Designing Distributed Applications with Visual Studio .NET

**ASP.NET Impersonation**

Another important security feature is the ability to control the identity under which code is executed. Impersonation is when ASP.NET executes code in the context of an authenticated and authorized client. By default, ASP.NET does not use impersonation and instead executes all code using the same user account as the ASP.NET process, which is typically the ASPNET account. This is contrary to the default behavior of ASP, which uses impersonation by default. In Internet Information Services (IIS) 6, the default identity is the NetworkService account.

**Note** Impersonation can significantly affect performance and scaling. It is generally more expensive to impersonate a client on a call than to make the call directly.

Using impersonation, ASP.NET applications can optionally execute the processing thread using the identity of the client on whose behalf they are operating. You usually use impersonation for resource access control. Delegation is a more powerful form of impersonation and makes it possible for the server process to access remote resources while acting as the client. For more information, see [ASP.NET Delegation](http://msdn.microsoft.com/en-us/library/aa291350(VS.71).aspx).

**Note** Impersonation is local to a particular thread. When code changes threads, such as when using thread pooling, the new thread executes using the process identity by default. When impersonation is required on the new thread, your application should save the security token ([WindowsIdentity.Token Property](http://msdn.microsoft.com/en-us/library/system.security.principal.windowsidentity.token(VS.71).aspx)) from the original thread as part of the state for the completion thread.

If you enable impersonation, ASP.NET can either impersonate the authenticated identity received from IIS or one specified in the application's Web.config file. You have the following three options when configuring impersonation:

* **Impersonation is disabled**. This is the default setting. For backward compatibility with ASP, you must enable impersonation and change the ASP.NET process identity to use the Local System account. In this instance, the ASP.NET thread runs using the process token of the application worker process regardless of which combination of IIS and ASP.NET authentication is used. By default, the process identity of the application worker process is the ASPNET account. For more information, see [ASP.NET Process Identity](http://msdn.microsoft.com/en-us/library/aa291339(VS.71).aspx).

<identity impersonate="false" />

* **Impersonation enabled**. In this instance, ASP.NET impersonates the token passed to it by IIS, which is either an authenticated user or the anonymous Internet user account (IUSR\_*machinename*).

<identity impersonate="true" />

* **Impersonation enabled for a specific identity**. In this instance, ASP.NET impersonates the token generated using an identity specified in the Web.config file.

<identity impersonate="true"

userName="domain\user"

password="password" />

If the application resides on a UNC share, ASP.NET always impersonates the IIS UNC token to access that share unless a configured account is used. If you provide an explicitly configured account, ASP.NET uses that account in preference to the IIS UNC token.

You should exercise care when using impersonation because it makes it possible for an application to potentially process code using permissions not anticipated by the application designer. For example, if your application impersonates an authenticated intranet user, that application possesses administrative privileges when impersonating a user with those privileges. Likewise, if the impersonated user possesses more restrictive permissions than anticipated, the user may not be able to use the application.