Authentication, authorization, and security in SharePoint 2013

In this article
What's new in SharePoint 2013 for authentication, authorization, and security
Authentication and authorization
Claims-based identity and authentication
Forms-based authentication
Additional resources

What's new in SharePoint 2013 for authentication, authorization, and security

The following are some of the enhancements added to SharePoint 2013:

- User sign-in
  - SharePoint 2013 continues to offer support for both claims and classic authentication modes. Claims authentication is the default authentication option in SharePoint 2013. Classic-mode authentication is deprecated and can be managed only by using Windows PowerShell. A lot of features in SharePoint 2013 require claims-mode.
  - The MigrateUsers method from SharePoint 2010 is now deprecated, it's no longer the correct way to migrate accounts. To migrate
accounts, use the new Windows PowerShell cmdlet called Convert-SPWebApplication. For more information see Migrate from classic-mode to claims-based authentication in SharePoint 2013.

- Requirement to register claims providers is eliminated. However, you do have to pre-configure claims type. You can choose the characters for the claim type and there is no enforcement on the ordering of claim types.
- Significantly more logging is provided to help troubleshoot authentication issues.

- Services and app authentication
  - In SharePoint 2013, you now have the ability to create apps for SharePoint. A app for SharePoint has its own identity and is associated with a security principal, called an app principal. Like users and groups, an app principal has certain permissions and rights. For more information, see Build apps for SharePoint.
  - In SharePoint 2013, the server-to-server security token service (STS) provides access tokens for server-to-server authentication. The server-to-server STS enables temporary access tokens to access other application services, such as Exchange Server 2013 and Microsoft Lync 2013, and apps for SharePoint 2013.

**Authentication and authorization**
SharePoint 2013 supports security for user access at the website, list, list or library folder, and item levels. Security management is role-based at all levels, providing coherent security management across the SharePoint 2013 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 website-level security, making it easier to manage user rights and group rights throughout a website. SharePoint 2013 also supports unique permissions on the folders and items contained within lists and document libraries.

**Note**
For information about authorization related to apps for SharePoint, see [Authorization and authentication for apps in SharePoint 2013](#).

Authorization refers to the process by which SharePoint 2013 provides security for websites, 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 SharePoint 2013 identifies the current user. SharePoint 2013 does not implement its own system for authentication or identity management, but instead relies on external systems, whether Windows authentication or non-Windows authentication.

SharePoint 2013 supports the following types of authentication:

- **Windows**: All Internet Information Services (IIS) and Windows authentication integration options, including Basic, Digest, Certificates, Windows NT LAN Manager (NTLM), and Kerberos are supported. Windows authentication allows IIS to perform the authentication for SharePoint 2013.

  For information about signing in to SharePoint by using Windows claims mode, see [Incoming claims: Signing into SharePoint 2013](#).

- **ASP.NET Forms**: A non-Windows identity management system that uses the pluggable ASP.NET forms-based authentication system is supported. This mode enables SharePoint 2013 to work with a variety of identity management systems, including externally defined groups or roles such as Lightweight...
Directory Access Protocol (LDAP) and lightweight database identity management systems. Forms authentication allows ASP.NET to perform the authentication for SharePoint 2013, often involving a redirect to a log-on page. In SharePoint 2013, ASP.NET forms are supported only under claims authentication. A forms provider must be registered within a web application that is configured for claims.

For information about signing in to SharePoint by using ASP.NET membership and role passive sign-in, see *Incoming claims: Signing into SharePoint 2013*.

**Note**

SharePoint 2013 does not support working with a case-sensitive membership provider. It uses case-insensitive SQL storage for all users in the database, regardless of the membership provider.

### Claims-based identity and authentication

Claims-based identity is an identity model in SharePoint 2013 that includes features such as authentication across users of Windows-based systems and systems that are not Windows-based, multiple authentication types, stronger real-time authentication, a wider set of principal types, and delegation of user identity between applications.

