|  |
| --- |
| AsymmetricSecurity binding in WCF |
| WS-Security Interoperability with JAVA Service |
| Calling a java service by using the C# with X509 Certificate at the client side, It includes the calling the service by adding the WSDL provided by the java service. SOAP request with WS-Security tags with SOAP 1.1v format. |

Author

Chaitanya Krishna Kolla

10/21/2013

Contents

* [What is WS-Security? 2](#_Toc369879444)
* [What is Asymmetric binding? 2](#_Toc369879445)
* [Sample Request 3](#_Toc369879446)
* [Sample Response 4](#_Toc369879447)
* [Steps to be followed format the request and response: 5](#_Toc369879448)
  + [Step1: Create server certificate 5](#_Toc369879449)
  + [Step 2: Create client certificate 5](#_Toc369879450)
  + [Step 3: Operation Contract 5](#_Toc369879451)
  + [Step 4: Injection of the Symmetric Binding Element 5](#_Toc369879452)
  + [Step 5: Service Configuration including the diagnostics 6](#_Toc369879453)
  + [Step 6: Add Proxy to client and client configuration 8](#_Toc369879454)

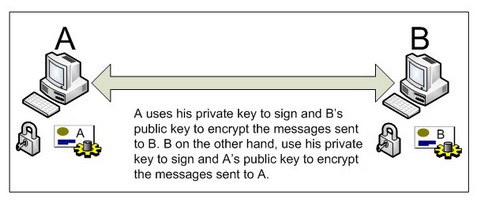
# What is WS-Security?

These standards cover many aspects of security, including digital signatures, authentication and encryption of SOAP messages. The generic name for the standards is WS-\*, and includes WS-Security, WS-Trust and WS-SecureConversation.

# What is Asymmetric binding?

According to the WS Security Specification, “The AsymmetricBinding assertion is used in scenarios in which message protection is provided by means defined in WSS: SOAP Message Security using asymmetric key (Public Key) technology”. If we put this in a simplified manner, Asymmetric Binding can be used when both parties possess key pairs. For example, if both the parties possess X.509 certificates, then it is possible to use asymmetric binding.

In asymmetric binding, message encryption and signing takes place using the Public Key Infrastructure(PKI), i.e. sender encrypts messages using the public key of the recipient and sign the messages using his private key. Then the recipient can decrypt the received messages using his private key and verify the signature of the message using the public key of the sender. This way, the confidentiality, integrity and the non-repudiation properties of the message exchanges can be assured.



# Sample Request

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

<s:Header>

<VsDebuggerCausalityData xmlns="http://schemas.microsoft.com/vstudio/diagnostics/servicemodelsink">uIDPo9BrQHxKdRBHnk55BHEVIegAAAAA9xEF2l05pE6oORfcQbIUtu/UUlm2aa1MoqgTA1LhHoAACQAA</VsDebuggerCausalityData>

<o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">

<u:Timestamp u:Id="uuid-6d16b382-19a0-4ab9-a6e0-b8a9e45934e7-1">

<u:Created>2013-10-18T10:44:48.021Z</u:Created>

<u:Expires>2013-10-18T10:49:48.021Z</u:Expires>

</u:Timestamp>

<o:BinarySecurityToken u:Id="uuid-fc380a85-c6d5-4d2f-802f-9408673f083d-2" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary">MIICBjCCAXOgAwIBAgIQSsDqfgBbIJxOmxixpri/</o:BinarySecurityToken>

<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

<SignedInfo>

<CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>

<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>

<Reference URI="#\_1">

<Transforms>

<Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>

</Transforms>

<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

<DigestValue>4AmVSd/mVml+VYVtYgAT8dhYnVE=</DigestValue>

</Reference>

<Reference URI="#uuid-6d16b382-19a0-4ab9-a6e0-b8a9e45934e7-1">

<Transforms>

<Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>

</Transforms>

<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

<DigestValue>Ij2dZ5HA5xAw6/5GwVqA+vqoCc8=</DigestValue>

</Reference>

</SignedInfo>

<SignatureValue>US78jxH5RkyFfPOCM05TDXvuwmk7MM6uPRluMXl9SlBNiigEf2R8BouwyHB8ODnVK3D84sPPrxttaapeVYDc2wCThWqtKS+Rf9+tETaucYxouttBaiarc6DE9k3FifU4LeeUBQz6nwpJJY1zKm1NviwwYwEvYTZy8ZU6rurzgqQ=</SignatureValue>

