**Microsoft Office SharePoint Server (MOSS) for Developers**

**(Part 1)**

**Topic 10: Users and Permissions**

[](http://msdn.microsoft.com/en-us/rampup/bb352986.aspx)

Ascend to new heights in your career

[**http://msdn.microsoft.com/rampup**](http://msdn.microsoft.com/rampup)

**Introduction**

In Windows SharePoint Services 3.0, access to Web sites, lists, folders, and list items is controlled through a role-based membership system by which users are assigned to roles that authorize their access to Windows SharePoint Services objects.

To give a user access to an object, you can do so either by adding the user to a group that already has permissions on the object, or by creating a role assignment object, setting the user for the role assignment, optionally binding the role assignment to the appropriate role definition with base permissions, and then adding the assignment to the collection of role assignments for the list item, folder, list, or Web site. If you do not bind the role assignment to a role definition when assigning a user to a role, the user has no permission.

Windows SharePoint Services provides the following ways to control access to its objects:

* Objects can either use the same permissions as the parent Web site, list, or folder, inheriting both the roles and users available on the parent object, or they can use unique permissions.
* Sites, lists, folders, and items each provide role assignment collections, allowing fine management of user access to objects.
* In Windows SharePoint Services 3.0, there is only one kind of group, previously called a cross-site group. Groups consist of users and may or may not be assigned to roles. Windows SharePoint Services includes the following three groups by default:
  + **owners** (administrator)
  + **members** (contributor)
  + **visitors** (reader)

When you create a Web site with unique permissions through the user interface, you are directed to a page where you can assign users to these groups as part of provisioning the site.

* Anonymous access allows users to contribute anonymously to lists and surveys, or to view pages anonymously. You can also grant access to "all authenticated users" to allow all members of your domain to access a Web site without having to enable anonymous access.
* Site creation rights (**CreateSSCSite** and **ManageSubwebs**) control whether users can create top-level Web sites, subsites, or workspaces.

Users become members of a SharePoint object either indirectly through a group that has a role assignment, or directly through a role assignment. In addition, users can be members of a Microsoft Windows NT Domain Group that is added to a group or to a role. A role definition associates a user or group with a single right or set of rights corresponding to values of the [Microsoft.SharePoint.SPBasePermissions](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.spbasepermissions.aspx) enumeration. Each user or group has a unique member [ID](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.spmember.id.aspx).

You can use the object model to create or modify role assignments and definitions differently than you can via the functionality of the files addrole.aspx and editrole.aspx. Unlike these pages presented in the UI, the object model does not enforce rights dependency, so you can create a role definition with an arbitrary combination of rights. However, plan carefully when using the object model to customize role definitions and permissions, because a poorly conceived role definition and inappropriately assigned rights can lead to an unpleasant user experience.

For more information about Windows SharePoint Services rights, see **SPBasePermissions**.

You will need to install the [**Windows SharePoint Services 3.0 SDK**](http://msdn.microsoft.com/en-us/library/ms441339.aspx)

**Security Policy**

A security policy provides a means to enforce uniform security throughout all site collections within a Web application (virtual server). Through policy you can assign a role, or collection of rights, to individual Windows SharePoint Services users, and to domain groups using Windows authentication or pluggable authentication systems, but not to SharePoint groups. Each policy entry specifies rights for a user or group in the Web application.

Policy is set at the logical Web application or zone level. A user can have, for example, different policies on http://Server and http://Server.extranet.microsoft.com, even if the two Web applications have the same content.

Rights can be granted or denied via policy. Granting a right gives that right to the user or group on all secured objects within the Web application regardless of local permissions on the object. Denying a right trumps granting the right, actively blocking that right for the user or group on all secured objects within the Web application. Denying all for a user prevents that user from accessing any content, even if the user has explicit permissions on specific content: policy overrides site-level permissions.

In policy roles, the users and groups are identified by both their security identifier (SID) and their login or user name. Applying a policy role is similar to managing permissions for a Web site, list, folder, or document: you add users or groups and assign them to one or more role definitions. However, unlike permissions on Windows SharePoint Services objects, policy roles are not shared with other Web applications or with the content role definitions that are used to manage the permissions on objects. Each Web application has its own policy roles. Another difference between policy roles and managing permissions is that central administrators can deny a right to a user throughout a Web application.

**Authorization and Authentication**

Windows SharePoint Services 3.0 supports security for user access at the Web site, list, list or library folder, and item levels. Security management is role-based at all levels, providing coherent security management across the Windows SharePoint Services platform with a consistent role-based user interface and object model for assigning permissions on objects. As a result, list-level, folder-level, or item-level security implements the same user model as Web site-level security, making it easier to manage user and group rights throughout a Web site. Windows SharePoint Services also supports unique permissions on the folders and items contained within lists and document libraries.

Authorization refers to the process by which Windows SharePoint Services provides security for Web sites, lists, folders, or items by determining which users can perform specific actions on a given object. The authorization process assumes that the user has already been authenticated, which refers to the process by which Windows SharePoint Services identifies the current user. Windows SharePoint Services does not implement its own system for authentication or identity management, but instead relies solely on external systems, whether Microsoft Windows or non-Windows authentication.

