Comparing between Message and Transport Security

Windows Communication Foundation (WCF) has two major modes for providing security (Transport and Message) and a third mode (TransportWithMessageCredential) that combines the two. This topic discusses message security and the reasons to use it.

What Is Message Security?

Message security uses the WS-Security specification to secure messages. The WS-Securityspecification describes enhancements to SOAP messaging to ensure confidentiality, integrity, and authentication at the SOAP message level (instead of the transport level).

In brief, message security differs from transport security by encapsulating the security credentials and claims with every message along with any message protection (signing or encryption). Applying the security directly to the message by modifying its content allows the secured message to be self-containing with respect to the security aspects. This enables some scenarios that are not possible when transport security is used.

Reasons to Use Message Security

In message-level security, all of the security information is encapsulated in the message. Securing the message with message-level security instead of transport-level security has the following advantages:

- End-to-end security. Transport security, such as Secure Sockets Layer (SSL) only secures messages when the communication is point-to-point. If the message is routed to one or more SOAP intermediaries (for example a router) before reaching the ultimate receiver, the message itself is not protected once an intermediary reads it from the wire. Additionally, the client authentication information is available only to the first intermediary and must be re-transmitted to the ultimate receiver in out-of-band fashion, if necessary. This applies even if the entire route uses SSL security between individual hops. Because message security works directly with the message and secures the XML in it, the security stays with the message regardless of how many intermediaries are involved before it reaches the ultimate receiver. This enables a true end-to-end security scenario.
- Increased flexibility. Parts of the message, instead of the entire message, can be signed or encrypted. This means that intermediaries can view the parts of the message that are intended for them. If the sender needs to make part of the information in the message visible to the intermediaries but wants to ensure that it is not tampered with, it can just sign it but leave it unencrypted. Since the signature is part of the message, the ultimate receiver can verify that the information in the message was received intact. One scenario might have a SOAP intermediary service that routes message according the Action header value. By default, WCF does not encrypt the Action value but signs it if message security is used. Therefore, this information is available to all intermediaries, but no one can change it.

- Support for multiple transports. You can send secured messages over many different transports, such as named pipes and TCP, without having to rely on the protocol for security. With transport-level security, all the security information is scoped to a single particular transport connection and is not available from the message content itself. Message security makes the message secure regardless of what transport you use to transmit the message, and the security context is directly embedded inside the message.
- Support for a wide set of credentials and claims. The message security is based on the WS-Security specification, which provides an extensible framework capable of transmitting any type of claim inside the SOAP message. Unlike transport security, the set of authentication mechanisms, or claims, that you can use is not limited by the transport capabilities. WCF message security includes multiple types of authentication and claim transmission and can be extended to support additional types as necessary. For those reasons, for example, a federated credentials scenario is not possible without message security. For more information about federation scenarios

How Message and Transport Security Compare

Pros and Cons of Transport-Level Security

Transport security has the following advantages:

- Does not require that the communicating parties understand XML-level security concepts. This can improve the interoperability, for example, when HTTPS is used to secure the communication.
- Generally improved performance.
- Hardware accelerators are available.
- Streaming is possible.

Transport security has the following disadvantages:

- Hop-to-hop only.
- Limited and inextensible set of credentials.
- Transport-dependent.

Disadvantages of Message-Level Security

Message security has the following disadvantages:

- Performance
- Cannot use message streaming.
- Requires implementation of XML-level security mechanisms and support for WS-Security specification. This might affect the interoperability.

Transport Security Overview

Transport security mechanisms in Windows Communication Foundation (WCF) depend on the binding and transport being used. For example, when using the WSHttpBinding class, the transport is HTTP, and the primary mechanism for securing the transport is Secure Sockets Layer (SSL) over HTTP, commonly called HTTPS. This topic discusses the major transport security mechanisms used in the WCF system-provided bindings.

Note

When SSL security is used with .NET Framework 3.5 and later an WCF client uses both the intermediate certificates in its certificate store and the intermediate certificates received during SSL negotiation to perform certificate chain validation on the service's certificate. .NET Framework 3.0 only uses the intermediate certificates installed in the local certificate store.

Caution

When transport security is used, the CurrentPrincipal property may be overwritten. To prevent this from happening set the PrincipalPermission to None. ServiceAuthorizationBehavior is a service behavior that can be set on the service description.

BasicHttpBinding

By default, the BasicHttpBinding class does not provide security. This binding is designed for interoperability with Web service providers that do not implement security. However, you can switch on security by setting the Mode property to any value except None. To enable transport security, set the property to Transport.

Interoperation with IIS

The BasicHttpBinding class is primarily used to interoperate with existing Web services, and many of those services are hosted by Internet Information Services (IIS). Consequently, the transport security for this binding is designed for seamless interoperation with IIS sites. This is done by setting the security mode to Transport and then setting the client credential type. The credential type values correspond to IIS directory security mechanisms. The following code shows the mode being set and the credential type set to Windows. You can use this configuration when both client and server are on the same Windows domain.

C#

VB

```
BasicHttpBinding b = new BasicHttpBinding();
b.Security.Mode = BasicHttpSecurityMode.Transport ;
b.Security.Transport.ClientCredentialType = HttpClientCredentialType.Windows;
```

Or, in configuration:

Xml

```
<bindings>
  <basicHttpBinding>
    <binding name="SecurityByTransport">
      <security mode="Transport">
        <transport clientCredentialType="Windows" />
       </security>
     </binding>
  </basicHttpBinding>
</bindings>
```

The following sections discuss other client credential types.

