

home articles quick answers discussions features community help

Search for articles, questions, tips



Articles » Platforms, Frameworks & Libraries » Windows Communication Foundation » General

Next →

Article

Browse Code

Stats

Revisions (3)

Alternatives

Comments & Discussions (14)

A Beginner's Tutorial on How to Host a WCF Service (IIS Hosting and Self Hosting)

By **Rahul Rajat Singh**, 22 Feb 2013

★★★★★ 4.92 (20 votes)

Sign Up to vote

Tweet 28

+1 2

Like 24

[Download sample - 61.5 KB](#)

Introduction

In this article we will see how we can host a WCF service. We will look at various ways a WCF service can be hosted and what are the benefits and drawbacks of each hosting method. We will also create a sample client application that will consume the WCF service hosted in different ways.

Background

This article assumes that reader has some basic knowledge of WCF services. Please refer to the following article to get a refresher on WCF in case it is needed: [A Beginner's Tutorial for Understanding Windows Communication Foundation \(WCF\)](#)

In a service oriented architecture the standalone service is of no use unless it is exposed to the client. To expose a service to the client it needs to be hosted somewhere. By hosting we not only mean a server for hosting but also the application that will host the service as a part of itself.

Traditional web services were hosted on IIS. This approach has a lot of benefits which includes not worrying about the creation and disposal of service, On demand availability of web services etc. But this approach has one major drawback i.e. protocols other than HTTP are not supported.

WCF comes with the possibility of being invoked and used by protocols other than HTTP. So along with IIS, WCF can also be hosted in different ways so that its full power can be utilized(if required).

A WCF service can be hosted in following ways:

1. Hosting in Internet Information Services(IIS).
2. Hosting in Windows Activation Services(WAS).
3. Hosting in a Console or Desktop application(Self hosting).
4. Hosting in a Windows Service.

Every method of hosting comes with its own benefits and drawbacks. Let us see each the theory behind each hosting method one by one.

Hosting a WCF service in IIS is perhaps the easiest method. It is straight forward process for the developers who are familiar with **ASP.NET** websites and **ASMX** web services because the WCF service is also hosted in IIS in similar manner. Also, IIS takes care of creation of service instances, disposal of service instances, recycling and other activities that it does for ASP.NET website. It treats a WCF service like an **ASP.NET** website and provides all the features of having multiple request handling and dynamic compilation to a WCF service out of the box.

if we are using IIS6 or previous version then the major drawback of IIS6.0 hosting is that only **HTTP/HTTPS** protocol can be used to communicate with the service. Which actually is sufficient for most of the project but if the project needs to communicate to the service using other protocols then this IIS6.0 hosting will not suffice.

Now the point to notice in the above text is that I specifically mentioned IIS 6.0 or previous. The reason for this is that IIS 7.0 comes with a feature/component called **Windows Activation Service(WAS)**. So hosting a WCF service in WAS and IIS 7.0 are actually not exclusive. TO host a WCF service using WAS, we

About Article

In this article we will see how we can host a WCF service(IIS Hosting and Self Hosting).

Type **Article**

Licence **CPOL**

First Posted **21 Feb 2013**

Views **70,648**

Downloads **2,541**

Bookmarked **60 times**

C# ASP.NET WCF Beginn
SOA



Top News

[Man throws away trove of Bitcoin worth \\$7.5 million](#)

need to still host the service in IIS 7.0 and then enable the support for protocols other than HTTP/HTTPS. Hosting in WAS comes with all the benefits of hosting in IIS and on top of that it supports all the protocols too.

Now the next process of hosting i.e. self hosting has the major benefits of providing us full control i.e. All the control of starting the service, stopping the service and error handling/logging can be done in our host application. The amount of code that needs to be written to self host a WCF service is very small and it is very easy too. Now the decision for choosing self hosting is purely on the requirements of whether or not you want so much control on the service.

Finally, Hosting the service in a windows service is same as self hosting but the service creation and disposing will be abstracted in form of a Windows service. It provides the same capabilities as a self hosted service provides and the decision of whether to use this method or not will rely on the same factors of whether or not you want so much control on the service.

