代码形式发布端点时，可以是多个绑定形式，

服务端代码如下：

ServiceHost host = null;

//启动

private void button1\_Click(object sender, EventArgs e)

{

//地址

Uri pipeaddress = new Uri("net.pipe://localhost/NetNamedPipeBinding");

Uri tcpaddress = new Uri("net.tcp://localhost:8088/TcpBinding");

//服务宿主对象

host = new ServiceHost(typeof(WcfServiceLibrary1.Service1), pipeaddress, tcpaddress);

//绑定

NetNamedPipeBinding pb = new NetNamedPipeBinding();

NetTcpBinding tp = new NetTcpBinding();

//添加元数据行为

ServiceMetadataBehavior mBehavior = new ServiceMetadataBehavior();

host.Description.Behaviors.Add(mBehavior);

//添加元数据端点

host.AddServiceEndpoint(typeof(IMetadataExchange), MetadataExchangeBindings.CreateMexNamedPipeBinding(), "mex");

host.AddServiceEndpoint(typeof(IMetadataExchange), MetadataExchangeBindings.CreateMexTcpBinding(), "mex");

//添加服务端点

host.AddServiceEndpoint(typeof(WcfServiceLibrary1.IService1), pb, pipeaddress);

host.AddServiceEndpoint(typeof(WcfServiceLibrary1.IService1), tp, tcpaddress);

host.Opened += delegate { lbMsg.Text = "服务已启动！"; };

host.Open();

}

//停止

private void button2\_Click(object sender, EventArgs e)

{

if (host.State != CommunicationState.Closed)

{

host.Close();

}

lbMsg.Text = "服务已停止!";

}

服务端，也可以使用配置文件的形式发布服务：

代码如下：

<system.serviceModel>

<services>

<service name="WcfServiceLibrary1.Service1" behaviorConfiguration="textBehavior">

<host>

<baseAddresses>

<add baseAddress="net.tcp://localhost:8088/tcpBinding"/>

<add baseAddress="http://localhost:3200/httpBinding"/>

<add baseAddress="net.pipe://localhost/pipeBinding"/>

</baseAddresses>

</host>

<endpoint address="tcpmex" binding="mexTcpBinding" contract="IMetaExchange"></endpoint>

<endpoint address="pipemex" binding="mexNamedPipeBinding" contract="IMetaExchange"></endpoint>

<endpoint address="" binding="wsHttpBinding" contract="WcfServiceLibrary1.IService1"></endpoint>

<endpoint address="" binding="netTcpBinding" contract="WcfServiceLibrary1.IService1"></endpoint>

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="textBehavior">

<serviceMetadata/>

<serviceDebug/>

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

客户端代码如下：

private void button1\_Click(object sender, EventArgs e)

{

if (radioButton1.Checked)

{

//namedpipe

NetNamedPipeBinding np = new NetNamedPipeBinding();

EndpointAddress address = new EndpointAddress("net.pipe://localhost/NetNamedPipeBinding");

ChannelFactory<WcfServiceLibrary1.IService1> factory = new ChannelFactory<WcfServiceLibrary1.IService1>(np,address);

WcfServiceLibrary1.IService1 pipeChannel = factory.CreateChannel();

string s = pipeChannel.GetData(3);

MessageBox.Show(s);

factory.Close();

}

else

{

//tcp

NetTcpBinding tp = new NetTcpBinding();

EndpointAddress address = new EndpointAddress("net.tcp://localhost:8088/TcpBinding");

ChannelFactory<WcfServiceLibrary1.IService1> factory = new ChannelFactory<WcfServiceLibrary1.IService1>(tp, address);

WcfServiceLibrary1.IService1 tcpChannel = factory.CreateChannel();

string s = tcpChannel.GetData(4);

MessageBox.Show(s);

factory.Close();