Windows SharePoint Services supports the following types of authentication:

* **Windows:** All Microsoft Internet Information Services (IIS) and Windows authentication integration options, including Basic, Digest, Certificates, Windows NT LAN Manager (NTLM), and Kerberos. Windows authentication allows IIS to perform the authentication for Windows SharePoint Services.

|  |
| --- |
| **NoteImportant:** |
| If you install "Infrastructure Update for Windows SharePoint Services 3.0 (KB951695)," custom solutions may fail if they call the SharePoint object model while impersonation is suspended. If you use Windows authentication and your code calls the SharePoint object model from an IIS worker process, the request must impersonate the calling user’s identity. Windows SharePoint Services configures ASP.NET to impersonate the calling user automatically, but your code may work unexpectedly, or fail, if you suspend impersonation--for example, by calling the **RevertToSelf** function of the Windows API, or by calling the **System.Security.Principal.WindowsIdentity.Impersonate** method and passing **IntPtr.Zero** as the value of the user token parameter. Even if your code does not explicitly revert to self, it might be called by ASP.NET after it reverts to self, such as happens when implementing a virtual path provider; if your code does not impersonate the calling user, it might not function properly. |

* **ASP.NET Forms:** A non-Windows identity management system that uses the pluggable Microsoft ASP.NET forms-based authentication system. This mode allows Windows SharePoint Services to work with a variety of identity management systems, including externally defined groups or roles such as Lightweight Directory Access Protocol (LDAP) and light-weight database identity management systems. Forms authentication allows ASP.NET to perform the authentication for Windows SharePoint Services, often involving redirect to a log-on page.
* **Delegated:** A system for delegating end-user credentials from a trusted system to Windows SharePoint Services. This allows trusted services to pass user identities to Windows SharePoint Services for authorization, conveying who the current user is without requiring that Windows SharePoint Services have that user's credentials.

|  |
| --- |
| **NoteNote:** |
| Windows SharePoint Services does not support working with a case-sensitive membership provider and uses case-insensitive SQL storage for all users in the database regardless of the membership provider. |

# **Forms-Based Authentication**

Forms-based authentication provides custom identity management in Windows SharePoint Services by implementing a membership provider, which defines interfaces for identifying and authenticating individual users, and a role manager, which defines interfaces for grouping individual users into logical groups or roles. Given a user name, the role provider system returns a list of roles to which the user belongs. The membership provider creates a user token from the logon name and password, while the role manager creates a set of group membership tokens for the current user.

The role manager is optional, so if a custom authentication system does not support groups (Windows Live ID, for example), then a role manager is not necessary. Windows SharePoint Services supports one membership provider and one role manager per URL zone ([SPUrlZone](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.administration.spurlzone.aspx)). The ASP.NET forms roles have no inherent rights associated with them. Instead, Windows SharePoint Services assigns rights to the forms roles through its policies and authorization.

The log-on form (login.aspx) is provided by Windows SharePoint Services and resides in the \_layouts virtual directory (in *Local\_Drive*:\\Program Files\Common Files\Microsoft Shared\web server extensions\12\TEMPLATE\LAYOUTS). This file implements the ASP.NET Web form controls used to collect user credentials and cache credentials in a cookie.

The central administrator for a Web site must register the membership provider and role manager by modifying the web.config file for the run-time virtual server. If no providers are registered, ASP.NET forms-based authentication does not work. The central administrator must similarly update the sptimer.exe.config file.

Windows SharePoint Services consumes the standard ASP.NET 2.0 role provider interface to gather group information about the current user. For authentication purposes, roles and groups are the same thing: a way of grouping users into logical sets for authorization. Each ASP.NET role is treated as a domain group by Windows SharePoint Services.

For information about the pluggable authentication framework provided by ASP.NET, see [New Security Features in ASP.NET 2.0](http://go.microsoft.com/fwlink/?LinkID=64972&clcid=0x409).

# **Delegated Authentication**

Windows SharePoint Services supports delegating end-user credentials from an external system to a SharePoint deployment, allowing trusted services to pass user identities for authorization when the end user does not connect directly to Windows SharePoint Services but connects through another system. Windows SharePoint Services thus allows trusted code to specify who the current user is without Windows SharePoint Services having that user’s credentials.

Through delegated authentication one server delegates an end-user's authentication to another server. The client computer makes a request to the first server, which in turn makes one or more requests to the second server on the part of the end user. To succeed, the first server must delegate the end-users authentication to the second server, which requires one of two trust relationships:

* The client trusts the first server with its password, for example, to use basic authentication, or a server-side credential store such as the pluggable single sign-on mechanism of Microsoft Office SharePoint Server 2007 or Active Directory Federation Services (ADFS).
* The second server trusts the first server to pass on valid credentials for valid purposes, for example, to establish a Kerberos-constrained delegation trust relationship between the servers.

# **Additional resources and references**

* [Working with users, groups and authorization](http://msdn.microsoft.com/en-us/library/ms414400.aspx)
* [Introduction to SharePoint Products and Technologies for the Professional .NET Developer](http://msdn.microsoft.com/en-us/library/cc537498.aspx)
* [Windows SharePoint Services Developer Center](http://msdn2.microsoft.com/en-us/SharePoint)
* [Windows SharePoint Services TechCenter](http://technet.microsoft.com/en-us/office/sharepointserver/default.aspx)
* [SharePoint Server Developer Center](http://msdn2.microsoft.com/en-us/office/aa905503.aspx)
* [SharePoint Server TechCenter](http://technet.microsoft.com/en-us/windowsserver/sharepoint/default.aspx)