This page is specific to

Microsoft Visual Studio 2010/.NET Framework 4

**Security Architecture**

This section describes various extensibility points that can be used to modify or extend the functionality of the Windows Communication Foundation (WCF) security component. To understand these extensibility points, it is necessary to understand the overall WCF security architecture. This topic describes the WCF security architecture in terms of its components and their relationships, and how the extensibility points described later in this section fit into the overall architecture model.

**Scope of the WCF Security Component**

WCF security spans multiple components in the WCF architecture. The main goal of security in WCF is to provide integrity, confidentiality, authentication, authorization, and auditing for the applications that are built on top with the WCF framework. WCF architecture splits these functions into the following pieces:

* *Transfer security* - Responsible for providing message confidentiality, data integrity, and authentication of communicating parties.
* *Authorization* - Responsible for providing a framework for making authorization decisions.
* *Auditing* - Responsible for logging security-related events to the audit log.

**Security Modes and Scope of this Document**

Transfer security can be performed using one of the following *security modes*:

* Transport. All three communication security functions are provided by the transport that is used to transmit messages between client and service.
* Message. The transfer security is provided solely at the SOAP message level, which means that the security is applied directly to the SOAP message on the XML level.
* Transport with Message Credential. The transfer security is performed on both transport and message layers. The transport layer provides communication confidentiality, data integrity, and service authentication. The message provides client authentication.

The rest of this document concentrates on Message security mode, although some of the information can be applied to the Transport with Message Credential mode as well. Specifically, the part of this document that applies to client authentication also applies to Transport With Message Credential mode because the Transport with Message Credential mode uses the message layer to perform client authentication in the same way that the Message mode does.

Discussions of Authorization and Auditing components apply to all three security modes in the same way. Therefore, all information related to these components described in this document applies to all supported security modes.

**Message Security Mode Concepts**

**WS-Security Model**

The foundation of the Message security mode is the *WS-Security* specification. The WS-Security specification defines a framework that allows security to be applied to SOAP messages. It specifies a message security model using security tokens combined with digital signatures and encryption to protect and authenticate SOAP messages. For the specification, see [Web Services Security (WS-Security)](http://go.microsoft.com/fwlink/?LinkId=102197).

**Terminology**

A *security token* asserts claims and can be used to assert the binding between authentication secrets or keys and security identities.

A *claim* is a declaration made by an entity about an entity (for example, a name, identity, group, key, group, or privilege). The entity that makes the claim is referred to as a *claim issuer*; the entity about which the claim is made is referred to as a *claim subject*.

A *claim issuer* can vouch for or endorse the claims in a security token by using its key to sign or encrypt the security token. This enables authentication of the claims in the security token.

*Message signatures* are used to verify message origin and integrity. Message signatures are also used by message producers to demonstrate knowledge of the key, typically from a third party, used to confirm the claims in a security token and thus to bind their identity (and any other claims represented by the security token) to the messages they create.

**Security Tokens**

WS-Security defines several types of security tokens and gives an extensible model that allows additional security token types to be defined independently. Every token type definition contains a XML serialization of the token. This allows adding the token representation directly to the message.

The following are some of the security token types defined in WS-Security:

* Username Token.
* X.509 Certificate Token.
* Kerberos Token.
* SAML Token.

There are four defined ways of using tokens, and the tokens attached to a given message fall into exactly one of these categories:

* SignedSupporting
* SignedEndorsing
* SignedEncrypted
* EncryptedEndorsing

In .NET Framework 3.0, a client message can contain only one token of any given type, but can contain tokens of different types.

In .NET Framework 3.5, client messages can contain multiple tokens of a given type, as well as tokens of different types.