}

}

}

注意，客户端也可以使用配置文件的方式调用

<system.serviceModel>

<bindings>

<netNamedPipeBinding>

<binding name="NetNamedPipeBinding\_IService1" closeTimeout="00:01:00"

openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"

transactionFlow="false" transferMode="Buffered" transactionProtocol="OleTransactions"

hostNameComparisonMode="StrongWildcard" maxBufferPoolSize="524288"

maxBufferSize="65536" maxConnections="10" maxReceivedMessageSize="65536">

<readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"

maxBytesPerRead="4096" maxNameTableCharCount="16384" />

<security mode="Transport">

<transport protectionLevel="EncryptAndSign" />

</security>

</binding>

</netNamedPipeBinding>

<netTcpBinding>

<binding name="NetTcpBinding\_IService1" closeTimeout="00:01:00"

openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"

transactionFlow="false" transferMode="Buffered" transactionProtocol="OleTransactions"

hostNameComparisonMode="StrongWildcard" listenBacklog="10"

maxBufferPoolSize="524288" maxBufferSize="65536" maxConnections="10"

maxReceivedMessageSize="65536">

<readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"

maxBytesPerRead="4096" maxNameTableCharCount="16384" />

<reliableSession ordered="true" inactivityTimeout="00:10:00"

enabled="false" />

<security mode="Transport">

<transport clientCredentialType="Windows" protectionLevel="EncryptAndSign" />

<message clientCredentialType="Windows" />

</security>

</binding>

</netTcpBinding>

</bindings>

<client>

<endpoint address="net.pipe://localhost/NetNamedPipeBinding"

binding="netNamedPipeBinding" bindingConfiguration="NetNamedPipeBinding\_IService1"

contract="server.IService1" name="NetNamedPipeBinding\_IService1">

<identity>

<userPrincipalName value="XP-201104191101\Administrator" />

</identity>

</endpoint>

<endpoint address="net.tcp://localhost:8088/TcpBinding" binding="netTcpBinding"

bindingConfiguration="NetTcpBinding\_IService1" contract="server.IService1"

name="NetTcpBinding\_IService1">

<identity>

<userPrincipalName value="XP-201104191101\Administrator" />

</identity>

</endpoint>

</client>

</system.serviceModel>

**服务行为**

* 服务行为可以修改和控制WCF服务的运行特性。在实现了WCF服务契约后，可以修改服务的很多执行特性。这些行为（或者特性）是通过配置运行时属性或者通过自定义行为来控制的。
* **WCF的行为分为两类：**
* 服务行为（Service Behavior）
* 操作行为（Operation Behavior）

**行为应用到实现接口的类上**

服务行为（Service Behavior）  
操作行为（Operation Behavior）

[ServiceBehavior]

public class ServiceClass:IService1

{

[OperationBehavior]

public int AddNumber(int x,int y)

{

return x+y;

}

[OperationBehavior]

public int SubtractNumber(int x,int y)

{

return x-y;

}

}

**[ServiceBehavior]属性**

* AutomaticSessionShutdown :用户指定客户端关闭输出会话时，服务端会话会自动关闭
* ConcurrencyMode ：对服务线程的支持：
* IgnoreExtensionDataObject :没有序列化的数据是否可以进行通讯
* IncludeExceptionDetaillnFaults ：如何处理未被处理的异常
* InstanceContextMode ：每次会话是否创建新的实例，且不能共享，是否可以回收，
* ReleaseServiceInstanceOnTransactionComplete
* TransactionAutoCompleteOnSessionClose
* TransactionlsolationLevel
* TtansactionTimeout

**[OperationBehavior]属性**

* AutoDisposeParameters 指定关联方法的参数是否自动销毁
* ReleaseInstanceMode 操作前后是否回收
* TransactionAutoComplete 指定是否自动提交事务

**使用配置指定行为**

<serviceBehaviors>

<behavior name="WcfService1.Service1Behavior">

<serviceAuthorization ></serviceAuthorization>

<serviceTimeouts/>

<serviceThrottling maxConcurrentCalls="" maxConcurrentInstances="" maxConcurrentSessions=""/>

<serviceMetadata httpGetEnabled="true"/>

<serviceDebug includeExceptionDetailInFaults="false"/>

</behavior>

</serviceBehaviors>

**异常处理**

* FaultException
* TimeoutException
* CommunicationException

服务端：

public void Divide(int x, int y)

{

try

{

int z = x / y;

}

catch(Exception e)

{

throw new FaultException("试图除零",new FaultCode("除法操作"));

}

}

客户端：

private void button1\_Click(object sender, EventArgs e)

{

server.Service1Client client = new wcfClient.server.Service1Client();

try

{

client.Divide(3, 2);

client.Abort();

client.Divide(3, 2);

}

catch (FaultException fe)

{

MessageBox.Show(fe.Reason.ToString());

}

catch (TimeoutException te)

{

MessageBox.Show("调用超时");

}

catch (CommunicationException ce)

{

MessageBox.Show("通信异常!");

}

}