**[访问需要HTTP Basic Authentication认证的资源的各种语言的实现](http://www.cnblogs.com/QLeelulu/archive/2009/11/22/1607898.html)**

无聊想调用下嘀咕的api的时候，发现需要HTTP Basic Authentication，就看了下。

什么是HTTP Basic Authentication？直接看[http://en.wikipedia.org/wiki/Basic\_authentication\_scheme](http://en.wikipedia.org/wiki/Basic_authentication_scheme" \o "http://en.wikipedia.org/wiki/Basic_authentication_scheme)吧。

在你访问一个需要HTTP Basic Authentication的URL的时候，如果你没有提供用户名和密码，服务器就会返回401，如果你直接在浏览器中打开，浏览器会提示你输入用户名 和密码(google浏览器不会，bug？)。你可以尝试点击这个url看看效果：<http://api.minicloud.com.cn/statuses/friends_timeline.xml>

要在发送请求的时候添加HTTP Basic Authentication认证信息到请求中，有两种方法：

* 一是在请求头中添加Authorization：  
  Authorization: "Basic 用户名和密码的base64加密字符串"
* 二是在url中添加用户名和密码：  
  <http://userName:password@api.minicloud.com.cn/statuses/friends_timeline.xml>

下面来看下对于第一种在请求中添加Authorization头部的各种语言的实现代码。

先看.NET的吧：

[复制代码](javascript:void(0);)

string username="username";  
string password="password";  
//注意这里的格式哦，为 "username:password"  
string usernamePassword = username + ":" + password;  
CredentialCache mycache = new CredentialCache();  
mycache.Add(new Uri(url), "Basic", new NetworkCredential(username, password));  
myReq.Credentials = mycache;  
myReq.Headers.Add("Authorization", "Basic " + Convert.ToBase64String(new ASCIIEncoding().GetBytes(usernamePassword)));   
  
WebResponse wr = myReq.GetResponse();  
Stream receiveStream = wr.GetResponseStream();  
StreamReader reader = new StreamReader(receiveStream, Encoding.UTF8);  
string content = reader.ReadToEnd();

**Transport Security with Basic Authentication**

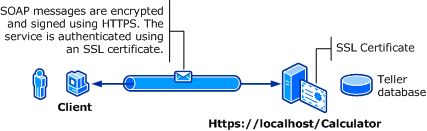
**.NET Framework 4.5**

[Other Versions](javascript:;)

http://i3.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635533092500576233

2 out of 3 rated this helpful - [Rate this topic](http://msdn.microsoft.com/en-us/library/ms733775%28v=vs.110%29.aspx#feedback)

The following illustration shows a Windows Communication Foundation (WCF) service and client. The server needs a valid X.509 certificate that can be used for Secure Sockets Layer (SSL), and the clients must trust the server’s certificate. Further, the Web service already has an SSL implementation that can be used. For more information about enabling basic authentication on Internet Information Services (IIS), see <http://go.microsoft.com/fwlink/?LinkId=83822>.



|  |  |
| --- | --- |
| **Characteristic** | **Description** |
| Security Mode | Transport |
| Interoperability | With existing Web service clients and services |
| Authentication (Server)  Authentication (Client) | Yes (using HTTPS)  Yes (through User name/Password) |
| Integrity | Yes |
| Confidentiality | Yes |
| Transport | HTTPS |
| Binding | [WSHttpBinding](http://msdn.microsoft.com/en-us/library/system.servicemodel.wshttpbinding%28v=vs.110%29.aspx) |

[**Service**](javascript:void(0))

The following code and configuration are meant to run independently. Do one of the following:

* Create a stand-alone service using the code with no configuration.
* Create a service using the supplied configuration, but do not define any endpoints.

**ms733775.collapse_all(en-us,VS.110).gifCode**

The following code shows how to create a service endpoint that uses a Windows domain user name and password for transfer security. Note that the service requires an X.509 certificate to authenticate to the client. For more information, see [Working with Certificates](http://msdn.microsoft.com/en-us/library/ms731899%28v=vs.110%29.aspx) and [How to: Configure a Port with an SSL Certificate](http://msdn.microsoft.com/en-us/library/ms733791%28v=vs.110%29.aspx).

C#

[VB](http://msdn.microsoft.com/en-us/library/ms733775%28v=vs.110%29.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

// Create the binding.

WSHttpBinding binding = new WSHttpBinding();

binding.Security.Mode = SecurityMode.Transport;

binding.Security.Transport.ClientCredentialType =

HttpClientCredentialType.Basic;

// Create the URI for the endpoint.

Uri httpUri = new Uri("https://localhost/Calculator");

// Create the service host and add an endpoint.

ServiceHost myServiceHost = new ServiceHost(

typeof(ServiceModel.Calculator), httpUri);

myServiceHost.AddServiceEndpoint(

typeof(ServiceModel.ICalculator), binding, "");

// Open the service.

myServiceHost.Open();

Console.WriteLine("Listening...");

Console.WriteLine("Press Enter to exit.");

Console.ReadLine();

// Close the service.

myServiceHost.Close();

[**Configuration Client**](javascript:void(0))

**ms733775.collapse_all(en-us,VS.110).gifCode**

The following code shows the client code that includes the user name and password. Note that the user must provide a valid Windows user name and password. The code to return the user name and password is not shown here. Use a dialog box or other interface to query the user for the information.

|  |
| --- |
| **NoteNote** |
| User name and password can only be set using code. |

C#

[VB](http://msdn.microsoft.com/en-us/library/ms733775%28v=vs.110%29.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-3)

// Create the binding.

WSHttpBinding myBinding = new WSHttpBinding();

myBinding.Security.Mode = SecurityMode.Transport;

myBinding.Security.Transport.ClientCredentialType =

HttpClientCredentialType.Basic;

// Create the endpoint address. Note that the machine name

// must match the subject or DNS field of the X.509 certificate

// used to authenticate the service.

EndpointAddress ea = new

EndpointAddress("https://machineName/Calculator");

// Create the client. The code for the calculator

// client is not shown here. See the sample applications

// for examples of the calculator code.

CalculatorClient cc =

new CalculatorClient(myBinding, ea);

// The client must provide a user name and password. The code

// to return the user name and password is not shown here. Use

// a database to store the user name and passwords, or use the

// ASP.NET Membership provider database.

cc.ClientCredentials.UserName.UserName = ReturnUsername();

cc.ClientCredentials.UserName.Password = ReturnPassword();

try

{

// Begin using the client.

cc.Open();

Console.WriteLine(cc.Add(100, 11));

Console.ReadLine();

// Close the client.

cc.Close();

}

**ms733775.collapse_all(en-us,VS.110).gifConfiguration**

The following code shows the client configuration.

|  |
| --- |
| **NoteNote** |
| You cannot use configuration to set the user name and password. The configuration shown here must be augmented using code to set the user name and password. |

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<system.serviceModel>

<bindings>

<wsHttpBinding>

<binding name="WSHttpBinding\_ICalculator" >

<security mode="Transport">

<transport clientCredentialType="Basic" />

</security>

</binding>

</wsHttpBinding>

</bindings>

<client>

<endpoint address="https://machineName/Calculator"

binding="wsHttpBinding"

bindingConfiguration="WSHttpBinding\_ICalculator"

contract="ICalculator"

name="WSHttpBinding\_ICalculator" />

</client>

</system.serviceModel>

</configuration>

# Authentication and Authorization in WCF Services - Part 1

100 out of 116 rated this helpful - [Rate this topic](http://msdn.microsoft.com/library/ff405740.aspx#feedback)

Robert Green

MCW Technologies

**Download**

* [C# version](http://code.msdn.microsoft.com/Project/Download/FileDownload.aspx?ProjectName=learnWCF&DownloadId=9143)
* [VB version](http://code.msdn.microsoft.com/Project/Download/FileDownload.aspx?ProjectName=learnWCF&DownloadId=9144)

**Articles in this series**

* [Introduction to Windows Communication Foundation](http://msdn.microsoft.com/en-us/library/dd936243.aspx)
* [Hosting Windows Communication Foundation Services](http://msdn.microsoft.com/en-us/library/ee939285.aspx)
* [Self Hosting Windows Communication Foundation Services](http://msdn.microsoft.com/en-us/library/ee939340.aspx)
* [Creating and Maintaining Service and Data Contracts](http://msdn.microsoft.com/en-us/library/ee922570.aspx)
* [Fault Handling in WCF Services](http://msdn.microsoft.com/en-us/library/ee942778.aspx)
* [Sessions, Instancing and Concurrency in WCF Services](http://msdn.microsoft.com/en-us/library/ff183865.aspx)
* [Transactions in WCF Services](http://msdn.microsoft.com/en-us/library/ff384250.aspx)
* [Message Patterns in WCF Services](http://msdn.microsoft.com/library/ff395349.aspx)
* [Authentication and Authorization in WCF Services - Part 1](http://msdn.microsoft.com/library/ff405740.aspx)
* [Authentication and Authorization in WCF Services - Part 2](http://msdn.microsoft.com/library/ff406125.aspx)

### Introduction

When client applications and WCF services communicate, they do so by passing XML messages. Securing these messages is an important part of building, hosting and calling WCF services. In this two-part tutorial, you will ask questions related to security. For example:

* Can any client call the service or do you want to control who can call the service?
* Can any client call any method of the service or do you want to control what clients can call what methods?
* Can any client execute all of the code in a method or do you want to control what clients can execute what code?

