**Endpoint Creation Overview**

All communication with a Windows Communication Foundation (WCF) service occurs through the *endpoints* of the service. Endpoints provide the clients access to the functionality that a WCF service offers. This section describes the structure of an endpoint and outlines how to define an endpoint in configuration and in code.

**The Structure of an Endpoint**

Each endpoint contains an address that indicates where to find the endpoint, a binding that specifies how a client can communicate with the endpoint, and a contract that identifies the methods available.

* **Address**. The address uniquely identifies the endpoint and tells potential consumers where the service is located. It is represented in the WCF object model by the [EndpointAddress](http://msdn.microsoft.com/en-us/library/system.servicemodel.endpointaddress.aspx) address, which contains a Uniform Resource Identifier (URI) and address properties that include an identity, some Web Services Description Language (WSDL) elements, and a collection of optional headers. The optional headers provide additional detailed addressing information to identify or interact with the endpoint. For more information, see [Specifying an Endpoint Address](http://msdn.microsoft.com/en-us/library/ms733749.aspx).
* **Binding**. The binding specifies how to communicate with the endpoint. The binding specifies how the endpoint communicates with the world, including which transport protocol to use (for example, TCP or HTTP), which encoding to use for the messages (for example, text or binary), and which security requirements are necessary (for example, Secure Sockets Layer [SSL] or SOAP message security). For more information, see [Using Bindings to Configure Services and Clients](http://msdn.microsoft.com/en-us/library/ms733865.aspx).
* **Service contract**. The service contract outlines what functionality the endpoint exposes to the client. A contract specifies the operations that a client can call, the form of the message and the type of input parameters or data required to call the operation, and the kind of processing or response message the client can expect. Three basic types of contracts correspond to basic message exchange patterns (MEPs): datagram (one-way), request/reply, and duplex (bidirectional). The service contract can also employ data and message contracts to require specific data types and message formats when being accessed. For more information about how to define a service contract, see [Designing Service Contracts](http://msdn.microsoft.com/en-us/library/ms733070.aspx). Note that a client may also be required to implement a service-defined contract, called a callback contract, to receive messages from the service in a duplex MEP. For more information, see [Duplex Services](http://msdn.microsoft.com/en-us/library/ms731064.aspx).

The endpoint for a service can be specified either imperatively by using code or declaratively through configuration. If no endpoints are specified then the runtime provides default endpoints by adding one default endpoint for each base address for each service contract implemented by the service. Defining endpoints in code is usually not practical because the bindings and addresses for a deployed service are typically different from those used while the service is being developed. Generally, it is more practical to define service endpoints using configuration rather than code. Keeping the binding and addressing information out of the code allows them to change without having to recompile and redeploy the application.

|  |
| --- |
| **ms735096.note(en-us,VS.100).gifNote:** |
| When adding a service endpoint that performs impersonation, you must either use one of the [AddServiceEndpoint](http://msdn.microsoft.com/en-us/library/system.servicemodel.servicehost.addserviceendpoint.aspx) methods or the [GetContract](http://msdn.microsoft.com/en-us/library/ms576728.aspx) method to properly load the contract into a new [ServiceDescription](http://msdn.microsoft.com/en-us/library/system.servicemodel.description.servicedescription.aspx) object. |

**Defining Endpoints in Code**

The following example illustrates how to specify an endpoint in code with the following:

* Define a contract for an **IEcho** type of service that accepts someone's name and echo with the response "Hello <name>!".
* Implement an **Echo** service of the type defined by the **IEcho** contract.
* Specify an endpoint address of http://localhost:8000/Echo for the service.
* Configure the **Echo** service using a [WSHttpBinding](http://msdn.microsoft.com/en-us/library/system.servicemodel.wshttpbinding.aspx) binding.

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl11_code');" \o "Copy Code)

Namespace Echo

{

// Define the contract for the IEcho service

[ServiceContract]

Interface IEcho

{

[OperationContract]

String Hello(string name)

}

// Create an Echo service that implements IEcho contract

Class Echo : IEcho

{

Public string Hello(string name)

{

return "Hello" + name + "!";

}

static void Main ()

{

//Specify the base address for Echo service.

Uri echoUri = new Uri("http://localhost:8000/");

//Create a ServiceHost for the Echo service.

ServiceHost serviceHost = new ServiceHost(typeof(Echo),echoUri);

// Use a predefined WSHttpBinding to configure the service.

WSHttpBinding binding = new WSHttpBinding();

// Add the endpoint for this service to the service host.

serviceHost.AddServiceEndpoint(

typeof(IEcho),

binding,

echoUri

);

// Open the service host to run it.

serviceHost.Open();

}

}

}

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl12_code');" \o "Copy Code)

' Define the contract for the IEcho service

<ServiceContract()> \_

Public Interface IEcho

<OperationContract()> \_

Function Hello(ByVal name As String) As String

End Interface

' Create an Echo service that implements IEcho contract

Public Class Echo

Implements IEcho

Public Function Hello(ByVal name As String) As String \_

Implements ICalculator.Hello

Dim result As String = "Hello" + name + "!"

Return result

End Function

' Specify the base address for Echo service.