Using the code

Now that we know the various methods of hosting a WCF service, we can say that if we know how to host a WCF service in IIS, we can make some changes on top of that and host it in WAS too. Similarly, If we understand how to self host a service, we can create a windows service host with the same process.

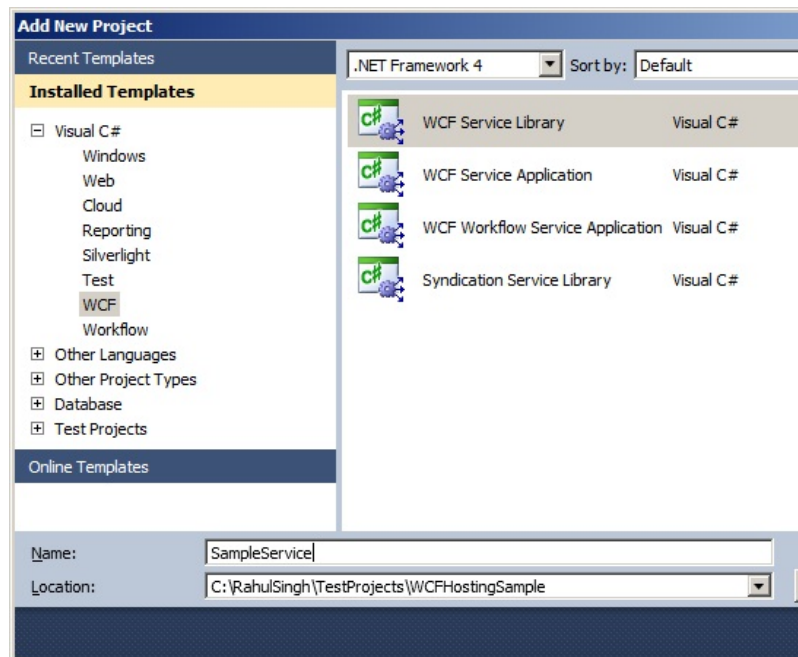
Now in the rest of the article, our focus will be to create a sample service and see how we can host it in IIS and how we can self host it. Then we will touch upon the steps that are needed to host the service on WAS and inside a windows service.

Note: Only IIS hosting and self hosting are discussed in detail in rest of the article because these are the most commonly used methods of hosting and other hosting process will be similar but they have some additional steps.

Creating the Service Library

We can always keep the service code in the host application (in case of self hosting) or inside an ASP.NET website(IIS hosting) but it is always a good idea to have the WCF service in a class library. That way we decouple the service implementation from its host application. Also, we can have a single WCF service and multiple host applications hosting that service.

Let us create a simple WCF service library that will take the users name in form of a **DataContract** and then return him a greeting message from the service. Let us start by creating a simple WCF Service library.



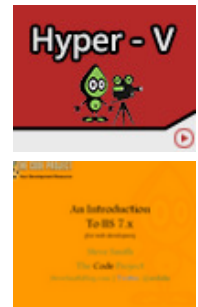
In this service we create a simple **DataContract** called Person which will keep the user Name.

```
[DataContract]
public class Person
{
    [DataMember]
    public string Name { get; set; }
}
```

Now we will have a very simple single method **ServiceContract** that will take the Person and return a string with containing the greeting message.

Get the [Insider News](#) free each morning.