The answers to these questions determine your use of authentication and authorization. Authentication enables you to identify clients that call the service. Authorization enables you to determine what operations authenticated clients can access. You will typically base authorization on roles.

To authorize, you first need to authenticate. To do that, you must be able to identify clients. Clients can identify themselves by providing evidence such as a Windows account, a user name / password or a certificate. Clients must also know that they are calling the service they intend to call. Services can identify themselves by providing a certificate.

In this two-part tutorial, you will see how to authenticate clients and control their access in the following three scenarios:

* Internal Self-Hosted Services. You are self-hosting a WCF service on the corporate network. You want anyone who can log onto the network to be able to access the service. You want only certain users to be able to call particular methods.
* Internal Web-Hosted Services. You are hosting a WCF service using Internet Information Services on the corporate intranet. Both employees and guests have access to the wireless network. You want only employees to be able to call the service. You want only certain users to be able to call particular methods.
* Public Web-Hosted Services. You are hosting a WCF service publicly on the Internet. You want to limit access to the service to users with a valid user name and password. You want only certain users to be able to call particular methods.

Before you explore authentication and authorization, you need a basic understanding of WCF security.

### Review Basic WCF Security Concepts

When you write a WCF service, you write it using the .NET Framework and either C# of Visual Basic. To the extent that a WCF service is a .NET application, you can use familiar security techniques, including code access and role-based security. However, securing the code that runs on the computer hosting the service is only part of the story.

Remember that the client and service communicate by exchanging XML messages. You may need to secure these messages as well. Protecting messages as they are transferred from client to server and back is known as transfer security. WCF provides two mechanisms for transfer security: transport security and message security.

#### Transport Security and Message Security

If you use transport security, security occurs at the transport level. The packets sent “on the wire” include the caller’s credentials and the message. Both of which are encrypted using whatever mechanism the transport protocol uses. For example, if you use TCP, you will likely use Transport Layer Security (TLS) and if you use HTTPS, you will likely use Secure Sockets Layer (SSL).

If you use message security, the caller’s credentials are included in the message and the message is encrypted using the WS-Security specification.

It is generally faster to encrypt and decrypt messages that use transport security and you can benefit from hardware acceleration to improve performance.

A downside to transport security is that messages are encrypted only from point to point. Suppose a client sends a message to a service. The client encrypts the message and the service decrypts it. If the service then forwards the message to another service, the service forwarding the message will not automatically encrypt it. This is not an issue with message security because the service will encrypt the message before passing it on to another service.

A downside to message security is that it requires both clients and services to support the WS-Security specification. Transport security does not have this requirement and is therefore more interoperable.

#### Configure Security

WCF supports the following six security modes:

* None. Messages are not secured.
* Transport. Messages are secured using transport security. You will use this in the sample applications that demonstrate the first two scenarios (internal self-hosted and Web-hosted services).
* Message. Messages are secured using message security. You will use this in the sample application that demonstrates the first scenario (internal self-hosted service).
* TransportWithMessageCredential. Message protection and authorization occur at the transport level and credentials are passed with the message. You will use this in the sample application that demonstrates the third scenario (public Web-hosted service).
* TransportCredentialOnly. Credentials are passed at the transport level but the message is not encrypted. This option is available only if you are using the BasicHttpBinding binding.
* Both. Messages are secured using both transport level and message level security. This is supported only if you are using Microsoft Message Queue Server.

Each binding has a default set of security settings. By default, the following bindings use message security: WSHttpBinding, WS2007HttpBinding, WSDualHttpBinding, WSFederationBinding and WS2007FederationBinding.

By default, the following bindings use transport security: NetTcpBinding, NetNamedPipesBinding, NetMsmqBinding, NePeerBinding and MsmqIntegrationBinding.

By default, the BasicHttpBinding binding uses None as its security mode. In other words, message sent using that binding are not secure. This enables interoperability with ASMX Web services.

You can modify the settings by modifying properties of the binding and by specifying service behaviors. You will see examples of how to do this in this tutorial. For more detailed information, see the Programming WCF Security (http://msdn.microsoft.com/en-us/library/ms731925.aspx) and Bindings and Security (http://msdn.microsoft.com/en-us/library/ms731172.aspx) Help topics.

#### Authenticate Clients and Services

Authentication enables you to identify clients and services. They identify themselves by passing credentials. WCF supports the following credential types when you are using transport level security:

* Windows. The client uses a Windows token representing the logged in user’s Windows identity. The service uses the credentials of the process identity or an SSL certificate. You will use this in the sample application that demonstrates the first scenario (internal self-hosted service).
* Basic. The client passes a user name and password to the service. Typically, the user will enter the user name and password in a login dialog box. The service uses a SSL certificate. This option is available only with HTTP protocols. You will use this in the sample application that demonstrates the second scenario (internal Web-hosted service).
* Certificate. The client uses an X.509 certificate and the service uses either that certificate or an SSL certificate.
* NTLM. The service validates the client using a challenge/response scheme against Windows accounts. The service uses a SSL certificate. This option is available only with HTTP protocols.
* None. The service does not validate the client.

WCF supports the following credential types when you are using message level security:

* Windows. The client uses a Windows token representing the logged in user’s Windows identity. The service uses the credentials of the process identity or an SSL certificate. You will use this in the sample application that demonstrates the first scenario (internal self-hosted service).
* UserName. The client passes a user name and password to the service. Typically, the user will enter the user name and password in a login dialog box. The service can validate the user name and password using a Windows account or the ASP.NET membership provider. You will use this in the sample application that demonstrates the third scenario (public Web-hosted service).
* Certificate. The client uses an X.509 certificate and the service uses either that certificate or an SSL certificate.
* IssueToken. The client and service use the Secure Token Service, which issues tokens the client and service trust. Windows CardSpace uses the Secure Token Service.
* None. The service does not validate the client.

#### Authorize Clients

Authorization enables you to determine what operations authenticated clients can access. WCF supports three basic approaches to authorization:

* Role-based. Access to a service and to operations of the service is based on the user’s role. All of the samples in this tutorial demonstrate the use of role-based authorization.
* Identity based. Access is based on claims made within the user’s credentials. This is an extension to role-based authorization and provides a more fine grained approach. This approach will typically be used with issue token authentication.
* Resource based. Resources, such as WCF services, are secured using Windows Access Control Lists (ACLs).

You have three options when deciding how to determine a user’s role:

* Windows groups. You can use the built-in Windows groups such as Administrators or Power Users or create your own Windows groups. You will use this in the sample applications that demonstrate the first two scenarios (internal self-hosted and Web-hosted services).
* Custom roles. You can create roles that are specific to your application, such as Manager, Employee, Administrator, etc.
* ASP.NET role management. You can use the ASP.NET role provider and use roles you have defined for a Web site. You will use this in the sample application that demonstrates the third scenario (public Web-hosted service).

### Control Access to and Usage of Internal Self-Hosted Services

The first example in this tutorial involves an internal self-hosted service. In this scenario, you are self-hosting a WCF service on the corporate network. You want anyone who can log onto the network to be able to access the service. However, you want only certain users to be able to call particular methods.

To explore authentication and authorization in this scenario, start Visual Studio 2008 and select File | Open | Project/Solution to display the Open Project dialog box. Navigate to the folder where you downloaded this tutorial’s sample project. Navigate to the InternalSelfHostedServiceDemo folder. Select InternalSelfHostedServiceDemo.sln and click OK to open the project. The sample application includes three projects. The SecureServiceLibrary project contains a simple WCF service. The ConsoleHost project uses a console application to host the service.

The WCF service has two operations. The SayHello method returns a string that includes the name of the logged in user on the client computer. The ReportSales method returns $10,000, representing today’s sales.

// C#

public string SayHello()

{

  return string.Format("Hello {0}",

    System.Threading.Thread.CurrentPrincipal.Identity.Name);

}

public decimal ReportSales()

{

  return 10000M;

}

' Visual Basic

Public Function SayHello() As String Implements ISecureService.SayHello

  Return String.Format("Hello {0}", \_

    Threading.Thread.CurrentPrincipal.Identity.Name)

End Function

Public Function ReportSales() As Decimal \_

  Implements ISecureService.ReportSales

  Return 10000D

End Function

Press F5 to run the application. You may see a Windows Security Alert (see Figure 1). Click Unblock to enable the service host to listen for client requests over TCP.