When a user signs in to SharePoint 2013, the user's token is validated and then used to sign in to SharePoint. The user's token is a security token issued by a claims provider. The following are supported sign-in or access modes:

- Windows claims–mode sign-in (default)
- SAML passive sign-in mode
- ASP.NET membership and role passive sign-in
- Windows classic–mode sign-in (deprecated in this release)

**Note**

For more information about signing into SharePoint and the different sign-in modes, see *Incoming claims: Signing into SharePoint 2013*.
When you build claims-aware applications, the user presents an identity to your application as a set of claims. One claim could be the user's name, another might be an email address. The idea here is that an external identity system is configured to give your application all the information that it needs about the user with each request, along with cryptographic assurance that the identity data received by your application comes from a trusted source.

Under this model, single sign-on is much easier to achieve, and your application is no longer responsible for the following:

- Authenticating users
- Storing user accounts and passwords
- Calling to enterprise directories to look up user identity details
- Integrating with identity systems from other platforms or companies

Under this model, your application makes identity-related decisions based on claims supplied by the user. This could be anything from simple application personalization with the user's first name, to authorizing the user to access higher-value features and resources in your application.

**Note**
For more information about claims-based identity and claims providers, see [Claims-based identity and concepts in SharePoint 2013](#) and [Claims provider in SharePoint 2013](#).

### Forms-based authentication

Forms-based authentication provides custom identity management in SharePoint 2013 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. In SharePoint 2013, a membership provider must implement the required System.Web.Security.Membership.ValidateUser method. Given a user name, the role provider system returns a list of roles to which the user belongs.

The membership provider is responsible for validating the credential information by using the System.Web.Security.Membership.ValidateUser method (required now in SharePoint 2013). But, the actual user token is created by the security token service (STS). The STS creates the user token from the user name validated by the membership provider and from the set of group memberships associated with the user name that are provided by the membership provider.

**Note**
For more information about STS, see [Claims-based identity and concepts in SharePoint 2013](#).
The role manager is optional. If a custom authentication system does not support groups, a role manager is not necessary. SharePoint 2013 supports one membership provider and one role manager per URL zone (SPUrlZone). The ASP.NET forms roles have no inherent rights associated with them. Instead, SharePoint 2013 assigns rights to the forms roles through its policies and authorization. In SharePoint 2013, the forms-based authentication is integrated with the claims-based identity model. If you need additional augmentation to bypass the limit of having one role provider per URL zone, you can rely on claims providers.

Note
For more information about claims-based identity and claims providers, see Claims-based identity and concepts in SharePoint 2013 and Claims provider in SharePoint 2013.

In ASP.NET membership and role passive sign-in, the sign-in happens by redirecting the client to a web page where the ASP.NET log-in controls are hosted. After the identity object that represents a user identity is created, SharePoint 2013 converts it to a ClaimsIdentity object (which represents a claims-based representation of a user).

Note
For more information about signing into SharePoint, see Incoming claims: Signing into SharePoint 2013.

SharePoint 2013 consumes the standard ASP.NET 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 SharePoint 2013.

For information about the pluggable authentication framework provided by ASP.NET, see ASP.NET developer documentation.
Authorization, users, groups, and the object model in SharePoint 2013

SharePoint 2013

Other Versions

Microsoft

5 out of 11 rated this helpful - Rate this topic

Last modified: July 01, 2013

Applies to: SharePoint Foundation 2013 | SharePoint Server 2013

In this article

Security policy
Users, groups, and principals
High-level view of object relations—scopes, users, groups, and roles
User tokens and access control lists
Object model changes: obsolete but backward-compatible security objects
Guest roles (Limited Access) to accommodate shared resources
Removing users from a scope or from all scopes

In this section

Additional resources

In SharePoint 2013, access to websites, 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 SharePoint 2013 objects.

To give a user access to an object, you can add the user to a group that already has permissions to the object, or you can create a role assignment object, set the user for the role assignment, optionally bind the role assignment to the appropriate role definition with base permissions, and then add the assignment to the collection of role assignments for the list item, folder, list, or website. If you do not bind the role assignment to a role definition when assigning a user to a role, the user has no permission.

Following are ways that SharePoint 2013 provides to control access to its objects:

- Objects can use the same permissions as the parent website, 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, enabling fine management of user access to objects.