Basic

This corresponds to the Basic authentication method in IIS. When using this mode, the IIS server must be configured with Windows user accounts and appropriate NTFS file system permissions. For more information about IIS 6.0, see Enabling Basic Authentication and Configuring the Realm Name. For more information about IIS 7.0, see IIS 7.0 Beta: Configure Basic Authentication.

Certificate

IIS has an option to require clients to log on with a certificate. The feature also enables IIS to map a client certificate to a Windows account. For more information about IIS 6.0, see Enabling Client Certificates in IIS 6.0. For more information about IIS 7.0, see IIS 7.0 Beta: Configuring Server Certificates in IIS 7.0.

Digest

Digest authentication is similar to Basic authentication, but offers the advantage of sending the credentials as a hash, instead of in clear text. For more information about IIS 6.0, see Digest Authentication in IIS 6.0. For more information about IIS 7.0, see IIS 7.0 Beta: Configure Digest Authentication.

Windows

This corresponds to integrated Windows authentication in IIS. When set to this value, the server is also expected to exist on a Windows domain that uses the Kerberos protocol as its domain controller. If the server is not on a Kerberos-backed domain, or if the Kerberos system fails, you can use the NT LAN Manager (NTLM) value described in the next section. For more information about IIS 6.0, see Integrated Windows Authentication in IIS 6.0. For more information about IIS 7.0, see IIS 7.0 Beta: Configuring Server Certificates in IIS 7.0.

NTLM

This enables the server to use NTLM for authentication if the Kerberos protocol fails. For more information about configuring IIS in IIS 6.0, see Forcing NTLM Authentication. For IIS 7.0, the Windows authentication includes NTLM authentication. For more information, see IIS 7.0 Beta: Configuring Server Certificates in IIS 7.0.

WsHttpBinding

The WSHttpBinding class is designed for interoperation with services that implement WS-* specifications. The transport security for this binding is Secure Sockets Layer (SSL) over HTTP, or HTTPS. To create an WCF application that uses SSL, use IIS to host the application. Alternatively, if you are creating a self-hosted application, use the HttpCfg.exe tool to bind an X.509 certificate to a specific port on a computer. The port number is specified as part of the WCF application as an endpoint address. When using transport mode, the endpoint address must include the HTTPS protocol or an exception will be thrown at run time. For more information, see HTTP Transport Security.

For client authentication, set the ClientCredentialType property of the HttpTransportSecurity class to one of the HttpClientCredentialType enumeration values. The enumeration values are identical to the client credential types for BasicHttpBinding and are designed to be hosted with IIS services.

The following example shows the binding being used with a client credential type of Windows.

C#

VB

```
// The code uses a shortcut to specify the security mode to Transport.
WSHttpBinding b = new WSHttpBinding(SecurityMode.Transport);
b.Security.Transport.ClientCredentialType = HttpClientCredentialType.Windows;
```

WSDualHttpBinding

This binding provides only message-level security, not transport-level security.

NetTcpBinding

The NetTcpBinding class uses TCP for message transport. Security for the transport mode is provided by implementing Transport Layer Security (TLS) over TCP. The TLS implementation is provided by the operating system.

You can also specify the client's credential type by setting the ClientCredentialType property of the TcpTransportSecurity class to one of the TcpClientCredentialType values, as shown in the following code.

C#

VB

```
NetTcpBinding b = new NetTcpBinding();
b.Security.Mode = SecurityMode.Transport;
b.Security.Transport.ClientCredentialType =
TcpClientCredentialType.Certificate;
```

Client

On the client, you must specify a certificate using the SetCertificate method of the X509CertificateInitiatorClientCredential class.

Note

If you are using Windows security, a certificate is not required.

The following code uses the thumbprint of the certificate, which uniquely identifies it. For more information about certificates, see Working with Certificates.

C#

VB

```
NetTcpBinding b = new NetTcpBinding();
b.Security.Mode = SecurityMode.Transport;
b.Security.Transport.ClientCredentialType =
TcpClientCredentialType.Certificate;
EndpointAddress a = new EndpointAddress("net.tcp://contoso.com/TcpAddress");
ChannelFactory<ICalculator> cf = new ChannelFactory<ICalculator>(b, a);
cf.Credentials.ClientCertificate.SetCertificate(
    StoreLocation.LocalMachine,
```

```
    StoreName.My,
    X509FindType.FindByThumbprint,
    "0000000000000000000000000000000000000000");
```

Alternatively, specify the certificate in the client's configuration using a <clientCredentials> element in the behaviors section.

Xml

```
<behaviors>
  <behavior>
   <clientCredentials>
     <clientCertificate findValue= "1010101010101010101010101010000000000"
      storeLocation="LocalMachine" storeName="My"
      X509FindType="FindByThumbPrint"/>
     </clientCertificate>
   </clientCredentials>
 </behavior>
</behaviors>
```

NetNamedPipeBinding

The NetNamedPipeBinding class is designed for efficient intra-machine communication; that is, for processes running on the same computer, although named pipe channels can be created between two computers on the same network. This binding provides only transport-level security. When creating applications using this binding, the endpoint addresses must include "net.pipe" as the protocol of the endpoint address.

WSFederationHttpBinding

When using transport security, this binding uses SSL over HTTP, known as HTTPS with an issued token (TransportWithMessageCredential). For more information about federation applications, see Federation and Issued Tokens.

NetPeerTcpBinding

The NetPeerTcpBinding class is a secure transport that is designed for efficient communication using the peer-to-peer networking feature. As indicated by the name of the class and binding, TCP is the protocol. When the security mode is set to Transport, the binding implements TLS over TCP. For more information about the peer-to-peer feature, see Peer-to-Peer Networking.

MsmqIntegrationBinding and NetMsmqBinding

For a complete discussion of transport security with Message Queuing (previously called MSMQ), see Securing Messages Using Transport Security.