**Figure 1. Click Unblock to enable the service host to listen for requests.**

You should see the following displayed in the console window:

The service is running and is listening on:

net.tcp://localhost:9000/SecureService (NetTcpBinding)

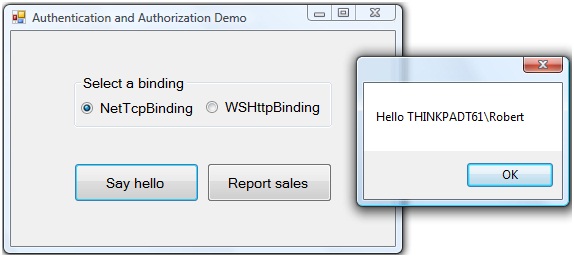
http://localhost:8090/SecureService (WSHttpBinding)

net.tcp://localhost:9001/mex (MetadataExchangeTcpBinding)

Press any key to stop the service.

The app.config file in the ConsoleHost project defines three endpoints for the service. Clients can communicate with the service using either the NetTcpBinding or WSHttpBinding bindings. When you add a service reference, Visual Studio uses the third endpoint to download metadata from the service.

In the form, select NetTcpBinding and then click Say hello. The application should display a message similar to the one shown in Figure 2, but with your logged in user name.



**Figure 2. The service returns to the client a greeting with your name.**

Click OK to dismiss the message. Select WSHttpBinding and click Say hello. You should see the same greeting. Click OK to dismiss the message. Click Report sales. You should see a message informing you that today’s sales are $10,000. Click OK to dismiss the message. Close the form and then press any key in the console window to stop the service.

When you clicked the Say hello button, you called the WCF service, which returned a string containing your user name. To understand how this happened, open the app.config file in the WindowsClient project. You will see the following:

<bindings>

  <netTcpBinding>

    <binding name="SecureService\_Tcp"

      …

      <security mode="Transport">

        <transport clientCredentialType="Windows"

                   protectionLevel="EncryptAndSign" />

        <message clientCredentialType="Windows" />

      </security>

    </binding>

  </netTcpBinding>

  <wsHttpBinding>

    <binding name="SecureService\_WsHttp"

      …

      <security mode="Message">

        <transport clientCredentialType="Windows"

                   proxyCredentialType="None"

                   realm="" />

        <message clientCredentialType="Windows"

                 negotiateServiceCredential="true"

                 algorithmSuite="Default"

                 establishSecurityContext="true" />

      </security>

    </binding>

  </wsHttpBinding>

</bindings>

By default, the NetTcpBinding binding uses transport security.

<security mode="Transport">

By default, the WSHttpBinding binding uses transport security.

<security mode="Message">

By default, both bindings use Windows as the client credential type.

<transport clientCredentialType="Windows"

<message clientCredentialType="Windows"

When the client code creates a new instance of the service proxy class, it passes your Windows credentials to the service. The SayHello method code uses Threading.Thread.CurrentPrincipal.Identity.Name to identify the user by name and build the greeting string.

// C#

public string SayHello()

{

  return string.Format("Hello {0}",

    System.Threading.Thread.CurrentPrincipal.Identity.Name);

}

' Visual Basic

Public Function SayHello() As String Implements ISecureService.SayHello

  Return String.Format("Hello {0}", \_

    Threading.Thread.CurrentPrincipal.Identity.Name)

End Function

The CurrentPrincipal property of the Thread class returns an instance of the IPrincipal interface. It represents the security context of the user on whose behalf the code is running. This context includes the user’s identity and roles to which he or she belongs. The Identify property of IPrincipal returns the identity. This property is an instance of the IIdentity interface.

Now that you know how to tell who is calling the service, you can add code to control what code they can execute. In the Solution Explorer, double-click the SecureService file. Add the following to the top of the code file:

// C#

using System.Security.Permissions;

' Visual Basic

Imports System.Security.Permissions

Make the following change to the ReportSales method:

// C#

[PrincipalPermission(SecurityAction.Demand,

  Role="BUILTIN\\BackupOperators")]

public decimal ReportSales()

' Visual Basic

<PrincipalPermission(SecurityAction.Demand, \_

  Role:="BUILTIN\\BackupOperators")> \_

Public Function ReportSales() As Decimal \_

  Implements ISecureService.ReportSales

In this code, you are using the PrincipalPermission attribute to demand that users belong to a specific role in order to execute the code in the ReportSales method. For the purposes of this demo, use a role to which you do not belong.

If you call this method and are not in the Windows built-in BackupOperators group, a security exception will occur. You will next trap for this in the client application. In the Solution Explorer, right-click the Form1 file and select View Code. Add the following to the top of the code file:

// C#

using System.ServiceModel.Security;

' Visual Basic

Imports System.ServiceModel.Security

Add the following code  to the reportSalesButton\_Click method:

// C#

try

{

  MessageBox.Show(String.Format(

    "Today's sales are {0:C}", proxy.ReportSales()));

}

catch (SecurityAccessDeniedException securityEx)

{

  MessageBox.Show(securityEx.Message);

}

catch (FaultException faultEx)

{

  MessageBox.Show(faultEx.Message);

}

' Visual Basic

Try

  MessageBox.Show(String.Format( \_

    "Today's sales are {0:C}", proxy.ReportSales))

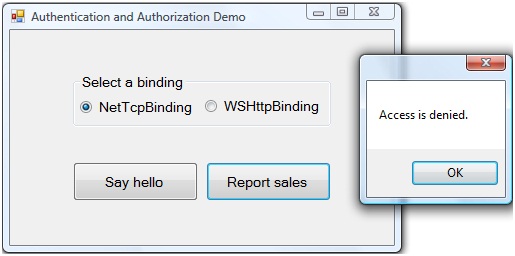
Catch securityEx As SecurityAccessDeniedException

  MessageBox.Show(securityEx.Message)

Catch faultEx As FaultException

  MessageBox.Show(faultEx.Message)

Save your changes and build the solution. Press CTRL+F5 to run the application in release mode. Confirm that the console window displays that the services are hosted. In the form, select either binding and click Report sales. The application should display the message shown in Figure 3.



**Figure 3. You are not authorized to see the sales number.**

Click OK to dismiss the message. Close the form and then press any key in the console window to stop the service.

You just used the PrincipalPermission attribute at the operation level to prevent unauthorized clients from calling a method. You might want more fine-grained control. You will next modify your service code to allow all clients to call the ReportSales method. However, only authorized users will be able to see the actual sales number.

Return to the SecureService code file. Add the following to the top of the code file:

// C#

using System.Security;

using System.Security.Principal;

' Visual Basic

Imports System.Security

Imports System.Security.Principal

Make the following changes  to the ReportSales method:

// C#

//[PrincipalPermission(SecurityAction.Demand,

//  Role="BUILTIN\\BackupOperators")]

public decimal ReportSales()

{

  var currentUser = new WindowsPrincipal((WindowsIdentity)

    System.Threading.Thread.CurrentPrincipal.Identity);

  if (currentUser.IsInRole(WindowsBuiltInRole.BackupOperator))

  {

    return 10000M;

  }

  else

  {

    return -1M;

  }

}

' Visual Basic

'<PrincipalPermission(SecurityAction.Demand, \_

'  Role:="BUILTIN\\BackupOperators")> \_

Public Function ReportSales() As Decimal \_

  Implements ISecureService.ReportSales

  Dim currentUser As New WindowsPrincipal(CType( \_

    Threading.Thread.CurrentPrincipal.Identity, WindowsIdentity))

  If currentUser.IsInRole(WindowsBuiltInRole.BackupOperator) Then

    Return 10000D

  Else

    Return -1D

  End If

End Function

The WindowsPrincipal class represents the security context of a Windows user. You can use the IsInRole method of that class to determine if a user has a particular role. The code above casts the thread’s identity to a WindowsIdentity object and passes that to the WindowsPrincipal constructor.

You will next modify the client application to display a more user-friendly message if the user is not authorized to view sales numbers. Return to the Form1 code file and make the following changes  to the reportSalesButton\_Click method:

// C#

try

{

  decimal sales = proxy.ReportSales();

  if (sales != -1)

  {

    MessageBox.Show(String.Format("Today's sales are {0:C}", sales));  }

  else

  {

    MessageBox.Show(String.Format(

      "You are not authorized to view sales numbers"));

  }

}

catch (SecurityAccessDeniedException securityEx)

' Visual Basic

Try

  Dim sales As Decimal = proxy.ReportSales

  If sales <> -1 Then

    MessageBox.Show(String.Format( \_

      "Today's sales are {0:C}", sales))

  Else

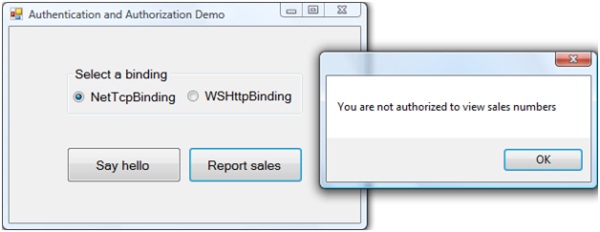
    MessageBox.Show( \_

      "You are not authorized to view sales numbers")

  End If

Catch securityEx As SecurityAccessDeniedException

Save your changes and build the solution. Press CTRL+F5 to run the application in release mode. Confirm that the console window displays that the services are hosted. In the form, select either binding and click Report sales. The application should display the message shown in Figure 4.