• Groups consist of users and may or may not be assigned to roles. SharePoint 2013 includes the following three groups by default:
  o owners (administrator)
  o members (contributor)
  o visitors (reader)

When you create a website 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 website without having to enable anonymous access.

• Site creation rights (CreateSSCSite and ManageSubwebs) control whether users can create top-level websites, subsites, or workspaces.

Users become members of a SharePoint object indirectly through a group that has a role assignment, or directly through a role assignment. Users also 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 enumeration. Each user or group has a unique member ID.

You can use the object model to create or modify role assignments and definitions differently than the way you can through the functionality of the addrole.aspx file and the editrole.aspx file. Unlike these pages, which are presented in the user interface, the object model does not enforce rights dependency, so you can create a role definition with an arbitrary combination of rights. But, plan carefully when using the object model to customize role definitions and permissions, because a poorly planned role definition and inappropriately assigned rights can lead to a bad user experience.

For more information about SharePoint 2013 rights, see SPBasePermissions.

Security policy

A security policy provides a way 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 SharePoint 2013 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 level or at the 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 through 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 is given a higher priority than 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 website, list, folder, or document: You add users or groups and assign them to one or more role definitions. 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.

**Note**

Central administration policy roles differ from the role definitions for a site collection.

**Users, groups, and principals**

An individual user (SPUser) gains access to a SharePoint object directly through an individual role assignment, or indirectly through membership in either a domain group or a SharePoint group (SPGroup) that has a role assignment. In a direct role assignment, the user is the principal (SPPrincipal). In a domain group or SharePoint group role assignment, the domain group or SharePoint group is the principal.

SharePoint Server supports Windows users (for example, DOMAIN\User_Alias) and external users (through pluggable authentication). The user identity is maintained by the identity management system (for example, the Active Directory directory service). The user profile (which includes the user's display name, email address, and other information) is scoped to the site-collection level. Changing a display name affects the entire site collection.

A group is a collection of users through which SharePoint Server manages security. User-based management is straightforward for simple sites, but becomes more complex as the number of uniquely secured resources grows. For example, a user may have the Contribute role for list 1, the Read role for list 2, and the Design role for list 3. This model does not scale well if there are, for example, 50,000 users—which would result in access control lists (ACLs) being 50,000 access control entries (ACEs) long on every uniquely secured object.
Groups provide an answer to the manageability and scale problems of user-based permissions management. Group-based management may be more abstract or more difficult to conceptualize, but it enables easier management of complex sites with many uniquely secured objects. For example, when adding a user to a group that has already been granted the appropriate role on various objects in the system. The permissions checking for groups scales better because far fewer group ACEs need to be stored.

SharePoint Server supports two kinds of groups: domain groups and SharePoint groups. Domain groups remain outside SharePoint Server control; users cannot use SharePoint Server to define, browse, or modify domain group membership. SharePoint groups are scoped to the site-collection level, and they can be used only within the site collection. Domain groups can be used anywhere within the scope of the Active Directory directory service.

A principal is a user or group that is used to control security. If you add a user to a site, the user is the principal, but if you add a group to the site, the group is the principal. The key to scaling security in SharePoint Server is to keep the number of principals per scope reasonable. By using groups, a smaller number of principals can be used to grant access to a much larger number of users.

**High-level view of object relations—scopes, users, groups, and roles**

Figure 1 shows a high-level view of the SharePoint Server security management system in a logical database diagram. Each box represents a security object in the system. The lines represent relationships between the objects. The 1 and N notation represents the type of relationship. The figure shows how permissions data is structured into a user token and an ACL.

**Figure 1. Authorization object relations**
A scope represents a uniquely secured object or set of objects. You can scope to site, list, folder or item level.

Users and groups have a many-to-many relationship (N to N). Each user (SPUser) can be a member of multiple groups, and each group (SPGroup) can contain multiple users.

Rights and role definitions also have a many-to-many relationship (N to N). Each right (SPBasePermissions) can be part of multiple role definitions. For example, the Insert List Items right is included in the Contributor, Designer, and Administrator role definitions. Each role definition (SPRoleDefinition) can also contain multiple rights. For example, Contributor includes the rights for inserting, updating, and deleting list items.

