How to: Host a WCF Service in a Managed Application

.NET Framework 4

To host a service inside a managed application, embed the code for the service inside the managed application code, define an endpoint for the service either imperatively in code, declaratively through configuration, or using default endpoints, and then create an instance of ServiceHost[1].

To start receiving messages, call Open[2] on **ServiceHost**. This creates and opens the listener for the service. Hosting a service in this way is often referred to as "self-hosting" because the managed application is doing the hosting work itself. To close the service, call System.ServiceModel.Channels.CommunicationObject.Close[3] on **ServiceHost**.

A service can also be hosted in a managed Windows service, in Internet Information Services (IIS), or in Windows Process Activation Service (WAS). For more information about hosting options for a service, see Hosting Services[4].

Hosting a service in a managed application is the most flexible option because it requires the least infrastructure to deploy. For more information about hosting services in managed applications, see Hosting in a Managed Application[5].

The following procedure demonstrates how to implement a self-hosted service in a console application.

To create a self-hosted service

1. Open Visual Studio 2010 and select **New**, **Project...** from the **File** menu.
2. In the **Installed Templates** list, select **Visual C#**, **Windows** or **Visual Basic**, **Windows**. Depending on your Visual Studio 2010 settings, one or both of these may be under the **Other Languages** node in the **Installed Templates** list.
3. Select **Console Application** from the **Windows** list. Type `SelfHost` in the **Name** box and click **OK**.
4. Right-click **SelfHost** in **Solution Explorer** and select **Add Reference...**. Select **System.ServiceModel** from the **.NET** tab and click **OK**.

   Tip:

   If the **Solution Explorer** window is not visible, select **Solution Explorer** from the **View** menu.

5. Double-click **Program.cs** or **Module1.vb** in **Solution Explorer** to open it in the code window if it is not already open. Add the following statements at the top of the file.

   C#

```
using System.ServiceModel;
using System.ServiceModel.Description;
```

6. Define and implement a service contract. This example defines a `HelloWorldService` that returns a message based on the input to the service.

C#

```
[ServiceContract]
public interface IHelloWorldService
{
    [OperationContract]
    string SayHello(string name);
}

public class HelloWorldService : IHelloWorldService
{
    public string SayHello(string name)
    {
        return string.Format("Hello, {0}", name);
    }
}
```

Note:

For more information about how to define and implement a service interface, see How to: Define a Windows Communication Foundation Service Contract[6] and How to: Implement a Windows Communication Foundation Service Contract[7].

7. At the top of the `Main` method, create an instance of the Uri[8] class with the base address for the service.

C#

```
Uri baseAddress = new Uri("http://localhost:8080/hello");
```

8. Create an instance of the **ServiceHost** class, passing a Type[9] that represents the service type and the base address Uniform Resource Identifier (URI) to the ServiceHost. Enable metadata publishing, and then call the **Open** method on the **ServiceHost** to initialize the service and prepare it to receive messages.

C#

```
// Create the ServiceHost.
using (ServiceHost host = new ServiceHost(typeof(HelloWorldService), baseAddress))
{
    // Enable metadata publishing.
    ServiceMetadataBehavior smb = new ServiceMetadataBehavior();
    smb.HttpGetEnabled = true;
    smb.MetadataExporter.PolicyVersion = PolicyVersion.Policy15;
    host.Description.Behaviors.Add(smb);

    // Open the ServiceHost to start listening for messages. Since
    // no endpoints are explicitly configured, the runtime will create
    // one endpoint per base address for each service contract implemented
    // by the service.
    host.Open();

    Console.WriteLine("The service is ready at {0}", baseAddress);
    Console.WriteLine("Press <Enter> to stop the service.");
```

```
        Console.ReadLine();

        // Close the ServiceHost.
        host.Close();
    }
```

> **Note:**
>
> This example uses default endpoints, and no configuration file is required for this service. If no endpoints are configured, then the runtime creates one endpoint for each base address for each service contract implemented by the service. For more information about default endpoints, see Simplified Configuration[10] and Simplified Configuration for WCF Services[11].

9. Press CTRL+SHIFT+B to build the solution.

To test the service

1. Press Ctrl + F5 to run the service.
2. Open **WCF Test Client**.

> **Tip:**
>
> To open **WCF Test Client**, open a Visual Studio 2010 command prompt and execute **WcfTestClient.exe**.

3. Select **Add Service...** from the **File** menu.
4. Type `http://localhost:8080/hello` into the address box and click **OK**.

> **Tip:**
>
> Make sure the service is running or else this step fails. If you have changed the base address in the code, then use the modified base address in this step.

5. Double-click **SayHello** under the **My Service Projects** node. Type your name into the **Value** column in the **Request** list, and click **Invoke**. A reply message appears in the **Response** list.