**Figure 4. You are not authorized to see the sales number.**

Click OK to dismiss the message. Close the form and then press any key in the console window to stop the service.

### Conclusion

In this tutorial, you first saw a high-level overview of WCF security. You reviewed the difference between transport security and message security and you reviewed the options each provides for authenticating users who call a service.

You then saw how to use authentication and authorization in the scenario where you are self-hosting a WCF service on the corporate network. You saw that if you use either the NetTcpBinding or WSHttpBinding binding, the client automatically passes to the service the credentials of the logged in user. You then saw how to limit usage of a service method based on the role of the user.

In the second part of this two-part tutorial, you will see how to use authentication and authorization in two additional scenarios: an internal Web-hosted service and a public Web-hosted service.

For additional information on securing WCF services, you should review the Microsoft patterns & practices Improving Web Services Security Guide at http://www.codeplex.com/WCFSecurityGuide.

# Authentication and Authorization in WCF Services - Part 2

38 out of 47 rated this helpful - [Rate this topic](http://msdn.microsoft.com/en-us/library/ff406125.aspx#feedback)

Robert Green

MCW Technologies

**Download**

* [C# version](http://code.msdn.microsoft.com/Project/Download/FileDownload.aspx?ProjectName=learnWCF&DownloadId=9145)
* [VB version](http://code.msdn.microsoft.com/Project/Download/FileDownload.aspx?ProjectName=learnWCF&DownloadId=9146)

**Articles in this series**

* [Introduction to Windows Communication Foundation](http://msdn.microsoft.com/en-us/library/dd936243.aspx)
* [Hosting Windows Communication Foundation Services](http://msdn.microsoft.com/en-us/library/ee939285.aspx)
* [Self Hosting Windows Communication Foundation Services](http://msdn.microsoft.com/en-us/library/ee939340.aspx)
* [Creating and Maintaining Service and Data Contracts](http://msdn.microsoft.com/en-us/library/ee922570.aspx)
* [Fault Handling in WCF Services](http://msdn.microsoft.com/en-us/library/ee942778.aspx)
* [Sessions, Instancing and Concurrency in WCF Services](http://msdn.microsoft.com/en-us/library/ff183865.aspx)
* [Transactions in WCF Services](http://msdn.microsoft.com/en-us/library/ff384250.aspx)
* [Message Patterns in WCF Services](http://msdn.microsoft.com/library/ff395349.aspx)
* [Authentication and Authorization in WCF Services - Part 1](http://msdn.microsoft.com/library/ff405740.aspx)
* [Authentication and Authorization in WCF Services - Part 2](http://msdn.microsoft.com/library/ff406125.aspx)

### Introduction

When client applications and WCF services communicate, they do so by passing XML messages. Securing these messages is an important part of building, hosting and calling WCF services. In this two-part tutorial, you will ask questions related to security. For example:

* Can any client call the service or do you want to control who can call the service?
* Can any client call any method of the service or do you want to control what clients can call what methods?
* Can any client execute all of the code in a method or do you want to control what clients can execute what code?

The answers to these questions determine your use of authentication and authorization. Authentication enables you to identify clients that call the service. Authorization enables you to determine what operations authenticated clients can access. You will typically base authorization on roles.

To authorize, you first need to authenticate. To do that, you must be able to identify clients. Clients can identify themselves by providing evidence such as a Windows account, a user name / password or a certificate. Clients must also know that they are calling the service they intend to call. Services can identify themselves by providing a certificate.

In this two-part tutorial, you will see how to authenticate clients and control their access in the following three scenarios:

* Internal Self-Hosted Services. You are self-hosting a WCF service on the corporate network. You want anyone who can log onto the network to be able to access the service. You want only certain users to be able to call particular methods.
* Internal Web-Hosted Services. You are hosting a WCF service using Internet Information Services on the corporate intranet. Both employees and guests have access to the wireless network. You want only employees to be able to call the service. You want only certain users to be able to call particular methods.
* Public Web-Hosted Services. You are hosting a WCF service publicly on the Internet. You want to limit access to the service to users with a valid user name and password. You want only certain users to be able to call particular methods.

[Part 1](http://msdn.microsoft.com/library/ff405740.aspx) of this tutorial explored the first scenario. You will explore the other scenarios here.

### Control Access to and Usage of Private Web-Hosted Services

The second example in this tutorial involves an internal Web-hosted service. In this scenario, you are hosting a WCF service using Internet Information Services (IIS) on the corporate intranet. Only clients with a valid user name and password should be able to call the service and only certain users should be able to call particular methods.

If the only way to reach the Web server is by logging onto the corporate network, then you can use the same techniques you used in the previous scenario. If you use the WSHttpBinding binding, the client will pass the logged-in user’s credentials to the service. You can then check whether the user is in a particular role and if not, deny access to methods or blocks of code.

In the scenario here, however, suppose there is a wireless network available. Guests can log onto the wireless network and therefore could call the WCF service. To authenticate clients, you will require users to log in with their network credentials. You will pass these to the WCF service. This is known as basic authentication. You can then authorize as needed.

To explore authentication and authorization in this scenario, start Visual Studio 2008 and select File | Open | Project/Solution to display the Open Project dialog box. Navigate to the folder where you downloaded this tutorial’s sample project. Navigate to the InternalWebHostedServiceDemo folder. Select InternalWebHostedServiceDemo.sln and click OK to open the project. The sample application includes two projects. The SecureServiceLibrary project contains the same WCF service you saw in the first scenario.

To review, the WCF service contains SayHello and ReportSales methods. The code for these methods is in the SecureService file and is as follows:

// C#

public string SayHello()

{

  return string.Format("Hello {0}",

    System.Threading.Thread.CurrentPrincipal.Identity.Name);

}

public decimal ReportSales()

{

  var currentUser = new WindowsPrincipal((WindowsIdentity)

    System.Threading.Thread.CurrentPrincipal.Identity);

  if (currentUser.IsInRole(WindowsBuiltInRole.BackupOperator))

  {

    return 10000M;

  }

    else

  {

    return -1M;

  }

}

' Visual Basic

Public Function SayHello() As String Implements ISecureService.SayHello

  Return String.Format("Hello {0}", \_

    Threading.Thread.CurrentPrincipal.Identity.Name)

End Function

Public Function ReportSales() As Decimal \_

  Implements ISecureService.ReportSales

  Dim currentUser As New WindowsPrincipal(CType( \_

    Threading.Thread.CurrentPrincipal.Identity, WindowsIdentity))

  If currentUser.IsInRole(WindowsBuiltInRole.BackupOperator) Then

    Return 10000D

  Else

    Return -1D

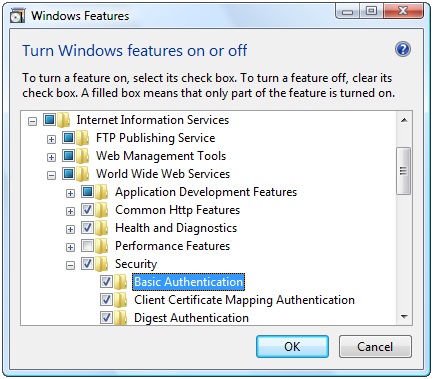
  End If

End Function

The SayHello method returns a string containing the name of the logged-in user. The ReportSales method returns $10,000 if an authorized user calls the method. Otherwise, it returns -1. For more information on the code in these methods, see the Authentication and Authorization Part 1 tutorial.

For this scenario, you need to enable basic authentication in IIS. Select Control Panel from the Windows Start menu. In the Control Panel window, select Programs. In In the Windows Explorer window that appears, click the Turn Windows features on and off link to display the Windows Features dialog box.

Expand the Internet Information Services node. Then expand the World Wide Web Services node. Next, expand the Security node and check Basic Authentication (see Figure 1). Click OK to close the dialog box.

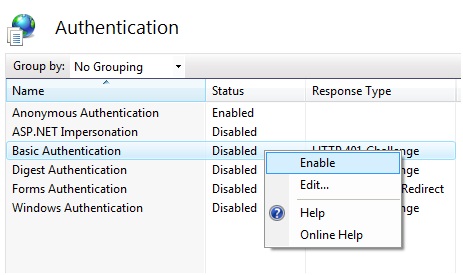


**Figure 1. Enable basic authentication for Internet Information Services.**

When you create the Web site to host the WCF service, you will use HTTPS and SSL to encrypt communications. SSL requires a digital certificate, so you must next create and specify a certificate to use.