Role definitions and role assignments (SPRoleAssignment) have a one-to-many relationship (1 to N). Each role definition is used in multiple role assignments. The readers on list 1 and the readers on list 2 may be different, but their role assignments can share a single role definition: Reader.

Users or groups and role assignments have a many-to-many relationship (N to N). Each user or group can be a member of multiple role assignments on a given object. For example, a user may have both the Designer role and the Administrator role on the same object.

Scopes and role assignments have a one-to-many relationship (1 to N). Each scope has multiple role assignments, but each role assignment has only one scope. For example, one user may be a
reader on the Events list, and another user may be a contributor on the Events list, but neither of these role assignments applies to the Announcements list. The only way for two lists to share the same role assignment is by inheriting their permissions from the parent container, in which case the security scope is the container, not the two lists.

**User tokens and access control lists**

To make checking permissions faster, SharePoint Server implements user tokens and ACLs in its security model. The user token identifies the authentication process applied to a user. A Windows user has a complex token: a unique string for the user (SID) and a list of all the Windows domain groups for the user (for example, DOMAIN\Department 15688). A user who does not have Windows authentication may have a very simple token with a unique string for the user name, or a complex token with group/role membership just as expressed in Windows authentication. SharePoint group membership for each user is expressed through a user token so that, by reading the user token, SharePoint Server identifies all groups for the current user.

An ACL is a binary object that determines the rights that users and groups have on a given object. An ACL consists of multiple ACEs, each security principal (user or group) being one ACE in the ACL. Rights, role definitions, and role assignments are structured into an ACL for each scope, so that SharePoint Server knows what each user or group is allowed to do within the given scope.

**Object model changes: obsolete but backward-compatible security objects**

In SharePoint 2013, all object scopes share the same basic permissions management experience. SharePoint 2013 manages permissions through role definitions, which enable a consistent experience at the list, folder, and item level. The following security objects used in Windows SharePoint Services 2.0 are obsolete, but continue to function for backward-compatibility:

- `Microsoft.SharePoint.SPPermission`
- `Microsoft.SharePoint.SPPermissionCollection`
- `Microsoft.SharePoint.SPRights`
- `Microsoft.SharePoint.SPRole`
- `Microsoft.SharePoint.SPRoleCollection`

To assign users to roles, use members of the `Microsoft.SharePoint.SPRoleAssignment` class and the `Microsoft.SharePoint.SPRoleAssignmentCollection` class. The `SPBasePermissions` enumeration, which replaced `SPRights`, includes additional permissions. The `SPBasePermissions` enumeration also includes legacy permissions that map to the same
constant values as previous permissions in SPRights. The SharePoint group concept maps to the existing SPGroup object and SPGroupCollection object, which represent cross-site groups.

**Policy roles: create or modify security policies for URL zones**

To create or modify security policies for URL zones, use the following classes and their members:

- Microsoft.SharePoint.Administration.SPPolicy
- Microsoft.SharePoint.Administration.SPPolicyCollection
- Microsoft.SharePoint.Administration.SPPolicyRole
- Microsoft.SharePoint.Administration.SPPolicyRoleCollection
- Microsoft.SharePoint.Administration.SPPolicy.SPPolicyRoleBindingCollection
- Microsoft.SharePoint.Administration.SPPolicyPermissions

**Guest roles (Limited Access) to accommodate shared resources**

The concept of a guest role is to accommodate the shared resources in the platform. For example, the theme and navigation structure of the website must be used to render the page for a list view. This concept is extended to include folder-level permissions and list-level permissions.

The SharePoint object model continues to call this the Guest role for semantic compatibility with the previous object model, although in the user interface the role is now called Limited Access.

**Folder and item extensions**

When a user is granted permissions on a folder, they are also granted the Guest role on the parent list of that folder and on the parent website of that list—on every uniquely secured scope above the folder, all the way to the first unique ancestor website. This is also true for list items: granting a user permissions on an item also grants that user the Guest role on all parent folders, lists, and websites up to the first unique ancestor website.

