**Mutual Certificate Authentication for WCF REST services**

Thursday, April 16, 2009 [REST](http://weblogs.asp.net/cibrax/Tags/REST) [REST Starter Kit](http://weblogs.asp.net/cibrax/Tags/REST%20Starter%20Kit) [WCF](http://weblogs.asp.net/cibrax/Tags/WCF)

When Mutual Certificate Authentication is configured for REST services, both, the client and the service perform identity verification or authentication through X509 certificates.

The client authenticates the service during the initial SSL handshake, when the server sends the client a certificate to authenticate itself. More information about this process can be [found here](http://support.microsoft.com/default.aspx/kb/257587).

The service on the other hand, performs similar validations on the certificate attached by the client consumer to the request message.

Some specific configuration settings are required in WCF to use this authentication scenario with REST services. The service must be configured with transport security (SSL), “certificate” as client credential type.

<webHttpBinding>

  <binding name="mutual">

    <security mode="Transport">

      <transport clientCredentialType="Certificate"/>

    </security>   </binding>

</webHttpBinding>

For example, the complete configuration for a REST service that provides ATOM feeds would be the following,

<system.serviceModel>

  <serviceHostingEnvironment aspNetCompatibilityEnabled="true"/>

  <services>

    <service name="MutualAuthenticationRest.FeedService">

      <endpoint address="" contract="MutualAuthenticationRest.IFeedService" binding="webHttpBinding" bindingConfiguration="mutual" behaviorConfiguration="mutual"></endpoint>

     </service>

  </services>

  <bindings>

    <webHttpBinding>

      <binding name="mutual">

        <security mode="Transport">

          <transport clientCredentialType="Certificate"/>

         </security>

       </binding>

     </webHttpBinding>

    </bindings>

    <behaviors>

      <endpointBehaviors>

        <behavior name="mutual">

          <webHttp/>

        </behavior>

     </endpointBehaviors>

   </behaviors>

</system.serviceModel>

The virtual directory used for hosting the service in IIS must also be configured with the same security settings.

The client application only need to attach the certificate at the moment of consuming the service. This can be easily done with the new HttpClient class shipped as part of the WCF REST Starter kit,

class Program

{

    static void Main(string[] args)

    {

        X509Certificate2 certificate = CertificateUtil.GetCertificate(StoreName.My, StoreLocation.LocalMachine, "CN=clientCert");

        HttpClient client = new HttpClient();

        client.TransportSettings.ClientCertificates.Add(certificate); //Cert attached to the request

        string content = client.Get("https://localhost/MutualAuthentication/Service.svc?numItems=10").Content.ReadAsString();

        Console.WriteLine(content);

    }

}

CertificateUtil is an helper class that looks for an specific X509 certificate in the windows certificate store. In addition, if you are using test certificates, you might want to disable the service authentication on the client side, that can be done with a callback attached to the static class “ServicePointManager”.

For instance,

public class PermissiveCertificatePolicy

{

    public static void Enable()

    {

        ServicePointManager.ServerCertificateValidationCallback +=

           new RemoteCertificateValidationCallback(RemoteCertValidate);

    }

    static bool RemoteCertValidate(object sender, X509Certificate cert, X509Chain chain, System.Net.Security.SslPolicyErrors error)

    {

        return true;

    }

}

The method PermissiveCertificatePolicy.Enable should be called before consuming the service.

///// Accept SSL Certification and Self Sign Certification

ServicePointManager.ServerCertificateValidationCallback +=

(sender, cert, chain, sslPolicyErrors) => true;

HttpClient htp\_client = new HttpClient();

/////////////////////////////////////////////////////////