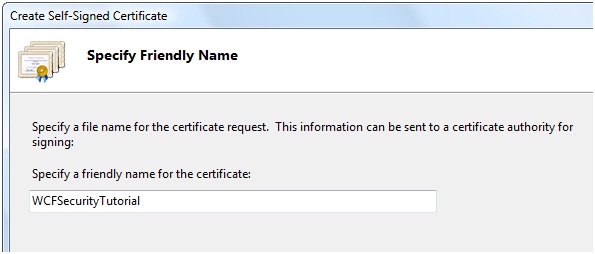
Select Control Panel | Administrative Tools from the Windows Start menu. If you have not configured the Start menu to display Administrative Tools, open the Control Panel, click Classic View and then click on Administrative Tools. In the Windows Explorer window that appears, double-click the Internet Information Services (IIS) Manager shortcut to display the IIS Manager applet.

First, ensure the Web server is set up for basic authentication. In the Connections pane, expand the Sites node. Select Default Web Site. Double-click Authentication in the IIS section to display the Authentication page. Confirm that Basic Authentication has a status of Enabled. If it does not, right-click Basic Authentication and select Enable (see Figure 2).

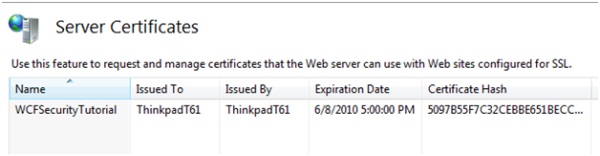


**Figure 2. Enable basic authentication for the Web site if necessary.**

Next you will create a certificate. In the Connections pane, select the top node (the node with the name of your computer. Double-click Server Certificates in the IIS section to display the Server Certificates page. In the Actions pane, select Create Self-Signed Certificate to display the Create Self-Signed Certificate dialog box. Name the cerificate WCFSecurityTutorial (see Figure 3). Click OK to close the dialog box. You should see the certificate listed in the Server Certificates page (see Figure 4).



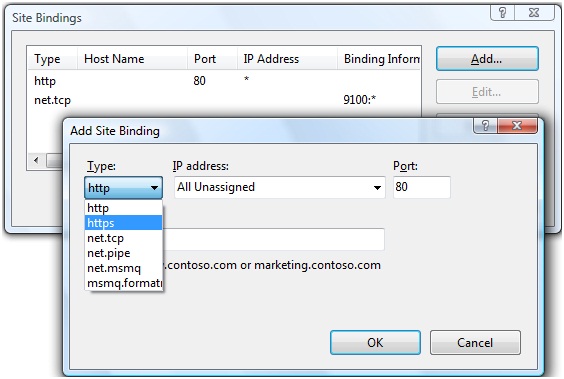
**Figure 3. Name the self-signed certificate.**



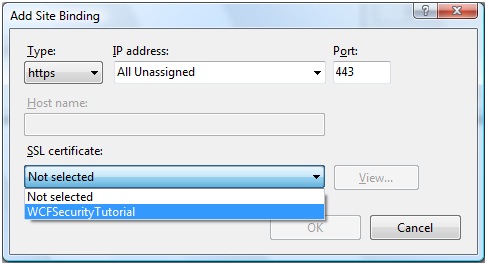
**Figure 4. You have created a self-signed certificate.**

In a production environment, you should not use a self-signed certificate. You should use a certificate from your organization or a trusted third party. You should only use a self-signed certificate for development and testing purposes on your own computer.

Next, you need to configure IIS to enable Web sites to communicate using SSL and the certificate you just created. In the Connections pane, right-click Default Web Site and select Edit Bindings to display the Site Bindings dialog box. Click Add to display the Add Site Binding dialog box. Select https from the Type drop-down list (see Figure 5). Next, select the WCFSecurityTutorial from the SSL certificate drop-down list (see Figure 6).

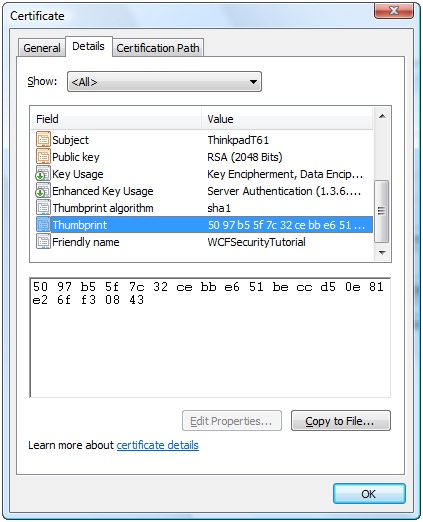


**Figure 5. Add a binding based on HTTPS.**



**Figure 6. Select the certificate to enable HTTPS.**

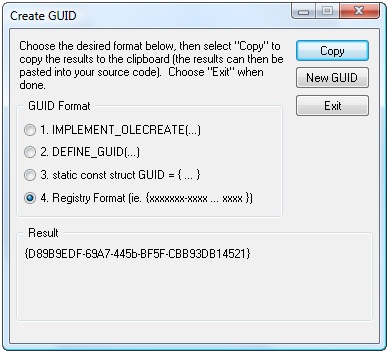
Click View to display the Certificate dialog box. Select the Details tab. In the Field list, select Thumbprint (see Figure 7). Write down the certificate’s thumbprint. You will use it shortly.



**Figure 7. You will use the certificate’s thumbprint shortly.**

Click OK to dismiss the Certificate dialog box. Click OK to close the Add Site Binding dialog box. Click Close to dismiss the Site Bindings dialog box.

You now need to configure a port with the certificate. You will need a GUID so return to Visual Studio and select Tools | Create GUID to display the Create GUID dialog box. Select Registry Format. Visual Studio creates a GUID and displays it in the dialog box (see Figure 8). Click Copy to copy the GUID and then click Exit to close the dialog box.



**Figure 8. Use this dialog box to create a GUID.**

Select Run from the Windows Start menu to display the Run menu. Enter cmd and press OK to display the command prompt. You need to run the command prompt with Administrator privileges. If this does not happen automatically, select All Programs then Accessories from the Windows Start menu. Right-click on Command Prompt and select Run as administrator.

In the command prompt, type the following, with appropriate substitutions, all on one line:

netsh http add sslcert ipport=0.0.0.0:8080

certhash=5097b55f7c32cebbe651beccd50e81e26ff30843

appid={D89B9EDF-69A7-445b-BF5F-CBB96DB14521}

The value of the certhash parameter is the certificate’s thumbprint without spaces. Use the value you wrote down earlier. The value of the appid parameter is the GUID you generated in Visual Studio.

Press ENTER. You should see the following in the command prompt:

SSL Certificate successfully added

Enter Exit to close the command prompt.

You will next create a Web site to host the service. Return to Visual Studio. In the Solution Explorer, right-click the InternalWebHostedServiceDemo solution and select Add | New Web Site to display the Add New Web Site dialog box. Select WCF Service in the Templates list. Select HTTP from the Location drop-down list. Select Visual C# or Visual Basic from the Language drop-down list. In the Location text box, enter https://localhost/InternalWebHostedServiceDemo. In this scenario, you will use basic authentication and SSL, so you specify https in the Web site address. Click OK.

In the Solution Explorer, expand the App\_Code folder under the new web site project you just created. Delete the IService and Service files. In the Solution Explorer, right-click the InternalWebHostedServiceDemo project, and select Add Reference from the context menu. In the Projects tab of the Add Reference dialog box, select SecureServiceLibrary. Click OK to add the reference to the service.

Rename the Service1.svc file to SecureService.svc. Open the file by double clicking on the file name in the Solution Explorer. Since you are referencing an existing service, change the contents of this file to the following:

<%@ ServiceHost Service="SecureServiceLibrary.SecureService" %>

In the Solution Explorer, double-click Web.config. In the system.web section, make the following change in bold to enable debugging of the service at runtime:

// C#

<compilation debug="true" strict="false" explicit="true">

' Visual Basic

<compilation debug="true">

Scroll down to the system.serviceModel section. Make the following changes in bold:

<service name="SecureServiceLibrary.SecureService"

         behaviorConfiguration="ServiceBehavior">

  <!-- Service Endpoints -->

  <endpoint address="" binding="wsHttpBinding"

            bindingConfiguration="WsHttpBindingConfig"

            contract="SecureServiceLibrary.ISecureService">

Add the following XML in bold to configure the WSHttpBinding binding to use Transport security and to expect the client to pass the user name and password:

<system.serviceModel>

  <bindings>

    <wsHttpBinding>

      <binding name="WsHttpBindingConfig">

        <security mode="Transport">

          <transport clientCredentialType="Basic" />

        </security>

      </binding>

    </wsHttpBinding>

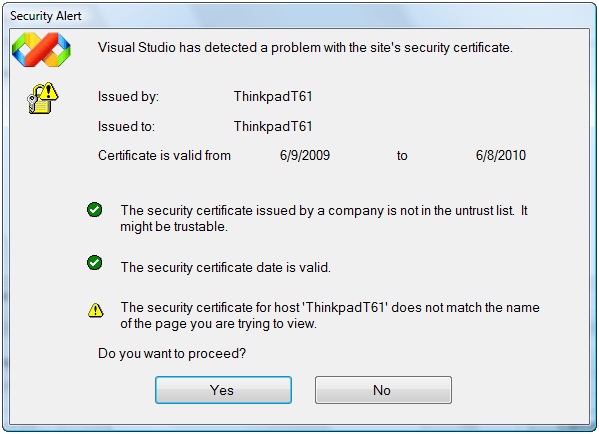
  </bindings>

  <services>

    <service name="SecureServiceLibrary.SecureService"

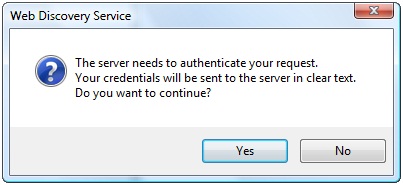
Save your changes and select Build | Build Web Site.