<KeyInfo>

<o:SecurityTokenReference>

<o:Reference ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" URI="#uuid-fc380a85-c6d5-4d2f-802f-9408673f083d-2"/>

</o:SecurityTokenReference>

</KeyInfo>

</Signature>

</o:Security>

</s:Header>

<s:Body u:Id="\_1">

<DoSomething xmlns="http://test.com/play">

<wow>Salutations</wow>

</DoSomething>

</s:Body>

</s:Envelope>

# Sample Response

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">

<s:Body>

<DoSomethingResponse xmlns="http://test.com/play">

<DoSomethingResult>yo</DoSomethingResult>

</DoSomethingResponse>

</s:Body>

</s:Envelope>

# Steps to be followed format the request and response:

## Step1: Create server certificate

makecert.exe -r -pe -n "CN=localhost" -ss my -sr LocalMachine -sky exchange -sp "Microsoft RSA SChannel Cryptographic Provider" -sy 12 LocalhostTestKey.cer

## Step 2: Create client certificate

makecert.exe -r -pe -n "CN=BradTestClientKey" -ss my -sr CurrentUser -sky exchange -sp "Microsoft RSA SChannel Cryptographic Provider" -sy 12 BradTestClientKey.cer

## Step 3: Operation Contract

[ServiceContract(Namespace = “http://test.com/play", ProtectionLevel=ProtectionLevel.Sign)]

interface ISer

{

[OperationContract]

string DoSomething(string wow);

}

## Step 4: Injection of the Symmetric Binding Element

using System;

using System.Collections.Generic;

using System.Text;

using System.ServiceModel;

using System.ServiceModel.Channels;

using System.ServiceModel.Dispatcher;

using System.ServiceModel.Description;

using System.ServiceModel.Security;

using System.ServiceModel.Security.Tokens;

using System.ServiceModel.Configuration;

namespace MySecurityBE

{

public class AsymetricSecurityBE : BindingElement

{

private AsymmetricSecurityBindingElement m\_asymSecBE;

public AsymetricSecurityBE()

{

MessageSecurityVersion securityVersion = MessageSecurityVersion.WSSecurity10WSTrustFebruary2005WSSecureConversationFebruary2005WSSecurityPolicy11BasicSecurityProfile10;

SecurityBindingElement securityBE = SecurityBindingElement.CreateMutualCertificateBindingElement(securityVersion, true);

securityBE.IncludeTimestamp = true;

securityBE.SetKeyDerivation(false);

securityBE.SecurityHeaderLayout = SecurityHeaderLayout.Lax;

securityBE.EnableUnsecuredResponse = true;

m\_asymSecBE = securityBE as AsymmetricSecurityBindingElement;

}

public AsymetricSecurityBE(AsymetricSecurityBE other)

{

m\_asymSecBE = other.m\_asymSecBE;

}

public override IChannelListener<TChannel> BuildChannelListener<TChannel>(BindingContext context)

{

return m\_asymSecBE.BuildChannelListener<TChannel>(context);

}

public override IChannelFactory<TChannel> BuildChannelFactory<TChannel>(BindingContext context)

{

return m\_asymSecBE.BuildChannelFactory<TChannel>(context);

}

public override BindingElement Clone()

{

AsymetricSecurityBE ret = new AsymetricSecurityBE(this);

return ret;

}

public override T GetProperty<T>(BindingContext context)

{

return m\_asymSecBE.GetProperty<T>(context);

}

}

class AsymetricSecurityBEExtentionElement : BindingElementExtensionElement

{

public override Type BindingElementType

{

get { return typeof(AsymetricSecurityBE); }

}

protected override BindingElement CreateBindingElement()

{

return new AsymetricSecurityBE();

}

}

}

## Step 5: Service Configuration including the diagnostics

configuration>

<system.serviceModel>

<extensions>

<bindingElementExtensions>

<add name="MySecurityBindingElement" type="MySecurityBE.AsymetricSecurityBEExtentionElement, MySecurityBE, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null" />