Related Videos



Related Articles

[A Beginner's Tutorial for Understanding Windows Communication Foundation \(WCF\)](#)

[WCF Self Hosting with Example](#)

[A Beginner's Tutorial for Understanding WCF Instance Management](#)

[Create, Host \(Self Hosting, IIS hosting\) and Consume WCF Service](#)

[Windows Communication Foundation Basics](#)

[WCF: From a Beginner's perspective & a Tutorial](#)

[Windows Communication Foundation FAQ quick starter: Part 1](#)

[A custom ServiceHostFactory](#)

[What's the Difference between WCF and Web Services?](#)

[Beginner's Walk - Web Development](#)

[4 Simple Steps to Consume WCF Service using Silverlight](#)

[Windows Communication Foundation - Simple Example, Step by Step](#)

[Implementing a Basic Hello World WCF Service](#)

[WCF \(Windows Communication Foundation\) Introduction and Implementation](#)

[WCF - A Quick Review](#)

[WCF for Beginners](#)

[A Windows Communication Foundation \(WCF\) Overview](#)

[Calling Cross Domain WCF service using JQuery/JavaScript](#)

[Host your WCF service with multiple host environment using multiple protocol](#)

[WCF Service Library with Windows Service Hosting](#)

Related Research



```
[ServiceContract]
public interface ISampleService
{
    [OperationContract]
    string GreetMe(Person person);
}
```

How to Secure Your Software
for the Mobile Apps Market in
the US and Canada



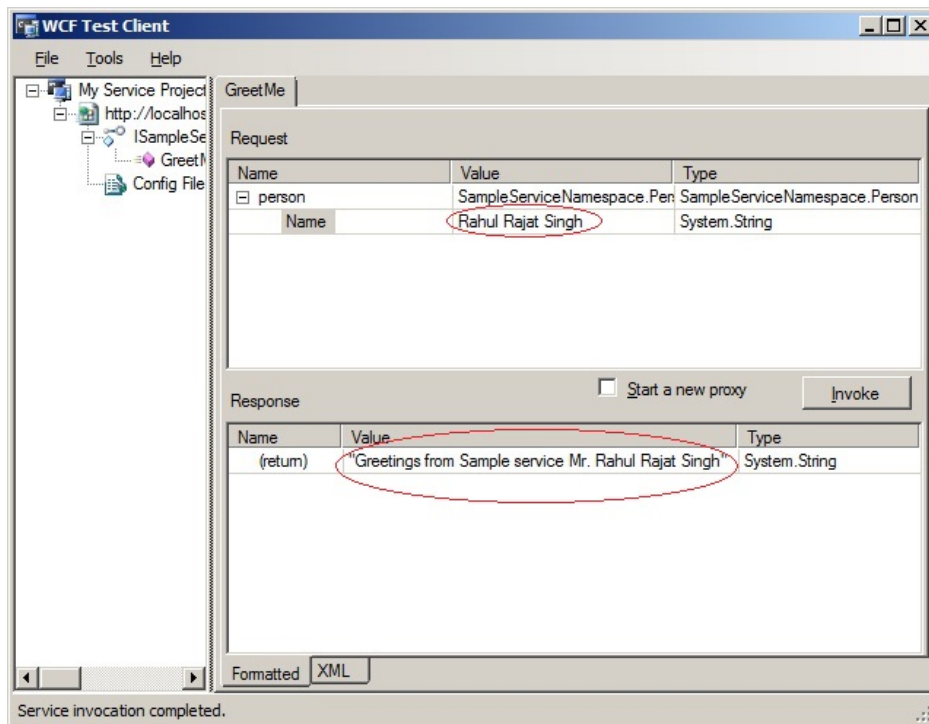
The Essential Guide to Mobile
App Testing: Tips for
Developers in USA & Canada

Finally, we will create the class that will implement the service contract.

[Collapse](#) | [Copy Code](#)

```
public class SampleService : ISampleService
{
    public string GreetMe(Person person)
    {
        return string.Format("Greetings from Sample service Mr. {0}", person.Name);
    }
}
```

Now we have a very simple toy service ready to play with. To check the working of the service we can simply run the WCF service library project. By doing this a WCF Test Client will open up using which we can test the working of our service. This tool can be used to test the calls to the service and response from the service.



Now one question we should ask is how this service started running without configuring any endpoints. Actually when we create a WCF service library some default configuration with **endpoints**, **address** and **Bindings** get created in the library's **App.config** file.