You are now ready to complete the client application. Return to Visual Studio. In the Solution Explorer window, right-click on the WindowsClient project and select Add Service Reference to display the Add Service Reference dialog box. Enter https://localhost/InternalWebHostedServiceDemo/SecureService.svc in the Address text box and click Go. Visual Studio displays the security alert shown in Figure 9.

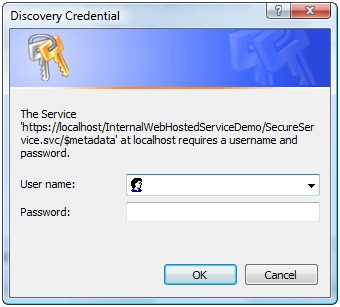


**Figure 9. Your use of a self-signed certificate generates this security alert.**

Visual Studio displays this message because the WCF runtime does not trust self-signed certificates. Ordinarily, you would not continue at this point, but for the purposes of this tutorial, click Yes to continue. Visual Studio displays the message shown in Figure 10. You have configured the service to use basic authentication, which requires valid login credentials. Click Yes. Visual Studio displays the Discovery Credential dialog box (see Figure 11). Enter your user name and password and click OK.



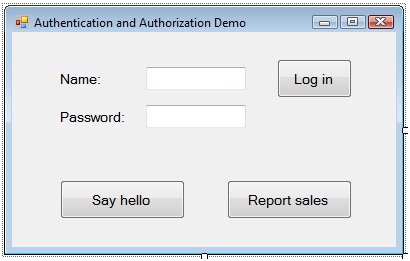
**Figure 10. You need to pass your credentials to the service.**



**Figure 11. Enter your name and password to authenticate your call to the service.**

In the Services pane of the Add Service Reference dialog box, select SecureService. Enter SecureService in the Namespace text box and click OK to add the service reference.

In the Solution Explorer, double-click Form1. The form (see Figure 12) has textboxes for user name and password. When the user clicks Log in, the form’s code will create a new instance of the service proxy class.



**Figure 12. This is the sample application’s form.**

Double-click the Log in button and add the following code to the logInButton\_Click method:

// C#

proxy = new SecureServiceClient("WSHttpBinding\_ISecureService");

proxy.ClientCredentials.UserName.UserName = nameTextBox.Text;

proxy.ClientCredentials.UserName.Password = passwordTextBox.Text;

' Visual Basic

proxy = New SecureServiceClient("WSHttpBinding\_ISecureService")

proxy.ClientCredentials.UserName.UserName = nameTextBox.Text

proxy.ClientCredentials.UserName.Password = passwordTextBox.Text

This code passes to the service the user name and password as credentials. If the credentials are not valid, the service will throw an exception. Add the following code in bold to the sayHelloButton\_Click method to catch this exception:

// C#

try

{

  MessageBox.Show(proxy.SayHello());

}

catch (MessageSecurityException)

{

  MessageBox.Show("You do not have permission to call this service");

}

catch (FaultException faultEx)

{

  MessageBox.Show(faultEx.Message);

}

' Visual Basic

Try

  MessageBox.Show(proxy.SayHello())

Catch messageSecurityEx As MessageSecurityException

  MessageBox.Show("You do not have permission to call this service")

Catch faultEx As FaultException

  MessageBox.Show(faultEx.Message)

 Add the following code in bold to the reportSalesButton\_Click method to catch the exception:

// C#

try

{

  decimal sales = proxy.ReportSales();

  if (sales != -1)

  {

    MessageBox.Show(String.Format(

      "Today's sales are {0:C}", sales));

  }

  else

  {

    MessageBox.Show(

      "You are not authorized to view sales numbers");

  }

}

catch (MessageSecurityException)

{

  MessageBox.Show("You do not have permission to call this service");

}

catch (SecurityAccessDeniedException securityEx)

{

  MessageBox.Show(securityEx.Message);

}

' Visual Basic

Try

  Dim sales As Decimal = proxy.ReportSales

  If sales <> -1 Then

    MessageBox.Show(String.Format( \_

      "Today's sales are {0:C}", sales))

  Else

    MessageBox.Show( \_

      "You are not authorized to view sales numbers")

  End If

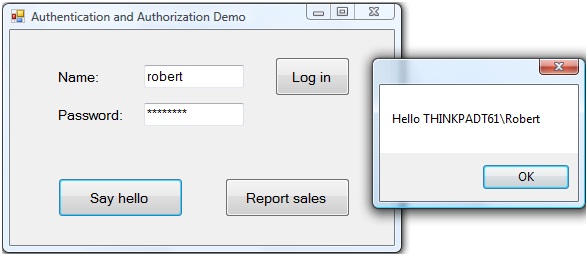
Catch messageSecurityEx As MessageSecurityException

  MessageBox.Show("You do not have permission to call this service")

Catch securityEx As SecurityAccessDeniedException

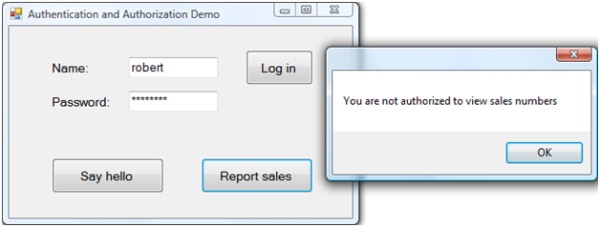
  MessageBox.Show(securityEx.Message)

Save your changes and build the solution. Press F5 to run the application. Enter your user name and password and click Log in. Click Say hello. The application should display a message similar to the one shown in Figure 13, but with your logged in user name.



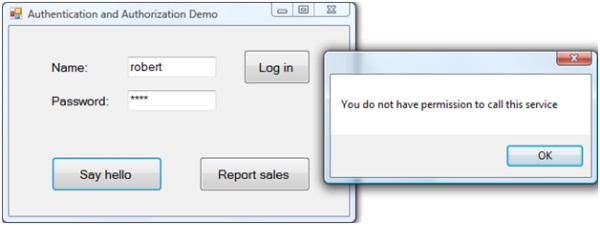
**Figure 13. The service returns to the client a greeting with your name.**

Click OK to dismiss the message. Click Report sales. The application should display the message shown in Figure 14.



**Figure 14. You are not authorized to see the sales number.**

You receive this message because the service is configured to only allow members of the Windows built-in Backup Operators group to view sales numbers. Click OK to dismiss the message. Enter a password other than your correct one and click Log in. Click Say hello. The application should display the message shown in Figure 15.



**Figure 15. You do not have permission to call the service.**

Click OK to dismiss the message. Close the form.

### Control Access to and Usage of Public Web-Hosted Services

You just explored how to control access to a WCF service hosted internally using Internet Information Services. In the final scenario in this tutorial, you will explore how to control access to a service hosted externally. You will still require users to log in and pass credentials. If the users of the service all have network logins, then you can use the same techniques you just saw. The only difference would be the address of the WCF service.

However, suppose you want your service to be publicly available but only to customers or subscribers. Further, suppose you want to group your users into roles to control who can access what code. You can use ASP.NET membership and roles to accomplish this. You can store user names, passwords and roles in a SQL Server database. You can then configure WCF to use the ASP.NET membership and roles providers to validate user names and passwords and to see if users are in particular roles.

This tutorial assumes you have configured SQL Server to work with these providers. If you have not, see the Creating the Application Services Database for SQL Server Help topic (http://msdn.microsoft.com/en-us/library/x28wfk74.aspx).

To explore authentication and authorization in this scenario, select File | Open | Project/Solution to display the Open Project dialog box. Navigate to the folder where you downloaded this tutorial’s sample project. Navigate to the PublicWebHostedServiceDemo folder. Select PublicWebHostedServiceDemo.sln and click OK to open the project. The sample application includes two projects. The SecureServiceLibrary project contains the same WCF service you saw in the previous two scenarios.

You will next create a Web site to host the service. In the Solution Explorer, right-click the PublicWebHostedServiceDemo solution and select Add | New Web Site to display the Add New Web Site dialog box. Select WCF Service in the Templates list. Select HTTP from the Location drop-down list. Select Visual C# or Visual Basic from the Language drop-down list. In the Location text box, enter http://localhost/PublicWebHostedServiceDemo. You will not use SSL here, so you do not need to specify https in the Web site address. Click OK.

