* **传输安全**
  + 保证信息在传输过程中的安全.传输安全是身份验证和授权的前提.

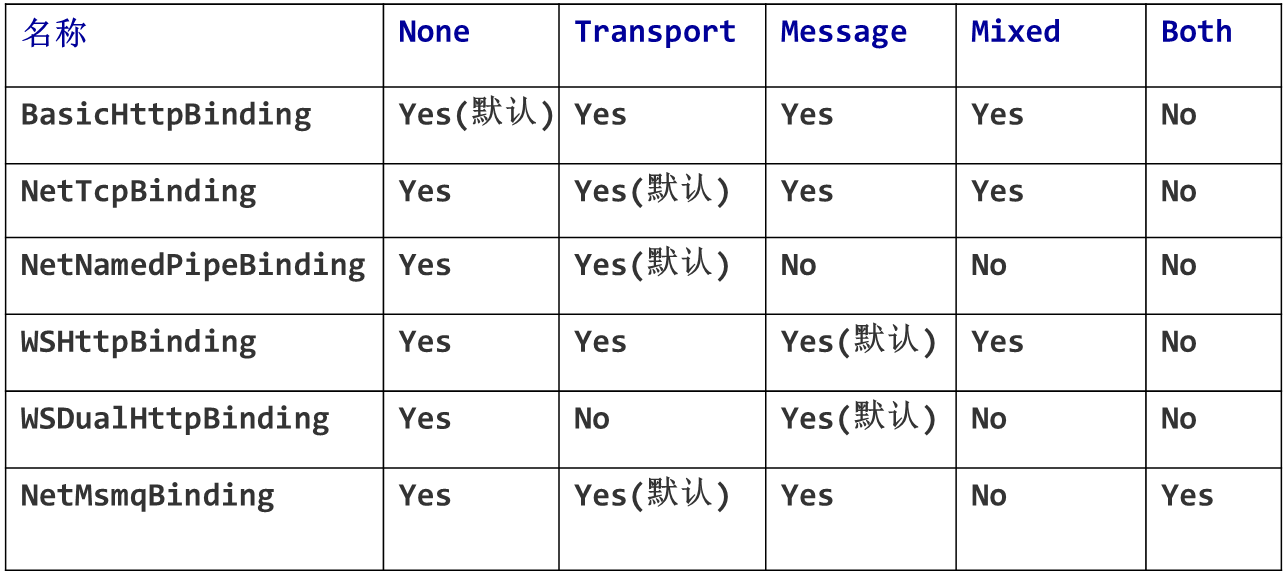
**传输安全模式**

* 传输安全模式：
  + None：关闭了传输安全的功能
  + transport security：对通道上所有的通信加密,除了接收者外,没有任何一方可以看见消息的内容。
  + message security：只是对消息的本身进行加密而不是对传输进行加密。
  + Mixed：在实现完整性和机密性以及服务端身份验证时采用的是Transport Security模式,而在保护客户端安全凭证的安全时采用了Message Security模式。
  + Both：同是使用Transport模式和Message模式,就是传输时安全的,传输的消息也是经过加密的.

传输安全模式是在绑定中配置的.

**binding与传输安全模式**

**传输安全模式与各种binding是一个组合的关系,并不是每种binding都能应用所有的传输安全模式**，**所有WCF的绑定默认都是安全的,只有BasicHttpBinding默认是非安全的**



**传输安全设置**

<bindings>

<netTcpBinding>

<binding name="tcpBinding">

<security mode="Transport">

<transport clientCredentialType="Windows“

protectionLevel="EncryptAndSign"/>

</security>

</binding>

</netTcpBinding>

</bindings>

**编程方式实现:**

NetTcpBinding binding = new NetTcpBinding(SecurityMode.Transport);

