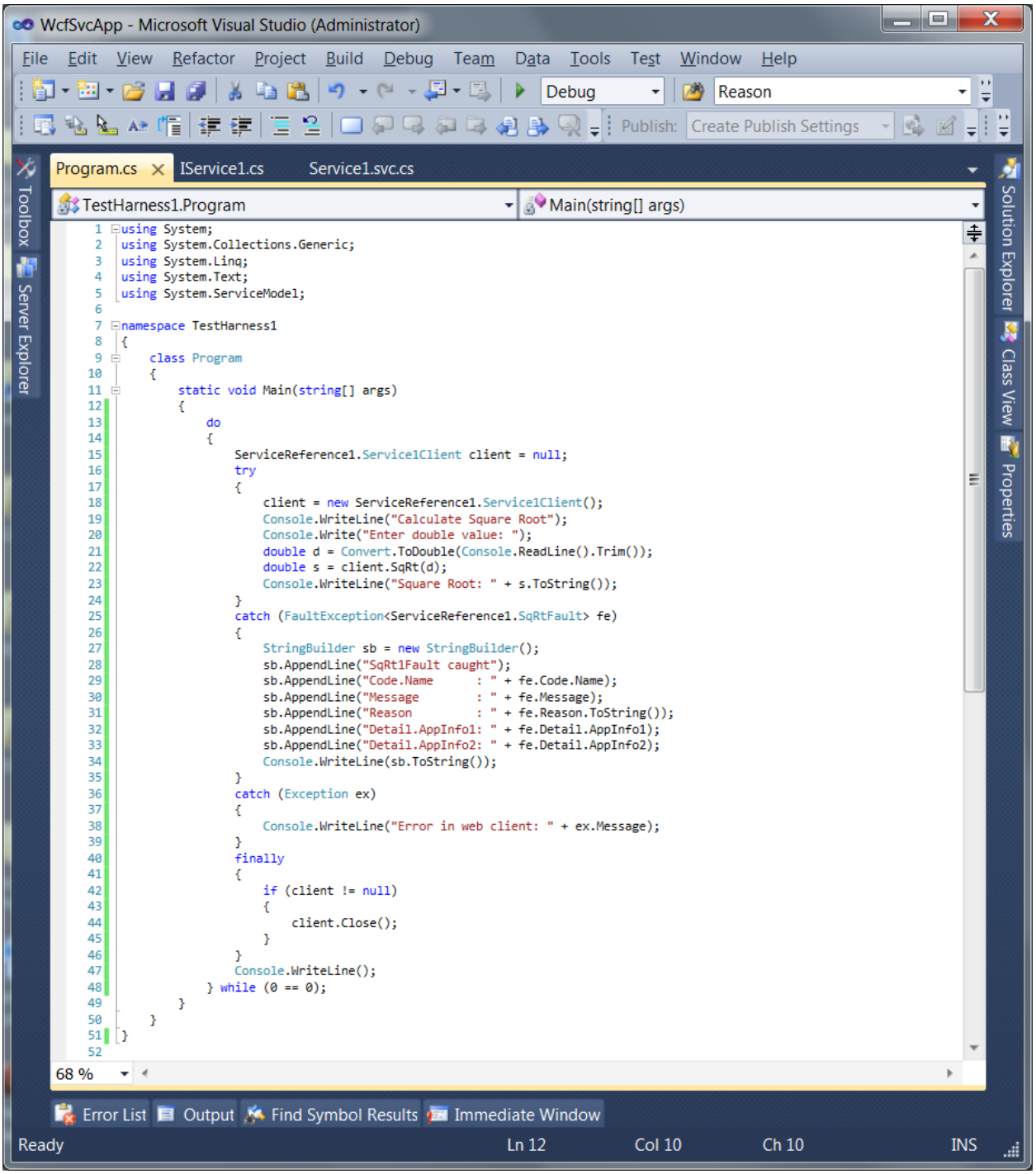
In this step Visual Studio implicitly uses the WSDL document exposed by the WCF service to 'discover' the web service method available, i.e. 'SqRt()' and its arguments and results.



Replace the content of the ‘Program.cs’ file with the following source code.