This feature enables a number of scenarios, including the following:

* Incremental claim sending. All operations on a service might require a set of claims to be present, but some operations might require additional claims. Instead of using separate issued tokens for each operation, the client can obtain one issued token with the initial set of claims and use another issued token with the rest of the claims required for the operation being called.
* Multifactor authentication. When the client must collect issued tokens from multiple issuers or issued tokens with different sets of claims before being allowed to perform an operation. WCF considers the issued token to be a token type, so this scenario requires the ability to have two supporting issued tokens in the message.

Note that you cannot configure a service in this way: a service can contain only one supporting token.

For more information, see [How to: Use Multiple Security Tokens of the Same Type](http://msdn.microsoft.com/en-us/library/bb885138.aspx).

**WS-Security Implementation**

Because WS-Security lays a foundation for message security, the WCF implementation of WS-Security is a cornerstone of the whole Message security mode. To extend the Message security mode functionality, it is necessary to understand how the WS-Security implementation works.

The WS-Security implementation in WCF handles the following:

* Serialization of security tokens to and from SOAP messages.
* Authentication of security tokens.
* Application and verification of message signatures.
* Encryption and decryption of SOAP messages.

WCF extensibility points allow customization of the first two items. It is possible to change the serialization of existing security tokens or the way WCF security authenticates those tokens. It is also possible to introduce completely new security token types to the WCF security, including the serialization and authentication functionality.

The following topics in this section show how the WS-Security implementation extensibility points can be used to customize the security tokens functionality.

**Authorization**

Security tokens are deserialized from the incoming message and authenticated. The authentication process results in a set of authorization-policy objects. Each object represents a part of the security token's data. That data is used during the authorization stage.

The *authorization policy* creates a set of claims that a given security token represents. This claim set is then provided for the authorization decisions to [ServiceAuthorizationManager](http://msdn.microsoft.com/en-us/library/system.servicemodel.serviceauthorizationmanager.aspx) and to the logic inside the [AuthorizationContext](http://msdn.microsoft.com/en-us/library/system.servicemodel.servicesecuritycontext.authorizationcontext.aspx) property.

**Identity**

In addition to claims, WCF creates an implementation of the [IIdentity](http://msdn.microsoft.com/en-us/library/system.security.principal.iidentity.aspx) interface to represent the caller to the existing infrastructure (created by the .NET Framework security model). This **IIdentity** instance represents either the Windows identity of the caller if the security token is mapped to a Windows account, or a Primary Identity that contains the caller name. Those identities are also accessible using the [ServiceSecurityContext](http://msdn.microsoft.com/en-us/library/system.servicemodel.servicesecuritycontext.aspx). (For more information, see [How to: Examine the Security Context](http://msdn.microsoft.com/en-us/library/aa347790.aspx).) It is possible to customize the way identities are created in WCF by using one of the following methods:

* Implementing a custom extension of the [SecurityTokenAuthenticator](http://msdn.microsoft.com/en-us/library/system.identitymodel.selectors.securitytokenauthenticator.aspx) class.
* Integrating with ASP.NET by extending the [MembershipProvider](http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.aspx) class, or by creating a custom [IAuthorizationPolicy](http://msdn.microsoft.com/en-us/library/system.identitymodel.policy.iauthorizationpolicy.aspx) implementation and plugging it into the **ServiceAuthorizationManager**.
* Creating a custom **IAuthorizationPolicy**.

The custom membership provider works only if user name/password authentication is used to authenticate the caller. The **MembershipProvider** validates the user name/password pair. If the pair is valid, WCF creates a Primary Identity that represents the authenticated caller after **MembershipProvider** validation.

To ease the integration with the existing .NET Framework security infrastructure, WCF sets by default the [CurrentPrincipal](http://msdn.microsoft.com/en-us/library/system.threading.thread.currentprincipal.aspx) property to an [IPrincipal](http://msdn.microsoft.com/en-us/library/system.security.principal.iprincipal.aspx) instance that represents the caller. The **IPrincipal** instance is created based on the information contained in the generated **IIdentity**.