</bindingElementExtensions>

</extensions>

<bindings>

<customBinding>

<binding name="MyCoolBinding">

<MySecurityBindingElement/>

<textMessageEncoding messageVersion="Soap11"/>

<httpTransport/>

</binding>

</customBinding>

</bindings>

<services>

<service behaviorConfiguration="ServBeh" name="Service.MyImpl">

<host>

<baseAddresses>

<add baseAddress="http://localhost:8000" />

</baseAddresses>

</host>

<endpoint address="MyCoolEndpoint" binding="customBinding" bindingConfiguration="MyCoolBinding"

name="MyCoolEndpoint" contract="Service.ISer">

</endpoint>

<endpoint name="MetadataHttp" address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="ServBeh">

<serviceMetadata httpGetEnabled="true" />

<serviceDebug includeExceptionDetailInFaults="true" />

<serviceCredentials>

<clientCertificate>

<authentication certificateValidationMode="None" revocationMode="NoCheck" />

</clientCertificate>

<serviceCertificate storeLocation="LocalMachine"

storeName="My"

x509FindType="FindBySubjectName"

findValue="LocalhostTestKey"/>

<issuedTokenAuthentication certificateValidationMode="PeerOrChainTrust" />

</serviceCredentials>

</behavior>

</serviceBehaviors>

</behaviors>

<diagnostics>

<messageLogging maxMessagesToLog="30000"

logEntireMessage="true"

logMessagesAtServiceLevel="false"

logMalformedMessages="true"

logMessagesAtTransportLevel="true">

<filters>

<clear/>

</filters>

</messageLogging>

</diagnostics>

</system.serviceModel>

<system.diagnostics>

<sources>

<source name="System.ServiceModel" switchValue="Warning, ActivityTracing" >

<listeners>

<add name="xml" />

</listeners>

</source>

<source name="System.ServiceModel.MessageLogging" switchValue="Warning">

<listeners>

<add name="xml" />

</listeners>

</source>

</sources>

<sharedListeners>

<add name="xml" type="System.Diagnostics.XmlWriterTraceListener" initializeData="..\Logs\Server.svclog" />

</sharedListeners>

<trace autoflush="true" />

</system.diagnostics>

</configuration>

## Step 6: Add Proxy to client and client configuration

<configuration>

<system.serviceModel>

<extensions>

<bindingElementExtensions>

<add name="MySecurityBindingElement" type="MySecurityBE.AsymetricSecurityBEExtentionElement, MySecurityBE, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null" />

</bindingElementExtensions>

</extensions>

<bindings>

<customBinding>

<binding name="MyCoolBinding">

<MySecurityBindingElement/>

<textMessageEncoding messageVersion="Soap11" />

<httpTransport/>

</binding>

</customBinding>

</bindings>

<behaviors>

<endpointBehaviors>

<behavior name="cliBeh">

<clientCredentials>

<clientCertificate storeLocation="CurrentUser" storeName="My" x509FindType="FindBySubjectName" findValue="BradTestClientKey"/>

<serviceCertificate>

<defaultCertificate storeLocation="CurrentUser" storeName="My" x509FindType="FindBySubjectName" findValue="localhost"/>

<authentication certificateValidationMode="None"

trustedStoreLocation="CurrentUser" />

</serviceCertificate>

</clientCredentials>

</behavior>

</endpointBehaviors>

</behaviors>

<client>

<endpoint address="http://localhost:8000/MyCoolEndpoint" binding="customBinding" bindingConfiguration="MyCoolBinding"

name="MyCoolEndpoint" contract="ISer" behaviorConfiguration="cliBeh">

<identity>

<dns value="LocalhostTestKey"/>

</identity>

</endpoint>

</client>

<diagnostics>

<messageLogging logEntireMessage="true"

logMalformedMessages="true"

logMessagesAtServiceLevel="true"

logMessagesAtTransportLevel="true"

maxMessagesToLog="3000"

maxSizeOfMessageToLog="2000"/>

</diagnostics>

</system.serviceModel>

<system.diagnostics>

<sources>

<source name="System.ServiceModel.MessageLogging">

<listeners>

<add name="messages" type="System.Diagnostics.XmlWriterTraceListener" initializeData="..\Logs\Client.svclog"/>

</listeners>

</source>

</sources>

</system.diagnostics>

</configuration>

# Conclusion

We have achieved the sample for the configuration

Server: Service behavior with X.509 certificate implementation.

Client: Sending authenticated request with X.509 certificate as signature to each request.