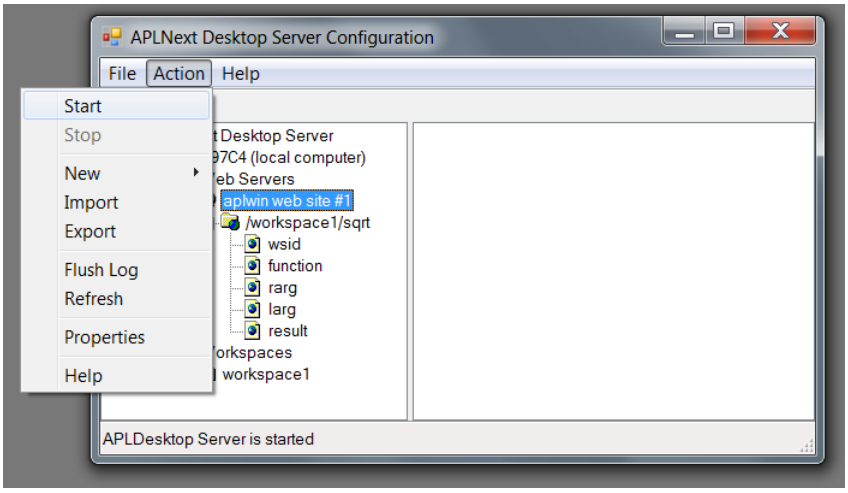
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ServiceModel;

namespace TestHarness1
{
    class Program
    {
        static void Main(string[] args)
        {
            do
            {
                ServiceReference1.Service1Client client = null;
                try
                {
                    client = new ServiceReference1.Service1Client();
                    Console.WriteLine("Calculate Square Root");
                    Console.Write("Enter double value: ");
                    double d = Convert.ToDouble(Console.ReadLine().Trim());
                    double s = client.SqRt(d);
                    Console.WriteLine("Square Root: " + s.ToString());
                }
