.NET Framework 4 - ASP.NET

**Accessing SQL Server from a Web Application**

When a Web application involves database access, it must provide credentials to SQL Server (that is, it must log in to SQL Server) just as any other user or process would. In a Web application, this can introduce complications. For example, if the Web application runs anonymously, there might not be credentials to pass to SQL Server.

There are a number of ways to design SQL Server access for your Web application. The strategy you choose depends on how your computers are configured and whether you are on an intranet. The simplest options are:

* Use Windows integrated security. This option passes the user's credentials to SQL Server. Because of delegation issues, this frequently only works by default if SQL Server is on the same computer as IIS.
* Map the identity of your ASP.NET application to a Windows domain user and then log into the database as that user. This works well for anonymous access if SQL Server and the Web server are on separate computers.
* Access the SQL Server as the local identity of your ASP.NET application (for example, the local ASPNET account on a Windows 2000 server or the local NETWORK SERVICE account on a Windows Server 2003). This option works well for anonymous access.
* Pass an explicit user name and password in a connection string. This option can be less secure than other options so you should always use protected configuration to secure the connection strings. You can pass a predetermined user name and password.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifIn This Section

|  |  |
| --- | --- |
| **Term** | **Definition** |
| [How to: Access SQL Server Using Windows Integrated Security](http://msdn.microsoft.com/en-us/library/bsz5788z.aspx) | Provides an example of how to use Windows integrated security for database access. |
| [How to: Access SQL Server Using a Mapped Windows Domain User](http://msdn.microsoft.com/en-us/library/2xzyzb0f.aspx) | Provides an example of how to use a mapped Windows domain user for database access. |
| [How to: Access SQL Server as a Local User](http://msdn.microsoft.com/en-us/library/e2t54ss5.aspx) | Provides an example of how to use a local user account for database access. |
| [How to: Access SQL Server Using Predetermined Credentials](http://msdn.microsoft.com/en-us/library/ds20z471.aspx) | Provides an example of how to use predetermined login information for database access. |

.NET Framework 4 - ASP.NET

**How to: Access SQL Server Using Windows Integrated Security**

If your application runs on a Windows-based intranet, you might be able to use Windows integrated authentication for database access. Integrated security uses the current Windows identity established on the operating system thread to access the SQL Server database. You can then map the Windows identity to a SQL Server database and permissions.

To connect to SQL Server using Windows integrated authentication, you must identify the Windows identity under which your ASP.NET application is running. You must also be sure that the identity has been granted access to the SQL Server database. This topic includes a code example that displays the current Windows identity of the ASP.NET application.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifConnecting to SQL Server

If SQL Server is on a different computer than the Web server, the Windows identity must be able to flow across the network to the remote instance of SQL Server. (Windows networks that have been configured appropriately with Kerberos authentication are able to do this.) However, depending on the settings in the [identity](http://msdn.microsoft.com/en-us/library/72wdk8cc.aspx) configuration element, the Windows identity established on the operating system thread for ASP.NET applications may not be able to flow properly to the remote SQL Server.

You can supply a specific user name and password for the Web site's worker process identity as shown in [How to: Access SQL Server Using a Mapped Windows Domain User](http://msdn.microsoft.com/en-us/library/2xzyzb0f.aspx), or you can impersonate the authenticated identity supplied by Internet Information Services (IIS). To impersonate the Windows identity supplied by IIS, set the **impersonate** attribute of the [identity](http://msdn.microsoft.com/en-us/library/72wdk8cc.aspx) configuration element to **true** as shown in the following example:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl16_ctl00_ctl03_code');" \o "Copy Code)

<system.web>

<identity impersonate="true" />

</system.web>

In IIS, only Basic Authentication logs users on with a security token that flows across the network to a remote SQL server. By default, other IIS security modes used in conjunction with the [identity](http://msdn.microsoft.com/en-us/library/72wdk8cc.aspx) configuration element settings will not result in a token that can authenticate to a remote SQL Server.