[Collapse](#) | [Copy Code](#)

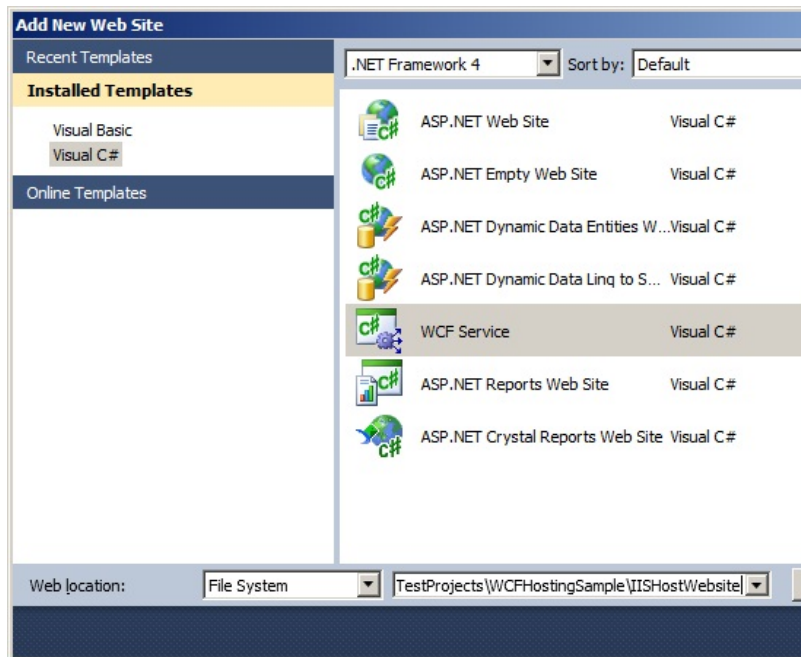
```
<service name="SampleServiceNamespace.SampleService">
  <endpoint address="" binding="wsHttpBinding"
    contract="SampleServiceNamespace.ISampleService">
  </endpoint>
```

Now this configuration will be used when we try to run a service library and Visual studio will take care of hosting this service for us for testing purpose. The important point here is that when we host this service ourselves then this **App.config** configuration will be of no use. The host application will have to define its own endpoints and specify the **Address**, **Binding** and **Contracts** separately.

Note: The configuration section shown above is not complete. Please refer to the sample code to see the complete **App.config** file.

Hosting the Service in IIS

Now that we have our service ready with us let us go and see how we can host this service in IIS. For this let us go and create a new website specifically WCF Service website.



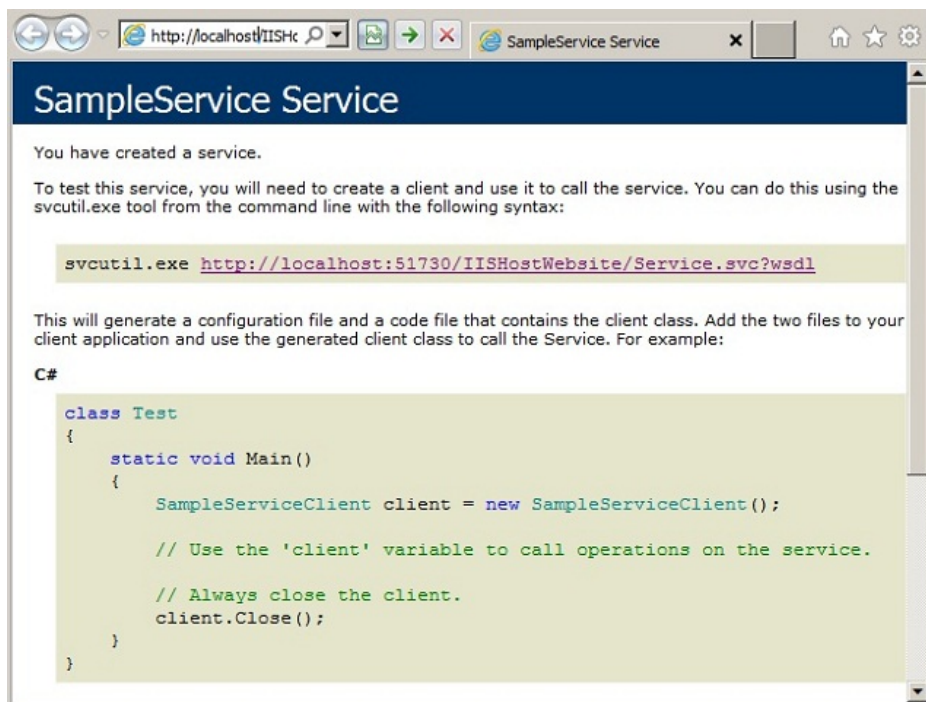
Now this website comes with its own predefined service template but we are not going to use them so we will delete all of them and then add a reference to our service library project in this website.