In the Solution Explorer, expand the App\_Code folder under the new web site project you just created. Delete the IService and Service files. In the Solution Explorer, right-click the PublicWebHostedServiceDemo Web site, and select Add Reference from the context menu. In the Projects tab of the Add Reference dialog box, select SecureServiceLibrary. Click OK to add the reference to the service.

Rename the Service.svc file to SecureService.svc. Open the file by double clicking on the file name in the Solution Explorer. Since you are referencing an existing service, change the contents of this file to the following:

<%@ ServiceHost Service="SecureServiceLibrary.SecureService" %>

In the Solution Explorer, double-click Web.config. In the system.web section, make the following change in bold to enable debugging of the service at runtime:

// C#

<compilation debug="true" strict="false" explicit="true">

' Visual Basic

<compilation debug="true">

Scroll down to the system.serviceModel section. Make the following changes in bold:

<service name="SecureServiceLibrary.SecureService"

         behaviorConfiguration="ServiceBehavior">

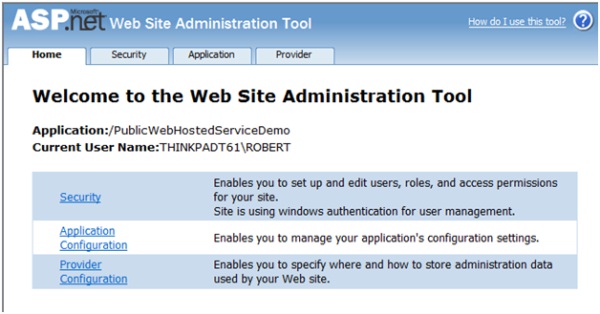
  <!-- Service Endpoints -->

  <endpoint address="" binding="wsHttpBinding"

            contract="SecureServiceLibrary.ISecureService">

#### Configure Membership and Roles

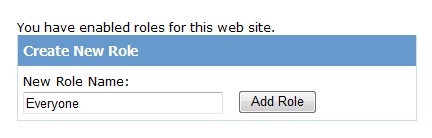
You will now configure the Web site to support ASP.NET membership and roles. In the Solution Explorer, select the PublicWebHostedServiceDemo Web site. Select Website | ASP.NET Configuration. This opens the Web Site Administration Tool in the browser (see Figure 16).



**Figure 16. Use the Web Site Administration Tool to configure security for the Web site.**

Select the Security link. Select the Use the security Setup Wizard to configure security step by step link to start the Security Setup Wizard. On the Welcome page of the wizard, click Next. In Step 2, select From the internet to use Forms based authentication. Click Next. The wizard informs you that your application is configured to use advanced provider settings. This means the application will store user names, passwords and roles in a SQL Server Express database. Click Next.

In Step 4 of the wizard, check Enable roles for this Web site. Click Next. Enter Everyone in the New Role Name text box and click Add Role (see Figure 17). To create a second role, enter Member in the New Role Name text box and click Add Role. Click Next.



**Figure 17. Create a new role by entering its name.**

In Step 5 of the wizard, you will create a new user. Enter Robert as the user name and P@ssword1 as the password. You must enter an e-mail address and a security question and answer. You can enter anything you like. Click Create User to add the user and when you see the confirmation page, click Continue.



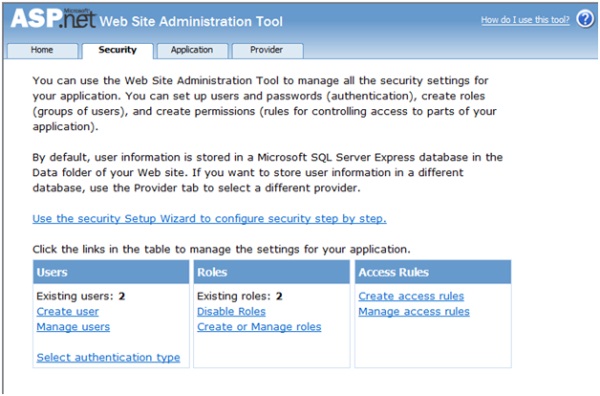
**Figure 18. Create a new user.**

Enter a second user with the name Manny. Use the same password. Click Create User to add the user and when you see the confirmation page, click Next.

In Step 6 of the wizard, you will deny access to anonymous users. Anyone wishing to call the WCF service must log in. Select Anonymous Users in the Rule applies to list. In the Permission list, select Deny (see Figure 19). Click Add This Rule to create the rule. Click Next and then click Finish. The Web Site Administration Tool should look like Figure 20.

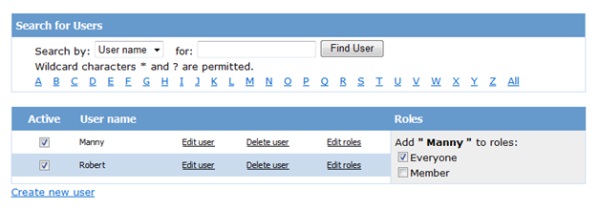


**Figure 19. Anonymous users will not be able to call the WCF service.**



**Figure 20. The Web Site Administration Tool should look like this.**

You will next assign users to roles. Select the Manage users link. Select the Edit roles link associated with Manny. Check Everyone to assign Manny to that role (see Figure 21).



**Figure 21. Assign a user to a role.**

Select the Edit roles link associated with Robert. Check Everyone and Member to assign Robert to those roles. Click Back. Close the browser.

Return to Visual Studio. In the Web.config file, add the following XML in bold to configure the WCF service to expect that clients pass a user name and password to the service:

<system.serviceModel>

  <bindings>

    <wsHttpBinding>

      <binding name="WsHttpBindingConfig">

        <security mode="TransportWithMessageCredential">

          <transport clientCredentialType="None" />

          <message clientCredentialType="UserName" />

        </security>

      </binding>

    </wsHttpBinding>

  </bindings>

  <services>

    <service name="SecureServiceLibrary.SecureService"

             behaviorConfiguration="ServiceBehavior">

      <!-- Service Endpoints -->

      <endpoint address="" binding="wsHttpBinding"

                bindingConfiguration="WsHttpBindingConfig"

                contract="SecureServiceLibrary.ISecureService">

Next, add the following XML in bold to configure the WCF service to use the ASP.NET membership provider to authenticate users and to use the ASP.NET roles provider to determine what roles users have so you can control their actions:

  <behaviors>

    <serviceBehaviors>

      <behavior name="ServiceBehavior">

        <serviceMetadata httpGetEnabled="true"/>

        <serviceDebug includeExceptionDetailInFaults="false"/>

        <serviceCredentials>

          <userNameAuthentication

            userNamePasswordValidationMode="MembershipProvider"

            membershipProviderName="AspNetSqlMembershipProvider"/>

        </serviceCredentials>

        <serviceAuthorization

          principalPermissionMode="UseAspNetRoles"

          roleProviderName="AspNetSqlRoleProvider" />

      </behavior>

    </serviceBehaviors>

  </behaviors>

</system.serviceModel>

Now that you have configured the WCF service to use the ASP.NET membership and role providers, you will write the code in the service to control who can call the ReportSales method.

In the Solution Explorer, right click on the SecureServiceLibrary project and select Add Reference to display the Add Reference dialog box. In the Component Name list, select System.Web and click OK to add the reference.

In the Solution Explorer, double-click on the SecureService file. Add the following to the top of the code file:

// C#

using System.Security.Principal;

using System.Web.Security;

' Visual Basic

Imports System.Security.Principal

Imports System.Web.Security

Replace the code in the ReportSales method with the following:

// C#

IIdentity currentUser =

  ServiceSecurityContext.Current.PrimaryIdentity;

if (Roles.IsUserInRole(currentUser.Name, "Member"))

{

  return 10000M;

}

else

{

  return -1;

}

' Visual Basic

Dim currentUser As IIdentity = \_

  ServiceSecurityContext.Current.PrimaryIdentity

If Roles.IsUserInRole(currentUser.Name, "Member") Then

  Return 10000D

Else

  Return -1

End If

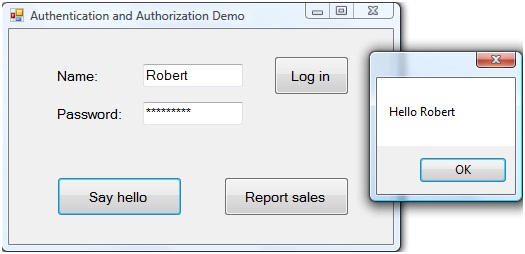
The ServiceSecurityContext class is in the System.ServiceModel namespace and represents the security context of the client. The PrimaryIdentity property is based on the credentials the client sends to the service. This property returns an instance of the IIdentity interface and represents the user on whose behalf the code is running.

The Roles class is in the System.Web namespace and manages users in roles. The code above uses the IsInRole method to check if the user is in the Member role. If so, the method returns today’s sales.

Save your changes. In the Solution Explorer, select the Web site and select Build | Build Web Site.