If the Web site is configured to support only anonymous access in IIS, then the security token passed from IIS will be that of the Windows user account for anonymous access as configured in IIS. The anonymous user account can be used to authenticate against a remote SQL Server. However, the default anonymous user account is a local machine account and thus will not exist as an account on the remote SQL Server. You can change the IIS anonymous account to use a domain account, or you can mirror the local machine account on the remote SQL Server by creating a local account on the remote SQL Server with the same user name and password. Additionally the **LogonMethod** metabase property for IIS6 must be set to an option that allows credentials to flow across the network. For example, the metabase setting **MD\_LOGON\_NETWORK\_CLEARTEXT** allows logon credentials to flow across the network.

If you are unsure of the Windows identity for your application and whether that identity is logged on with a token that can flow across the network, you can run the following ASP.NET page as part of your application to display the name of the Windows identity and a value indicating whether the identity can flow across the network. Note that the following sample does not take into account whether or not Kerberos delegation has been successfully set up for your domain.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl16_ctl00_ctl05_code');" \o "Copy Code)

<%@ Page Language="VB" %>

<%@ Import namespace="System.Security.Principal" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html >

<head runat="server">

<title>ASP.NET Example</title>

<script runat="server">

Public Function WillFlowAcrossNetwork(w As WindowsIdentity) As Boolean

For Each s As SecurityIdentifier In w.Groups

If s.IsWellKnown(WellKnownSidType.InteractiveSid) Then Return True

If s.IsWellKnown(WellKnownSidType.BatchSid) Then Return True

If s.IsWellKnown(WellKnownSidType.ServiceSid) Then Return True

Next

Return False

End Function

</script>

</head>

<body>

<%

Dim current As WindowsIdentity = WindowsIdentity.GetCurrent()

Response.Write(current.Name & ", " & WillFlowAcrossNetwork(current) & "<br />")

%>

</body>

</html>

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl16_ctl00_ctl06_code');" \o "Copy Code)

<%@ Page Language="C#" %>

<%@ Import namespace="System.Security.Principal" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html >

<head runat="server">

<title>ASP.NET Example</title>

<script runat="server">

public bool WillFlowAcrossNetwork(WindowsIdentity w)

{

foreach (SecurityIdentifier s in w.Groups)

{

if (s.IsWellKnown(WellKnownSidType.InteractiveSid)) { return true; }

if (s.IsWellKnown(WellKnownSidType.BatchSid)) { return true; }

if (s.IsWellKnown(WellKnownSidType.ServiceSid)) { return true; }

}

return false;

}

</script>

</head>

<body>

<%

WindowsIdentity current = WindowsIdentity.GetCurrent();

Response.Write(current.Name + ", " + WillFlowAcrossNetwork(current) + "<br />");

%>

</body>

</html>

The following procedure shows how to access a SQL Server database using Windows integrated authentication in an intranet scenario, where each user has been granted access to the SQL Server individually.

To begin, you need to configure your application in IIS to turn off anonymous access and turn on Windows authentication.

**To configure IIS for Windows integrated authentication**

1. In Windows, open the **Internet Information Services** administration tool.
   * In the Microsoft Windows 2000 Server or Windows Server 2003 operating systems: In the Windows **Start** menu, point to **Programs**, then **Administrative Tools**, and then **Internet Services Manager**.
   * In the Microsoft Windows XP Professional operating system: open **Administrative Tools** in the Control Panel.
2. Open the node for your server, and then open nodes until you find the node for your application, which is typically located under **Default Web Site**.
3. Right-click your application and then click **Properties**.
4. On the **Directory Security** tab, click **Edit**.
5. In the **Authentication Methods** dialog box, clear the **Anonymous Access** check box, and then do one of the following:
   * If SQL Server is on the same computer as IIS, select the **Integrated Windows authentication** check box.
   * If SQL Server is a remote server, select the **Basic Authentication** check box and clear the **Integrated Windows authentication** check box.
6. Click all the dialog boxes.

In the application configuration file (Web.config), specify that the application will impersonate the user's credentials supplied by IIS.

**To configure Web.config to impersonate the identity supplied by IIS**

* Open the Web.config file for your application and add the following to the **system.web** element:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl16_ctl00_ctl07_code');" \o "Copy Code)

<identity impersonate="true"/>

|  |
| --- |
| **Description: NoteNote** |
| Elements in Web.config are case sensitive. |

When you create a connection string to access SQL Server, you must include attributes that tell SQL Server that you are using integrated security.