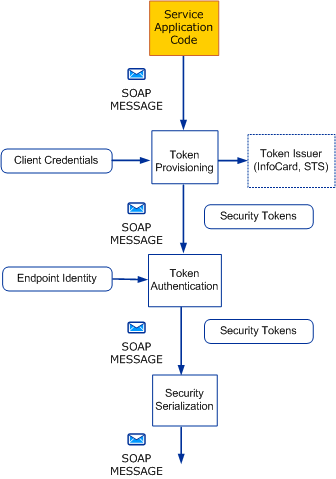
The **IIdentity** can be further augmented by integrating with the ASP.NET [RoleProvider](http://msdn.microsoft.com/en-us/library/system.web.security.roleprovider.aspx). The **RoleProvider** adds a set of roles that the caller belongs to. The authorization logic uses this information to make the access decision.

For more information about identity, see [Service Identity and Authentication](http://msdn.microsoft.com/en-us/library/ms733130.aspx).

**Sending Secured Messages**

The following illustration shows how a message is secured on the client when using the Message security mode. The illustration shows which components are involved and the relationships between them:

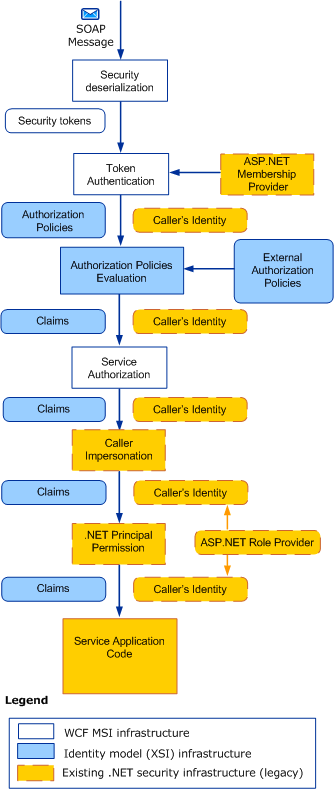
1. The application code runs and generates a message.
2. In the *Token Provisioning* stage, the client credentials (such as X.509 certificates) are attached. In a federated scenario, a token issuer is contacted to supply the credentials.
3. The credentials are used to create the security token.
4. In the *Token Authentication* stage, the tokens are verified.
5. Finally, the security tokens are serialized and sent.



**Receiving Secured Messages**

The following illustration shows the processes that occur when a secure message is extracted from the wire and verified on the receiving side:

1. The security tokens are deserialized and processed in the token authentication stage. If desired, an ASP.NET membership provider can be used to supply user names and passwords at this point.
2. After authentication, the authorization policies are extracted.
3. In the *Authorization Policies Evaluation* stage, the authorization policies are evaluated and claims can be added to an *Evaluation Context*. External authorization policies are also used at this point. This step, as well as the following step, is done by methods of the **ServiceAuthorizationManager**.
4. In the *Service Authorization* stage, the correct authorizations are given based on claims added by the authorization policies. This step is done by methods of the **ServiceAuthorizationManager**.
5. After authorization occurs, caller impersonation occurs if the caller has allowed it and the service method requires it, or the [ImpersonateCallerForAllOperations](http://msdn.microsoft.com/en-us/library/system.servicemodel.description.serviceauthorizationbehavior.impersonatecallerforalloperations.aspx) property is set on the service authorization behavior. For more information, see [Delegation and Impersonation with WCF](http://msdn.microsoft.com/en-us/library/ms730088.aspx).
6. WCF generates a [PrincipalPermission](http://msdn.microsoft.com/en-us/library/system.security.permissions.principalpermission.aspx) using the credentials at this point. If required, an ASP.NET role provider can be used at this point.
7. The application code runs.



**Overview of Security Extensibility Points**

The following illustration shows the extensibility points provided by the WCF security components. The figure is divided into four different categories depending on when the extensibility point is reached during the message processing. Those categories map to the message security processing stages as described in the previous two sections. The figure also shows that existing infrastructure technologies can be integrated with WCF security.