Dim echoUri As Uri = New Uri("http://localhost:8000/")

' Create a ServiceHost for the Echo service.

Dim svcHost As ServiceHost = New ServiceHost(GetType(HelloWorld), echoUri)

' Use a predefined WSHttpBinding to configure the service.

Dim binding As New WSHttpBinding()

' Add the endpoint for this service to the service host.

serviceHost.AddServiceEndpoint(GetType(IEcho), binding, echoUri)

' Open the service host to run it.

serviceHost.Open()

|  |
| --- |
| **ms735096.note(en-us,VS.100).gifNote:** |
| The service host is created with a base address and then the rest of the address, relative to the base address, is specified as part of an endpoint. This partitioning of the address allows multiple endpoints to be defined more conveniently for services at a host. |
| **ms735096.note(en-us,VS.100).gifNote:** |
| Properties of **ServiceDescription** in the service application must not be modified subsequent to the [OnOpening](http://msdn.microsoft.com/en-us/library/system.servicemodel.channels.communicationobject.onopening.aspx) method on [ServiceHostBase](http://msdn.microsoft.com/en-us/library/system.servicemodel.servicehostbase.aspx). Some members, such as the [Credentials](http://msdn.microsoft.com/en-us/library/system.servicemodel.servicehostbase.credentials.aspx) property and the **AddServiceEndpoint** methods on **ServiceHostBase** and [ServiceHost](http://msdn.microsoft.com/en-us/library/system.servicemodel.servicehost.aspx), throw an exception if modified past that point. Others permit you to modify them, but the result is undefined.  Similarly, on the client the [ServiceEndpoint](http://msdn.microsoft.com/en-us/library/system.servicemodel.description.serviceendpoint.aspx) values must not be modified after the call to **OnOpening** on the [ChannelFactory](http://msdn.microsoft.com/en-us/library/system.servicemodel.channelfactory.aspx). The [Credentials](http://msdn.microsoft.com/en-us/library/system.servicemodel.channelfactory.credentials.aspx) property throws an exception if modified past that point. The other client description values can be modified without error, but the result is undefined.  Whether for the service or client, it is recommended that you modify the description prior to calling [Open](http://msdn.microsoft.com/en-us/library/ms405515.aspx). |

**Defining Endpoints in Configuration**

When creating an application, you often want to defer decisions to the administrator who is deploying the application. For example, there is often no way of knowing in advance what a service address (a URI) will be. Instead of hard-coding an address, it is preferable to allow an administrator to do so after creating a service. This flexibility is accomplished through configuration. For details, see [Configuring Services](http://msdn.microsoft.com/en-us/library/ms733830.aspx).

|  |
| --- |
| **ms735096.note(en-us,VS.100).gifNote:** |
| Use the [ServiceModel Metadata Utility Tool (Svcutil.exe)](http://msdn.microsoft.com/en-us/library/aa347733.aspx) with the **/config:***filename[,filename]* switch to quickly create configuration files. |

**Using Default Endpoints**

If no endpoints are specified in code or in configuration then the runtime provides default endpoints by adding one default endpoint for each base address for each service contract implemented by the service. The base address can be specified in code or in configuration, and the default endpoints are added when Open is called on the **ServiceHost**. This example is the same example from the previous section, but since no endpoints are specified, the default endpoints are added.

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl27_code');" \o "Copy Code)

Namespace Echo

{

// Define the contract for the IEcho service

[ServiceContract]

Interface IEcho

{

[OperationContract]

String Hello(string name)

}

// Create an Echo service that implements IEcho contract

Class Echo : IEcho

{

Public string Hello(string name)

{

return "Hello" + name + "!";

}

static void Main ()

{

//Specify the base address for Echo service.

Uri echoUri = new Uri("http://localhost:8000/");

//Create a ServiceHost for the Echo service.

ServiceHost serviceHost = new ServiceHost(typeof(Echo),echoUri);

// Open the service host to run it. Default endpoints

// are added when the service is opened.

serviceHost.Open();

}

}

}

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl28_code');" \o "Copy Code)

' Define the contract for the IEcho service

<ServiceContract()> \_

Public Interface IEcho

<OperationContract()> \_

Function Hello(ByVal name As String) As String

End Interface

' Create an Echo service that implements IEcho contract

Public Class Echo

Implements IEcho

Public Function Hello(ByVal name As String) As String \_

Implements ICalculator.Hello

Dim result As String = "Hello" + name + "!"

Return result

End Function

' Specify the base address for Echo service.

Dim echoUri As Uri = New Uri("http://localhost:8000/")

' Open the service host to run it. Default endpoints

' are added when the service is opened.

serviceHost.Open()

If endpoints are explicitly provided, the default endpoints can still be added by calling [AddDefaultEndpoints](http://msdn.microsoft.com/en-us/library/system.servicemodel.servicehostbase.adddefaultendpoints.aspx) on the **ServiceHost** before calling **Open**. For more information about default endpoints, see [Simplified Configuration](http://msdn.microsoft.com/en-us/library/ee358768.aspx) and [Simplified Configuration for WCF Services](http://msdn.microsoft.com/en-us/library/ee530014.aspx).