**或者：**

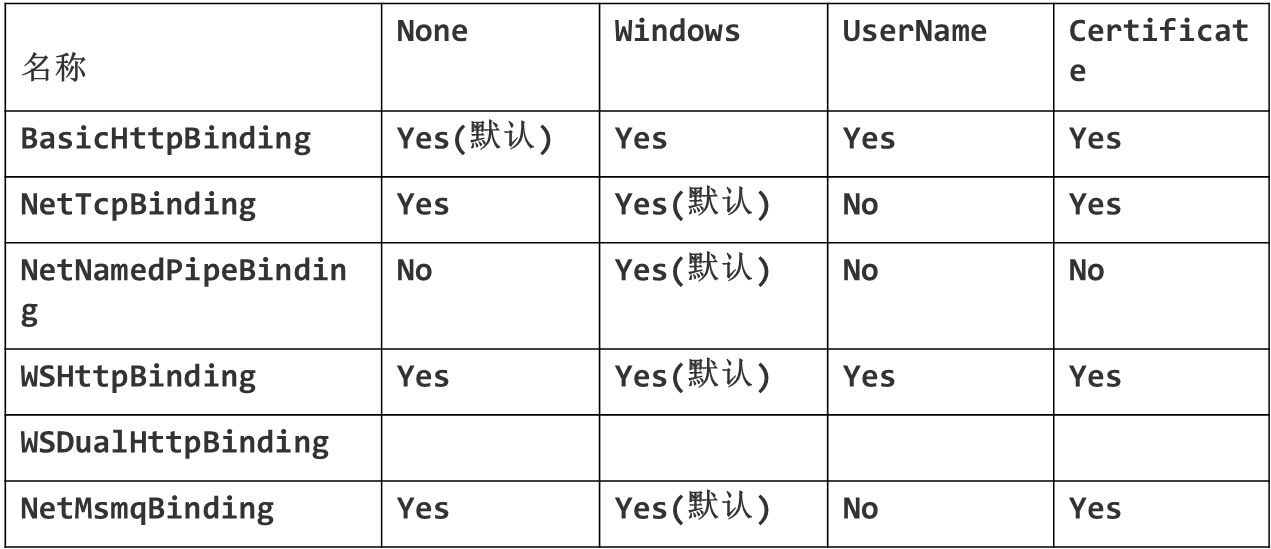
NetTcpBinding binding1 = new NetTcpBinding();

binding1.Security.Mode = SecurityMode.Transport;

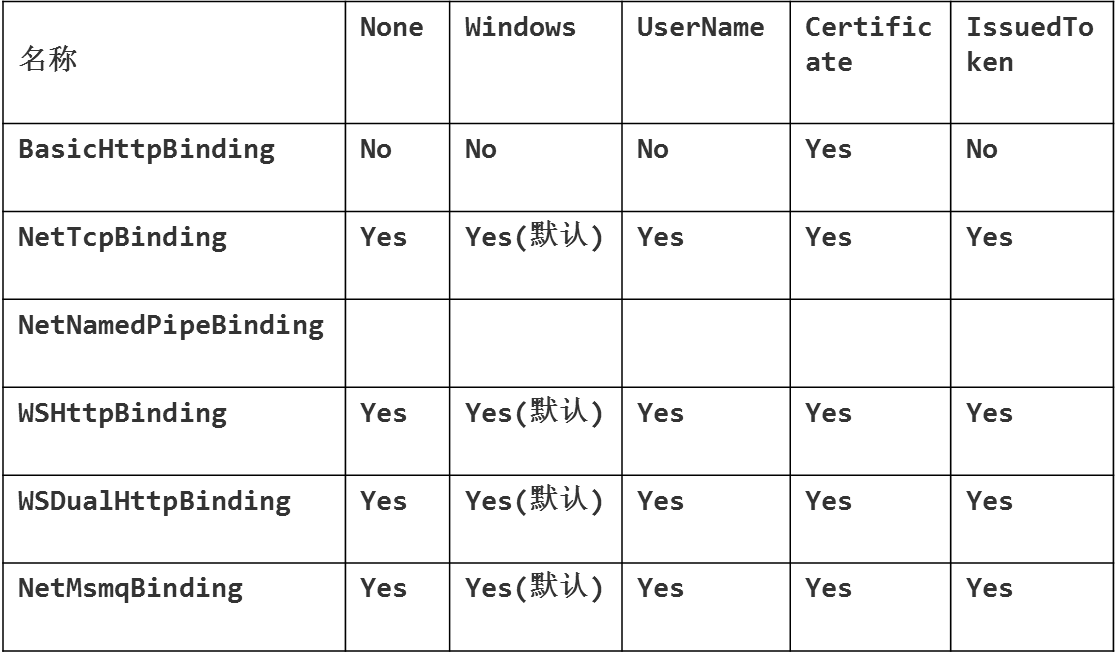
Transport模式的身份验证

**下面看一下,在传输模式下(Transport、Message),身份验证与binding的关系**

在Transport模式下:



**Message**模式的身份验证



除了BasicHttpBinding和NetNamedPipeBinding以外,其他的模式默认都是用Windows凭证

**证书与传输安全**

* 传输安全必须要使用证书
  + SSL安全套接字层一个数据传输加密机制，它可以确保在客户机与服务器之间传输的数据仍然是安全与隐密的。传输安全 (HTTPS) 确保保密性和完整性。SSL 广泛用于 Internet 中，以便向客户端证明服务的身份，并且随后向通道提供保密性（加密）。
  + 证书就相当于是身份证一样，客户端可以不进行身份验证。但是要有一个服务器证书来保证客户端和服务器之间能够建立SSL安全套接字连接。

**证书制作**

* 使用Makecert命令创建证书
* Makecert证书创建工具生成的 X.509 证书。它创建用于数字签名的公钥和私钥对，并将其存储在证书文件中。此工具还将密钥对与指定发行者的名称相关联，并创建一个 X.509 证书，该证书将用户指定的名称绑定到密钥对的公共部分。

makecert -sr localmachine -ss My -n CN=WCFServerPK -sky exchange -pe -r

mmc命令 查看证书

**示例：**

例：WSHttpBinding 绑定，匿名客户端（none）的传输安全模式（Transport）

1、制作证书

2、SSL证书设置

3、服务端配置

4、客户端配置

服务端配置

<!-- 部署服务库项目时，必须将配置文件的内容添加到

主机的 app.config 文件中。System.Configuration 不支持库的配置文件。-->

<system.serviceModel>

<bindings>

<wsHttpBinding>

<binding name="binding1">

<security mode="Transport">

<transport clientCredentialType="None"/>

</security>

</binding>

</wsHttpBinding>

</bindings>

<services>

<service name="WcfServiceLibrary.Service1" behaviorConfiguration="WcfServiceLibrary.Service1Behavior">

<host>

<baseAddresses>

<add baseAddress = "https://localhost:3300/cert" />

</baseAddresses>

</host>

<!-- Service Endpoints -->

<!-- 除非完全限定，否则地址将与上面提供的基址相关 -->

<endpoint address ="" binding="wsHttpBinding" bindingConfiguration="binding1" contract="WcfServiceLibrary.IService1">

<!--

部署时，应删除或替换下列标识元素，以反映

在其下运行部署服务的标识。删除之后，WCF 将

自动推导相应标识。

-->

<identity>

<dns value="localhost"/>

</identity>

</endpoint>

<!-- Metadata Endpoints -->

<!-- 元数据交换终结点由服务用于向客户端做自我描述。-->

<!-- 此终结点不使用安全绑定，应在部署前确保其安全或将其删除-->

<endpoint address="mex" binding="mexHttpsBinding" contract="IMetadataExchange"/>

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="WcfServiceLibrary.Service1Behavior">

<!-- 为避免泄漏元数据信息，

请在部署前将以下值设置为 false 并删除上面的元数据终结点 -->

<serviceMetadata httpsGetEnabled="True"/>

<!-- 要接收故障异常详细信息以进行调试，

请将下值设置为 true。在部署前

设置为 false 以避免泄漏异常信息-->

<serviceDebug includeExceptionDetailInFaults="False" />

<serviceCredentials>

<serviceCertificate storeLocation="LocalMachine" x509FindType="FindBySubjectName" findValue="WCFServerPK" storeName="My"/>

</serviceCredentials>

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

Program.cs 代码里：

ServiceHost host = new ServiceHost(typeof(WcfServiceLibrary.Service1));

host.Open();

Console.WriteLine("服务已启动!");

Console.ReadLine();

客户端添加服务引用：

Programs.cs里；

static void Main(string[] args)

{

host.Service1Client client = new client.host.Service1Client();

ServicePointManager.ServerCertificateValidationCallback += RemoteCertificateValidate;

string s = client.hello("北风");

Console.WriteLine(s);

Console.ReadLine();

}

//强制信任证书

private static bool RemoteCertificateValidate(

object sender, X509Certificate cert,

X509Chain chain, SslPolicyErrors error)

{

return true;

}