Once we have the reference to our service library added, its time to create a service and configure endpoints with required **address**, **binding** and **contracts**. Since we will host this website in the IIS, the **address** of the **.SVC** file will become the address of the Service. Now the **binding** used by default when we try to host a service in IIS is **basicHttpBinding**. And for the **contract** part, we need to specify the contract that should be used from our service library. This can be done by changing the **SVC** file and pointing the **SVC** file to the right service implementation.

[Collapse](#) | [Copy Code](#)

```
<%@ ServiceHost Language="C#" Debug="true" Service="SampleServiceNamespace.SampleService"%>
```

And this will suffice for the service to run within IIS. If we configure this website in IIS and run the **.svc** file the service will start running hosted in the IIS.



Now we have a service hosted on IIS and it is running. We will see how we can consume this IIS hosted service later in this article.

Self Hosting the Service

To self host the service let us create a simple Console application that will host the service, open the service and close the service. Let us start by having a simple console application.

Once the console application is created, let us add the reference to the Service library into this application.

Along with the reference of service library, we also have to add reference to **System.ServiceModel** and **System.Runtime.Serialization** in this project to get it to work with WCF hosting process.

Now we have the service library added in this project along with all other required references. Now we have to define the **endpoints**, **address**, **binding** and **contracts** for this service. This can be done either declarative (via configuration file) or imperatively(via code). Let us do it declarative by having the service configured in the **app.config** file.

[Collapse](#) | [Copy Code](#)

```
<services>
  <service name="SampleServiceNamespace.SampleService">
    <endpoint address="" binding="basicHttpBinding" bindingConfiguration=""
      contract="SampleServiceNamespace.ISampleService" />
    <host>
      <baseAddresses>
        <add baseAddress="http://localhost/SelfHostedServiceConsole" />
      </baseAddresses>
    </host>
  </service>
</services>
```

We configured this service to use **HTTP** protocol with **basicHttpBinding**. The address of the service is **http://localhost/SelfHostedServiceConsole** and the contract is specified as the service contract from our service library i.e. **SampleServiceNamespace.ISampleService**

Note: I really recommend using the "**Edit WCF Configuration Tool**" that comes with Visual studio(right click the app.config file). This really makes the declarative configuration of WCF service very easy.

Now that we have specified the **ABC** of this service and exposed an endpoint, its time to see how we can host the service. To host the service and open the service host we need to use the **ServiceHost** class. below code shows how the service can be hosted using the endpoints defined in the above config file.