                catch (FaultException<ServiceReference1.SqRtFault> fe)
                {
                    StringBuilder sb = new StringBuilder();
                    sb.AppendLine("SqRt1Fault caught");
                    sb.AppendLine("Code.Name      : " + fe.Code.Name);
                    sb.AppendLine("Message        : " + fe.Message);
                    sb.AppendLine("Reason         : " + fe.Reason.ToString());
                    sb.AppendLine("Detail.AppInfo1: " + fe.Detail.AppInfo1);
                    sb.AppendLine("Detail.AppInfo2: " + fe.Detail.AppInfo2);
                    Console.WriteLine(sb.ToString());
                }
                catch (Exception ex)
                {
                    Console.WriteLine("Error in web client: " + ex.Message);
                }
                finally
                {
                    if (client != null)
                    {
                        client.Close();
                    }
                }
                Console.WriteLine();
            } while (0 == 0);
        }
    }
}
```

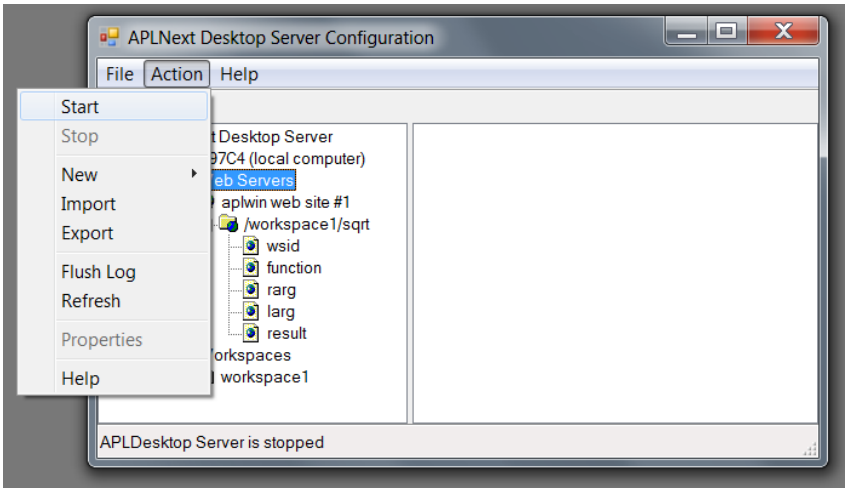


## Start the APL+Win- and WCF-based web services

Start the APLNext Application Server ‘aplwin web site #1’ server:

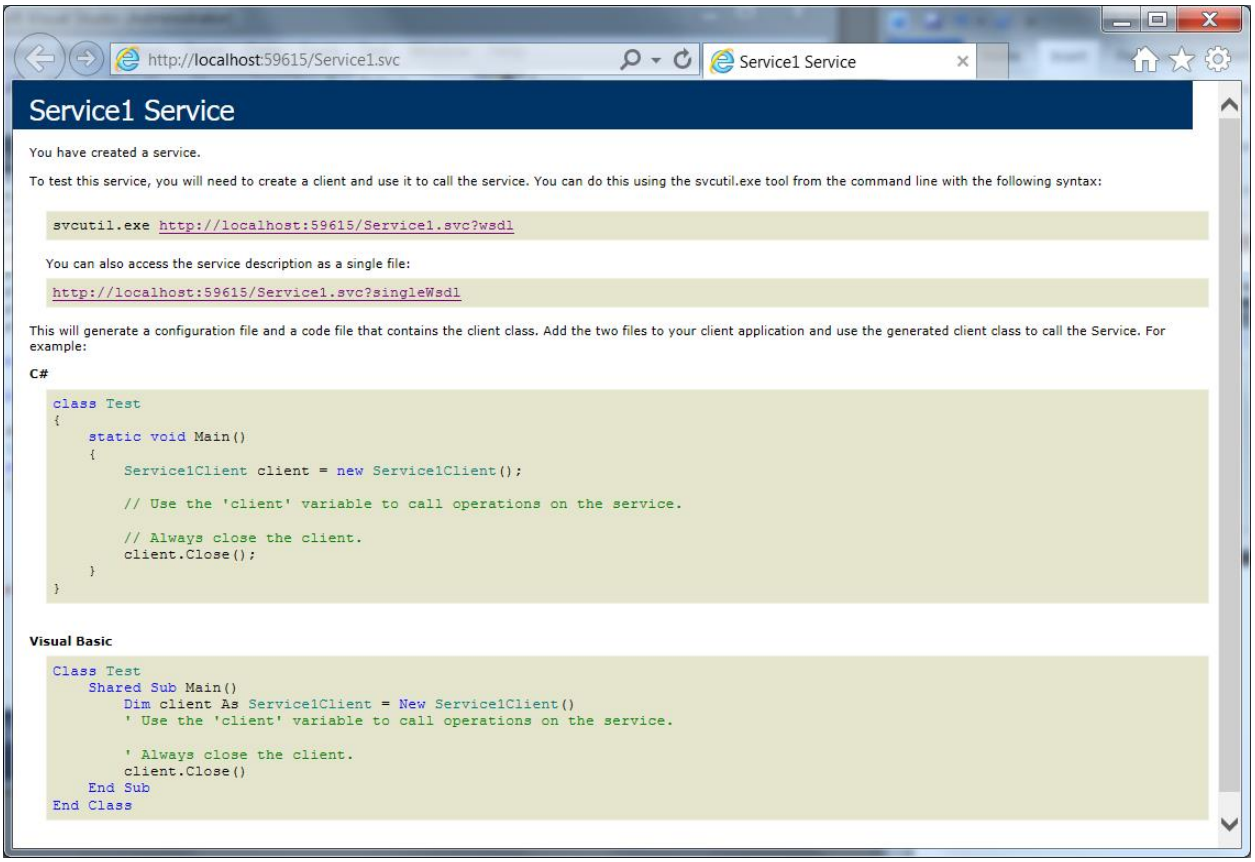



Start the APLNext Application Server ‘WebServers’:

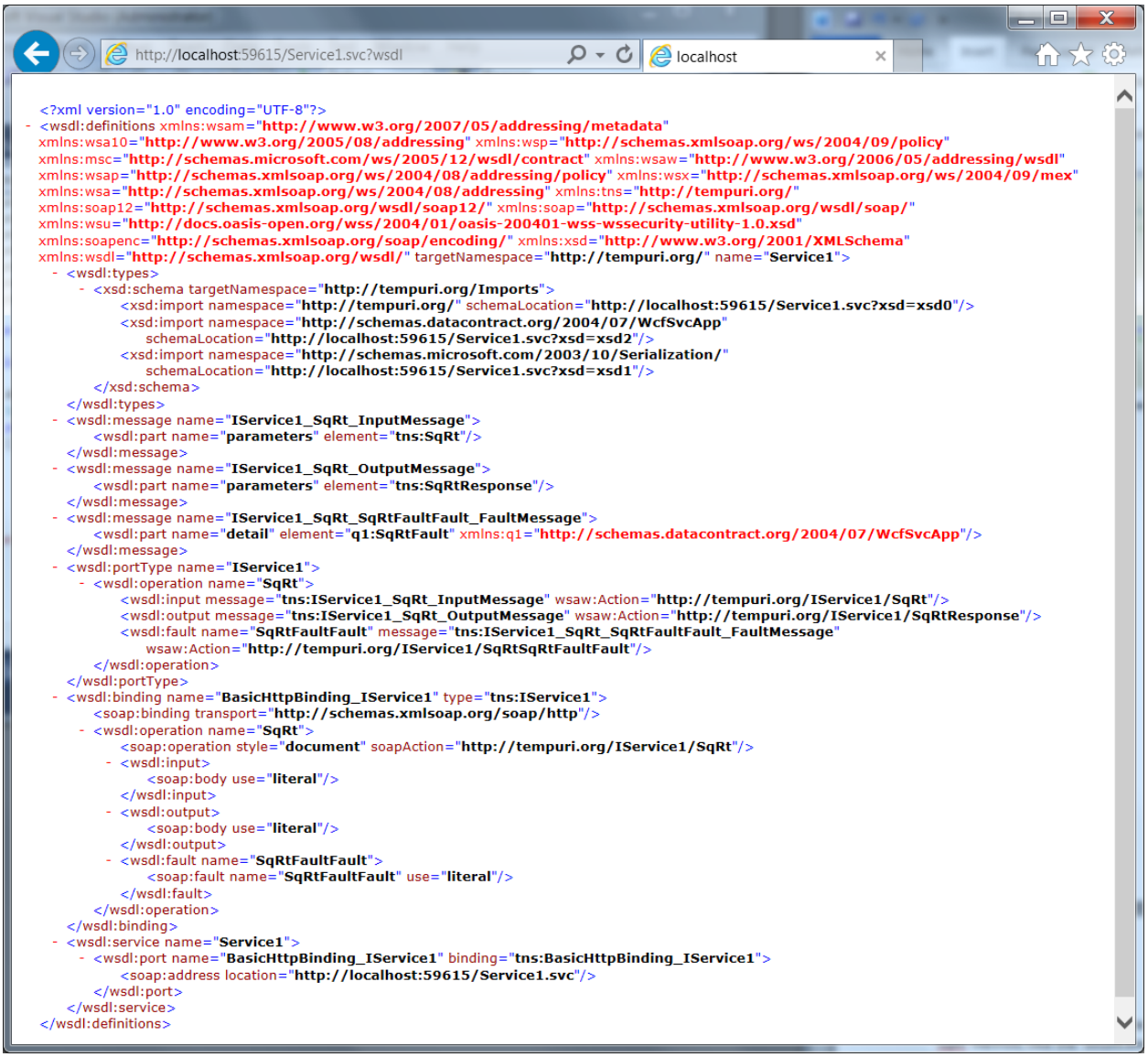


Start the WCF-based web service supporting the WSDL interface layer. From the Visual Studio ‘Solution Explorer’ right click the ‘Service1.svc’ node and select ‘View in Browser’.