**To configure connection strings for Windows integrated security**

* In any connection string for SQL Server, include the attribute Trusted\_Connection=Yes and remove the username and password attributes.

The following shows a typical connection string configured for Windows integrated security:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl16_ctl00_ctl09_code');" \o "Copy Code)

"workstation id=WebServer1;packet size=4096;

Trusted\_Connection=Yes;data source=SQLServer01";

persist security info=False;initial catalog=northwind"

Set up SQL Server to recognize the users who will be accessing it.

**To configure SQL Server for Windows integrated security**

1. From the Windows **Start** menu, select **Microsoft SQL Server**, and then select **Enterprise Manager**.
2. Open the node for the server and expand the node for the database you want to give users permissions for.
3. Right-click the **Users** node and select **New Database User**.
4. In the **Database User Properties** dialog box, enter *domain*\*username* in the **Login name** box, and then click **OK**. Additionally, configure the SQL Server to allow all domain users to access the database.

.NET Framework 4 - ASP.NET

**How to: Access SQL Server Using a Mapped Windows Domain User**

By default, in Microsoft Windows 2000 and Microsoft Windows XP, ASP.NET applications run in the context of the local user account named ASPNET and in Windows Server 2003 in the context of the local user account named NETWORK SERVICE. These user accounts have limited access rights. However, the ASPNET account is local to the Web server. Because the ASPNET account is local to the Web server, it is not recognized as a user on remote computers. To work around this limitation, you can have your application run in the context of a Windows domain user who is recognized on both the Web server and the computer that is running Microsoft SQL Server.

Mapping your application process to a Windows domain user account requires that you configure the following:

* The Web server.

You must make sure that the Windows domain user account that you specify has sufficient user rights (but no more) to run a Web application.

* Your application.

You must configure the Web.config file for ASP.NET to recognize the domain user account name.

|  |
| --- |
| **Description: NoteNote** |
| For information about the Machine.config and Web.config files, see [ASP.NET Configuration Overview](http://msdn.microsoft.com/en-us/library/ms178683.aspx). |

* A connection string.

When you create connection strings for connection objects in your application, you have to specify that the connection strings will use Windows integrated security.

* SQL Server.

You must add the specified domain user account as a SQL Server login user.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifConfiguring a User Account on the Web Server

**To set user rights for the Windows domain user account**

1. On the Web server, use Windows administrative tools to make sure that the mapped Windows domain user account has the required user rights.

For detailed information, see [ASP.NET Required Access Control Lists (ACLs)](http://msdn.microsoft.com/en-us/library/kwzs111e.aspx).

1. Run aspnet\_regiis.exe with the **-ga** switch to grant the common user rights that are required by the identity that you will use for application impersonation.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifMapping to the Windows User Account and Enabling Impersonation

After establishing the correct user rights for the domain user account, configure the application identity impersonation.

|  |
| --- |
| **Description: Security noteSecurity Note** |
| When you place user credentials in your Web.config file, there are potential security threats. Users with access rights to the directory containing the Web.config file can read the file, and thus see the credentials. For details on how to protect against this threat, see [Encrypting Configuration Information Using Protected Configuration](http://msdn.microsoft.com/en-us/library/53tyfkaw.aspx). |

**To configure the Web application for impersonation**

* Open the Web.config file for your application, and then add the following identity impersonation code:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl16_ctl00_ctl02_code');" \o "Copy Code)

<identity impersonate="true" userName="domain\username" password="\*\*\*\*\*\*\*\*"/>

Substitute the correct password for the value listed in the previous example.

|  |
| --- |
| **Description: NoteNote** |
| Elements in the Web.config file are case sensitive. |

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifUsing Windows Security in the Connection String

Finally, when you create connection strings for database access, configure the connection strings to use Windows integrated security.

**To use Windows integrated security in a connection string**

* When you create a connection string for your application, do not include a user name and password. Instead, for the connection string, set the **Integrated Security** attribute to **SSPI**.

The following example shows a connection string that includes the appropriate attributes:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl17_ctl00_ctl00_code');" \o "Copy Code)

data source=myserver;initial catalog=northwind;Integrated Security=SSPI

**To configure SQL Server for integrated security**

1. In Windows, click **Start**, point to **Microsoft SQL Server**, and then click **Enterprise Manager**.
2. Open the node for the server, and then expand the node for the database to which you want to grant user rights.
3. Right-click **Users**, and then click **New Database User**.
4. In the **Database User Properties** dialog box, in the **Login name** box, enter *domain\username*, and then click **OK**.