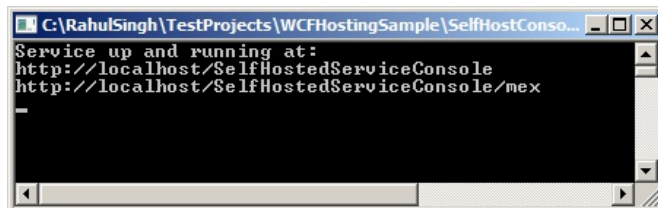
[Collapse](#) | [Copy Code](#)

```
using (ServiceHost host = new ServiceHost(typeof(SampleServiceNamespace.SampleService)))
{
    host.Open();

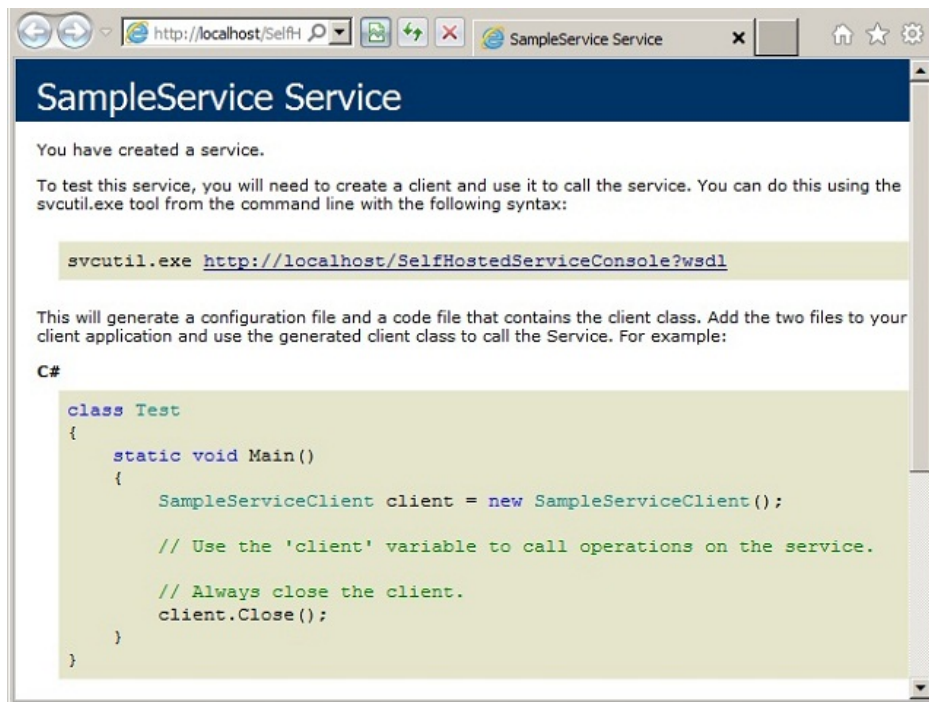
    Console.WriteLine("Service up and running at:");
    foreach (var ea in host.Description.Endpoints)
    {
        Console.WriteLine(ea.Address);
    }

    Console.ReadLine();
    host.Close();
}
```

And when we run the host:



To check the self hosted service, Let us start the host and then type the address in the browser.

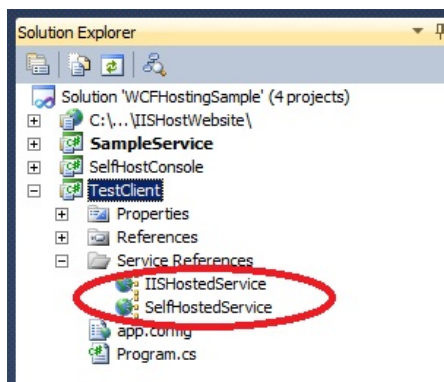


And this confirms that the service has been hosted successfully.

Consuming the WCF service

Let us now go ahead and create a simple Console application to test these two services. First let us create a simple console application. Once we have the console application created, let us add two service reference to it. One for the IIS hosted service and other for the self hosted service.

Note: To add the IIS hosted service reference, use the address of the **.svc** file and to add the reference of self hosted service, use the address specified in the config file of self hosting console. Also, make sure that the hosting console application is running while adding the service reference.



Once we add the service reference the client side endpoints have already been configured in the app.config file. For now we will use this configurations only. Adding the service reference also created the proxy classes for the service. So let us see how we can use these proxy classes to call our IIS hosted and Self hosted WCF service.