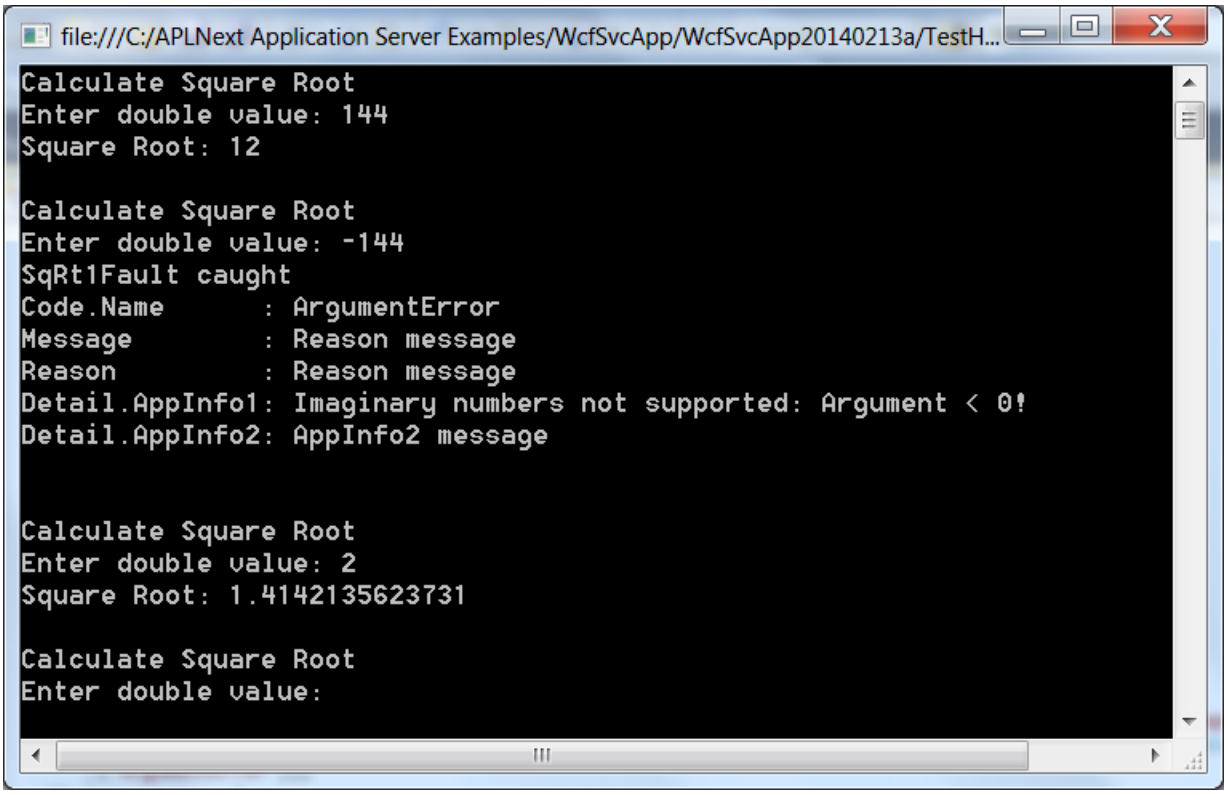
To view th WSDL document for the WCF web service which exposes the APL+Win-based web service ‘SqRt’ method, click the ‘svcutil.exe http://...’ link on the ‘Service1 Service’ web page.



## Test the WSDL interface layer

In Visual Studio use the 'Debug > Start Debugging' menu item to start up the 'TestHarness1' test client for the web service. Enter selected values into the 'Command Prompt' dialog presented by this 'Console Project'.

When an exception occurs, the exception information is reported to the client in an industry-standard manner.



## More WSDL Information

- Several web service features are possible to implement in an APL+Win-based web service, but the details of such implementations are beyond the scope of this document.
    - The web service client for the WSDL-enabled, APL+Win-based web service in this example illustrates how a Windows operating system-based client can be developed. Developing a web service client for other operating systems is possible using the WSDL interface layer.
    - This example illustrates synchronous web service calls, but it is possible to implement asynchronous web service calls for an APL+Win-based web service, including client feedback indicating processing status and performance.
    - It is possible to incorporate security into a web service at both the APLNext Application Server level and at the WCF level.
    - In this example for testing convenience the WCF web service supporting the WSDL interface layer and the APLNext Application Server supporting the APL+Win-based web service are deployed on the same workstation, however they may be deployed on different production servers if desired.
    - In this example web service scalability is not illustrated, but it may be implemented on the WCF and the APLNext Application server levels if desired.
    - Deploying the APL+Win-based web service and the associated WCF-based web service which provides the WSDL interface layer to a production server can be done using 'publishing' tools in Visual Studio.
- Wikipedia WSDL
- W3.org WSDL
- Microsoft WSDL Overview
- Contact APL2000 for assistance with a specific web services project.
- Detailed information about deploying the APLNext Application Server and creating an APL+Win-based web service is available on the APL2000 Forum for Web Services. A limited-time demonstration/evaluation version of APLNext Application Server or APLNext Desktop Server is available.