Example

The following example creates a **ServiceHost** object to host a service of type `HelloWorldService`, and then calls the **Open** method on **ServiceHost**. A base address is provided in code, metadata publishing is enabled, and default endpoints are used.

C#

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.ServiceModel;
using System.ServiceModel.Description;

namespace SelfHost
{
    [ServiceContract]
    public interface IHelloWorldService
    {
        [OperationContract]
        string SayHello(string name);
    }

    public class HelloWorldService : IHelloWorldService
    {
        public string SayHello(string name)
        {
            return string.Format("Hello, {0}", name);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Uri baseAddress = new Uri("http://localhost:8080/hello");

            // Create the ServiceHost.
            using (ServiceHost host = new ServiceHost(typeof(HelloWorldService), baseAd
dress))
            {
                // Enable metadata publishing.
                ServiceMetadataBehavior smb = new ServiceMetadataBehavior();
                smb.HttpGetEnabled = true;
                smb.MetadataExporter.PolicyVersion = PolicyVersion.Policy15;
                host.Description.Behaviors.Add(smb);

                // Open the ServiceHost to start listening for messages. Since
                // no endpoints are explicitly configured, the runtime will create
                // one endpoint per base address for each service contract implemented
                // by the service.
                host.Open();

                Console.WriteLine("The service is ready at {0}", baseAddress);
                Console.WriteLine("Press <Enter> to stop the service.");
                Console.ReadLine();

                // Close the ServiceHost.
                host.Close();
            }
        }
    }
}
```

See Also

Tasks

How to: Host a WCF Service in IIS[12]
Self-Host[13]
How to: Define a Windows Communication Foundation Service Contract[6]
How to: Implement a Windows Communication Foundation Service Contract[7]

Reference

Uri[8]
AppSettings[14]
ConfigurationManager[15]

Concepts

Hosting Services[4]
ServiceModel Metadata Utility Tool (Svcutil.exe)[16]
Using Bindings to Configure Services and Clients[17]
System-Provided Bindings[18]

Links Table

[1]*http://msdn.microsoft.com/en-us/library/system.servicemodel.servicehost.aspx*

[2]*http://msdn.microsoft.com/en-us/library/ms195524.aspx*

[3]*http://msdn.microsoft.com/en-us/library/system.servicemodel.channels.communicationobject.close.aspx*

[4]*http://msdn.microsoft.com/en-us/library/ms730158.aspx*

[5]*http://msdn.microsoft.com/en-us/library/ms733765.aspx*

[6]*http://msdn.microsoft.com/en-us/library/ms731835.aspx*

[7]*http://msdn.microsoft.com/en-us/library/ms734686.aspx*

[8]*http://msdn.microsoft.com/en-us/library/system.uri.aspx*

[9]*http://msdn.microsoft.com/en-us/library/system.type.aspx*

[10]*http://msdn.microsoft.com/en-us/library/ee358768.aspx*

[11]*http://msdn.microsoft.com/en-us/library/ee530014.aspx*

[12]*http://msdn.microsoft.com/en-us/library/ms733766.aspx*

[13]*http://msdn.microsoft.com/en-us/library/ms750530.aspx*

[14]*http://msdn.microsoft.com/en-us/library/system.configuration.configurationmanager.appsettings.aspx*

[15]*http://msdn.microsoft.com/en-us/library/system.configuration.configurationmanager.aspx*

[16]*http://msdn.microsoft.com/en-us/library/aa347733.aspx*

[17]*http://msdn.microsoft.com/en-us/library/ms733865.aspx*

[18]*http://msdn.microsoft.com/en-us/library/ms730879.aspx*

Community Content

Default Endpoint Creation didn't work for me, so this fixed it.

In the sample, following the Behaviors.Add method call, I created the Endpoint myself with the following code. $0$0 $0 $0// Add MEX endpoint$0 $0host.AddServiceEndpoint (ServiceMetadataBehavior.MexContractName, MetadataExchangeBindings.CreateMexHttpBinding(),"mex");$0 $0// Add application endpoint$0 $0host.AddServiceEndpoint(typeof(I-YOUR-INTERFACE-GOES-HERE), new WSHttpBinding(), "");$0

4/6/2011 Chuck Miller  4/6/2011 Chuck Miller 

AddressAccessDeniedException

This will generate an System.ServiceModel.AddressAccessDeniedException unless you follow the steps in http://go.microsoft.com/fwlink/?LinkId=70353 to reserve the address of your service.

8/19/2010 Andy Pennell MSFT  8/19/2010 Andy Pennell MSFT 

Don’t Wrap Wcf Service Hosts or Clients in a Using Statement.

Don’t Wrap Wcf Service Hosts or Clients in a Using Statement. http://www.danrigsby.com/blog/index.php/2008/02/26/dont-wrap-wcf-service-hosts-or-clients-in-a-using-statement/

7/2/2010 aluty