.NET Framework 4 - ASP.NET

**How to: Access SQL Server as a Local User**

By default, when an ASP.NET application runs, the application runs in the context of a special local user account named ASPNET (in Microsoft Windows 2000 and Microsoft Windows XP) or NETWORK SERVICE (in Microsoft Windows Server 2003). This user account has sufficient user rights on the Web server to allow the application to run.

If Microsoft SQL Server and the Web server are on the same computer, you can define the ASPNET or NETWORK SERVICE user account as a local user account on the computer running SQL Server.

The advantage of this method is that it is secure, because the Web application runs within a limited security context. Using a single user name with SQL Server is also efficient because it enables SQL Server to take advantage of connection pooling, which further enhances the scalability of the application.

**To grant SQL Server user rights to the ASPNET or NETWORK SERVICE user account**

1. In Windows, click **Start**, point to **Programs**, point to **Microsoft SQL Server**, and then click **Enterprise Manager**.
2. Expand the node for the server, and then expand the node for the database for which you want to grant user rights.
3. Right-click **Users**, and then click **New Database User**.
4. In the **Database User Properties** dialog box, in the **Login name** box, enter *computername***\ASPNET** or **NT AUTHORITY\NETWORK SERVICE**, and then click **OK**.

.NET Framework 4 - ASP.NET

**How to: Access SQL Server Using Predetermined Credentials**

A reliable way to connect to SQL Server is to pass a user name and password in the connection string. You can use a predetermined user name and password. The recommended method is to store the predetermined user name and password on the server as part of the [connectionStrings](http://msdn.microsoft.com/en-us/library/bf7sd233.aspx) configuration section and then use protected configuration to encrypt the connection string contents. For details, see to [Overview of Protected Configuration](http://msdn.microsoft.com/en-us/library/hh8x3tas.aspx). It is also recommended that you restrict access to the Web.config file using NTFS file system permissions.

|  |
| --- |
| **Description: Security noteSecurity Note** |
| Never hard-code credentials as strings into programs in your application. Anyone who can get access to the code file, even the compiled code, will be able to get at the credentials. |
| **Description: Security noteSecurity Note** |
| Always give a predetermined user name the minimal access privileges to a resource. Never use "sa" or any other administrative-level user name. Always use strong passwords. |

**To store credentials in the Web.config file**

1. In the Web.config file, create a new **add** key in the [connectionStrings](http://msdn.microsoft.com/en-us/library/bf7sd233.aspx) element. The **connectionStrings** element must appear as a child of the **configuration** element. For details, see [Configuration Sections Schema](http://msdn.microsoft.com/en-us/library/0hyxd0xc.aspx).

The following example illustrates an **add** key that contains a user name and password:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl07_code');" \o "Copy Code)

<configuration>

<connectionStrings>

<add name="NorthwindConnection"

connectionString="Data Source=localhost;

Initial Catalog=Northwind;

User Id=ApplicationUserID;

Password=#P%19!ef2" />

</connectionStrings>

</configuration>

1. Encrypt the connection string value using protected configuration as shown in [Walkthrough: Encrypting Configuration Information Using Protected Configuration](http://msdn.microsoft.com/en-us/library/dtkwfdky.aspx).
2. In your application, read the credentials from the [ConnectionStrings](http://msdn.microsoft.com/en-us/library/system.configuration.configurationmanager.connectionstrings.aspx) property of the [ConfigurationManager](http://msdn.microsoft.com/en-us/library/system.configuration.configurationmanager.aspx) class.

The following example shows how you can read credentials at run time and concatenate them into a connection string:

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl11_code');" \o "Copy Code)

Dim settings As ConnectionStringSettings

settings = System.Configuration.ConfigurationManager.ConnectionStrings("NorthwindConnection")

Dim connectionString As String = settings.ConnectionString

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl12_code');" \o "Copy Code)

ConnectionStringSettings settings;

settings = System.Configuration.ConfigurationManager.ConnectionStrings["NorthwindConnection"];

string connectionString = settings.ConnectionString;