[Collapse](#) | [Copy Code](#)

```
static void Main(string[] args)
{
    // Using the IIS hosted service
    Console.WriteLine("Using the IIS hosted service");
    using (IISHostedService.SampleServiceClient client = new
    IISHostedService.SampleServiceClient())
    {
        IISHostedService.Person p = new IISHostedService.Person { Name = "Rahul" };
        Console.WriteLine(client.GreetMe(p));
    }

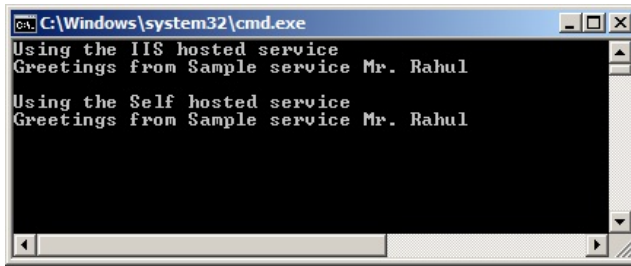
    Console.WriteLine();

    // Using the Self hosted service
    Console.WriteLine("Using the Self hosted service");
    using (SelfHostedService.SampleServiceClient client = new
    SelfHostedService.SampleServiceClient())
    {
        SelfHostedService.Person p = new SelfHostedService.Person { Name = "Rahul" };
        Console.WriteLine(client.GreetMe(p));
    }

    Console.ReadLine();
}
```

```
}
```

And when we run the application



```
C:\Windows\system32\cmd.exe
Using the IIS hosted service
Greetings from Sample service Mr. Rahul

Using the Self hosted service
Greetings from Sample service Mr. Rahul
```

And this shows us that both our IIS hosted and self hosted services are running.

Point of interest

In this article, we started our discussion with the various possible ways of hosting a WCF service. We saw how to host a service in IIS and how to self host a service. We have not touched the WAS hosting and Hosting inside a Windows service but these processes will require something additional to be done on top of IIS hosting and self hosting respectively. Since WAS and Windows service knowledge will automatically let the user know how and what additional needs to be done, I skipped these two methods from this article (otherwise the article would have become digressing).

Also, I used basic HTTP binding and default behaviors for all demos because the intention of the article was to talk about hosting the WCF service. But these things can easily be tweaked in the hosts and clients (proved they conform to hosting environments capabilities). This article was written from a perspective of beginner's in WCF. I hope this has been informative.

History

- **22 February 2013:** First version.


License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)

About the Author



Rahul Rajat Singh

Software Developer (Senior)
India 

I Started my Programming career with C++. Later got a chance to develop Windows Form applications using C#. Currently using C#, ASP.NET & ASP.NET MVC to create Information Systems, e-commerce/e-governance Portals and Data driven websites.

My interests involves Programming, Website development and Learning/Teaching subjects related to Computer Science/Information Systems. IMO, C# is the best programming language and I love working with C# and other Microsoft Technologies.

Follow on  Twitter  Google  LinkedIn

[Article Top](#)

Comments and Discussions

You must [Sign In](#) to use this message board.

Search this forum

Go

☒ Profile popups

Spacing

Relaxed

Noise

Very High

Layout

Normal

Per page

10

Update

First

Prev

Next

This was very helpful	TomP69	18-Nov-13 9:48
WCF Service Application deployment	Ganesh Satpute	31-Oct-13 22:22
Nice	Bhuvanesh Mohankumar	29-Oct-13 18:45
Client from another Computer in the network	XcyTheR	28-Oct-13 6:02
Register WCF Service with IIS and ASP Dot Net	Ravi.Kumar021	30-Sep-13 18:25
My vote of 5	Nandakishorerao	5-Aug-13 1:30
If I use VS2012 how to host a WCF service in IIS7.0?	halower	7-Jul-13 4:53
Server Error in '/' Application	wShaiza	19-Jun-13 20:52
Re: Server Error in '/' Application	Bhuvanesh Mohankumar	29-Oct-13 19:20
My vote of 5	are there any names not used ?	30-May-13 23:46

Last Visit: 31-Dec-99 18:00

Last Update: 3-Dec-13 4:19

Refresh

1 2 Next »

General News Suggestion Question Bug Answer Joke Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.