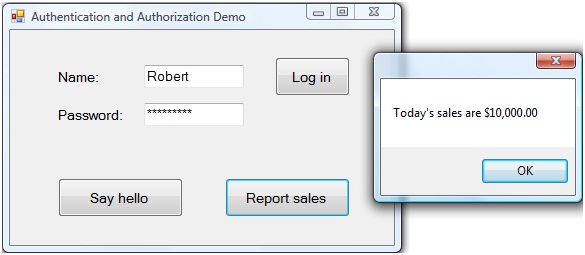
You are now ready to complete the client application. In the Solution Explorer window, right-click on the WindowsClient project and select Add Service Reference to display the Add Service Reference dialog box. Enter http://localhost/PublicWebHostedServiceDemo/SecureService.svc in the Address text box and click Go. In the Services pane of the Add Service Reference dialog box, select SecureService. Enter SecureService in the Namespace text box and click OK to add the service reference.

Save your changes and build the solution. Press F5 to run the application. Enter Robert in the Name text box and P@ssword1 in the Password text box and click Log in. Click Say hello. The application should display a message shown in Figure 22.



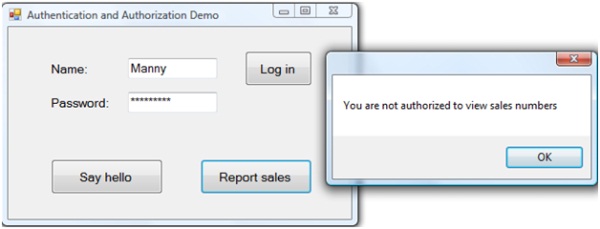
**Figure 22. The service returns to the client a greeting with the user’s name.**

Click OK to dismiss the message. Click Report sales. The application should display the message shown in Figure 23. Robert is in the Everyone and Member roles, so he can view sales numbers.



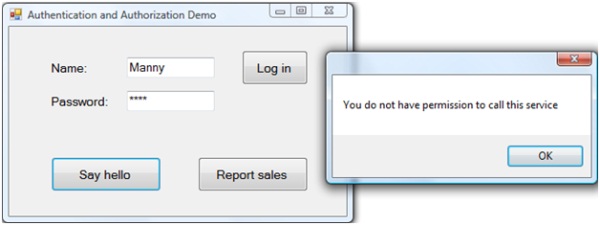
**Figure 23. The user is authorized to see the sales number.**

Enter Manny in the Name text box and P@ssword1 in the Password text box and click Log in. Click Report sales. The application should display the message shown in Figure 24. Manny is not in the Member role, so he cannot view sales numbers.



**Figure 24. The user is not authorized to see the sales number.**

Click OK to dismiss the message. Enter a password other than the correct one in the Password text box and click Log in. Click Say hello. The application should display the message shown in Figure 25.



**Figure 25. You do not have permission to call the service.**

Click OK to dismiss the message. Close the form.

### Conclusion

In this tutorial, saw how to use authentication and authorization in two scenarios where you are hosting a WCF service using Internet Information Services. In the first scenario, you are hosting the service on the corporate intranet. Both employees and guests have access to the wireless network. You saw that you could use basic authentication and SSL to require users pass their user name and password to the service in order to call it. You then saw how to limit use of a method to certain users.

In the second scenario, you are hosting the service on the public Internet. You saw how to configure the service to use the ASP.NET membership provider to authenticate users. You then saw how to configure the service to use the ASP.NET role provider to limit use of a method to certain users.

The completed versions of this tutorial’s sample applications are included in the download accompanying this article. The app.config file in the WindowsClient project for the InternalWebHostedServiceDemo solution includes the following endpoint:

<endpoint

  address=

   "https://thinkpadt61/InternalWebHostedServiceDemo/SecureService.svc"

  binding="wsHttpBinding"

  bindingConfiguration="WSHttpBinding\_ISecureService"

  contract="SecureService.ISecureService"

  name="WSHttpBinding\_ISecureService">

The app.config file in the WindowsClient project for the PublicWebHostedServiceDemo solution includes the following endpoint:

<endpoint

  address=

   "https://thinkpadt61/PublicWebHostedServiceDemo/SecureService.svc"

  binding="wsHttpBinding"

  bindingConfiguration="WSHttpBinding\_ISecureService"

  contract="SecureService.ISecureService"

  name="WSHttpBinding\_ISecureService">

You can build the two solutions from scratch by following the steps in this tutorial. If you do that, your endpoints will include the name of your Web server computer. You can also use the completed versions of these solutions. If you do that, you will need to replace thinkpadt61 with the name of your Web server computer.

For additional information on securing WCF services, you should review the Microsoft patterns & practices Improving Web Services Security Guide at http://www.codeplex.com/WCFSecurityGuide.

**How to: Authenticate with a User Name and Password**

**.NET Framework 4.5**

[Other Versions](javascript:;)

http://i3.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635533092500576233

4 out of 9 rated this helpful - [Rate this topic](http://msdn.microsoft.com/en-us/library/ms733131%28v=vs.110%29.aspx#feedback)

This topic demonstrates how to enable a Windows Communication Foundation (WCF) service to authenticate a client with a Windows domain username and password. It assumes you have a working, self-hosted WCF service. For an example of creating a basic self-hosted WCF service see, [Getting Started Tutorial](http://msdn.microsoft.com/en-us/library/ms734712%28v=vs.110%29.aspx). This topic assumes the service is configured in code. If you would like to see an example of configuring a similar service using a configuration file see [Message Security User Name](http://msdn.microsoft.com/en-us/library/ms752233%28v=vs.110%29.aspx)

To configure a service to authenticate its clients using Windows Domain username and passwords use the [WSHttpBinding](http://msdn.microsoft.com/en-us/library/system.servicemodel.wshttpbinding%28v=vs.110%29.aspx) and set its Security.Mode property to Message. In addition you must specify an X509 certificate that will be used to encrypt the username and password as they are sent from the client to the service.

On the client, you must prompt the user for the username and password and specify the user’s credentials on the WCF client proxy.

**To configure a WCF service to authenticate using Windows domain username and password.**

1. Create an instance of the [WSHttpBinding](http://msdn.microsoft.com/en-us/library/system.servicemodel.wshttpbinding%28v=vs.110%29.aspx), set the security mode of the binding to SecurityMode.Message, set the ClientCredentialType of the binding to MessageCredentialType.UserName, and add a service endpoint using the configured binding to the service host as shown in the following code:
2. // ...
3. WSHttpBinding userNameBinding = new WSHttpBinding();
4. userNameBinding.Security.Mode = SecurityMode.Message;
5. userNameBinding.Security.Message.ClientCredentialType = MessageCredentialType.UserName;
6. svcHost.AddServiceEndpoint(typeof(IService1), userNameBinding, "");
7. // ...
8. Specify the server certificate used to encrypt the username and password information sent over the wire. This code should immediately follow the code above. The following example uses the certificate that is created by the setup.bat file from the [Message Security User Name](http://msdn.microsoft.com/en-us/library/ms752233%28v=vs.110%29.aspx) sample:
9. // ...
10. svcHost.Credentials.ServiceCertificate.SetCertificate(StoreLocation.LocalMachine, StoreName.My, X509FindType.FindBySubjectName, "localhost");
11. // ...

You can use your own certificate, just modify the code to refer to your certificate. For more information about creating and using certificates see [Working with Certificates](http://msdn.microsoft.com/en-us/library/ms731899%28v=vs.110%29.aspx). Make sure the certificate is in the Trusted People certificate store for the Local Machine. You can do this by running mmc.exe and selecting the File, Add/Remove Snap-in... menu item. In the Add or Remove Snap-ins dialog, select the Certificates snap-in and click Add. In the Certificates Snap-in dialog select Computer account. By default the certificate generated from the Message Security User name sample will be located in the Personal/Certificates folder. It will be listed as “localhost” under the Issued to column in the MMC window. Drag and drop the certificate into the Trusted People folder. This will allow WCF to treat the certificate as a trusted certificate when performing authentication.

**To call the service passing username and password**

1. The client application must prompt the user for their username and password. The following code asks the user for username and password.

|  |
| --- |
| **Caution noteCaution** |
| This code should not be used in production as the password is displayed while being entered. |

1. public static void GetPassword(out string username, out string password)
2. {
3. Console.WriteLine("Provide a valid machine or domain account. [domain\\user]");
4. Console.WriteLine(" Enter username:");
5. username = Console.ReadLine();
6. Console.WriteLine(" Enter password:");
7. password = Console.ReadLine();
8. return;
9. }
10. Create an instance of the client proxy specifying the client’s credentials as shown in the following code:
11. string username;
12. string password;
13. // Instantiate the proxy
14. Service1Client proxy = new Service1Client();
15. // Prompt the user for username & password
16. GetPassword(out username, out password);
17. // Set the user’s credentials on the proxy
18. proxy.ClientCredentials.UserName.UserName = username;
19. proxy.ClientCredentials.UserName.Password = password;
21. // Treat the test certificate as trusted
22. proxy.ClientCredentials.ServiceCertificate.Authentication.CertificateValidationMode = System.ServiceModel.Security.X509CertificateValidationMode.PeerOrChainTrust;
23. // Call the service operation using the proxy