**Removing users from a scope or from all scopes**

Removing a user from a scope also removes that user from all uniquely secured scopes beneath the current scope. For example, removing a user from a website also removes that user from uniquely secured lists in the site.
The only way to remove a user from all scopes is to delete that user from the site collection, which removes the user from all roles in all scopes within the site collection.
Role, inheritance, elevation of privilege, and password changes in SharePoint 2013

Roles, role definitions, and role assignments

A role consists of two parts: a role definition and a role assignment.

The role definition, or permission level, is the list of rights associated with the role. A right is a uniquely controllable action within a SharePoint website. For example, a user with the Read role can browse pages in the website and view items in lists. User permissions are never managed directly by using rights. All user and group permissions are managed through roles. A role definition is a collection of rights bound to a specific object. Role definitions (for example, Full Control, Read, Contribute, Design, or Limited Access) are scoped to the website and mean the same thing everywhere within the website, but their meanings can differ between sites within the same site collection. Role definitions can also be inherited from the parent website, just as permissions can be inherited.

The role assignment is the relationship among the role definition, the users and groups, and the scope (for example, one user may be a reader on list 1, while another user is a reader on list 2).
The relationship expressed through the role assignment is the key to making SharePoint 2013 security management role-based. All permissions are managed through roles; you never assign rights directly to a user. You assign only meaningful collections of rights (role definitions) that are well-defined and consistent. You manage unique permissions by adding or removing users and groups to or from role definitions through role assignments.

The website administrator can customize the default role definitions and create additional custom roles by using the Manage Roles page, which lists the available role definitions in the site.

**Role definition inheritance**

SharePoint 2013 supports inheriting role definitions similarly to how it supports inheriting permissions, and breaking role definition inheritance requires also breaking permissions inheritance.

Each SharePoint object can have its own set of permissions or inherit its permissions from its parent container. SharePoint 2013 does not support partial inheritance, where an object would inherit all the permissions of its parent and also have some of its own permissions. Permissions are either unique or inherited. SharePoint 2013 does not support directed inheritance. For example, an object can inherit only from its parent container, not from some other object or container.

When a website inherits role definitions, the roles are read-only, like the read-only permissions in an inherited website. The user can navigate to the parent site that holds the unique role definitions via a link. The default setting for all new websites, even sites with unique permissions, is to inherit role definitions from the parent website. When the permissions are unique, role definitions can be reverted to inherited role definitions or edited as local role definitions.

Role definition inheritance in a website affects permissions inheritance following these rules:

- Cannot inherit permissions unless it also inherits role definitions.
- Cannot create unique role definitions unless it also creates unique permissions.
- Cannot revert to inherited role definitions unless it also reverts all unique permissions within the website. The existing permissions are dependent on the role definitions.
- Cannot revert to inherited permissions unless it also reverts to inherited role definitions.

The permissions for a website are always tied to the role definitions for that website.

**Managing user tokens**
SharePoint fetches user token information from the SharePoint database. If the user has never visited the site or if the user’s token was generated more than 24 hours previously, SharePoint generates a new user token by trying to refresh the list of groups that the user belongs to.

If the user account is an NT account, SharePoint uses the AuthZ interface to query the Active Directory directory service for the TokenGroups property. This may fail if SharePoint is running in an extranet mode, and does not have permission to query Active Directory for this property.

If the user account is a membership user, SharePoint queries the ASP.NET RoleManager for all the roles that the user belongs to. This may fail if there is not a proper .config file for the current executable file.

If SharePoint can't obtain the user's group memberships from Active Directory or <roleManager>, the newly generated token contains only the user's unique security ID (SID). No exception is thrown, but an entry is written into the ULS server log. The new token is also written into the SharePoint database so that it will not be regenerated within 24 hours.

After SharePoint obtains a fresh token, from the SharePoint database or by generating a new token, SharePoint sets the timestamp to be the current time and then returns it to the caller. This guarantees that the token is fresh for 24 hours.

After the SPUserToken object is returned to the caller, it is the caller's responsibility to not use the token after it is expired. You can write a helper utility to track the token expiration by recording the time when you get the token, and compare the diff with current time against SPWebService.TokenTimeout.

If an expired token is used to create a SharePoint website, an exception is thrown. The default token timeout value is 24 hours. It can be accessed via SPWebService.TokenTimeout.

**Elevation of privilege**

Elevation of privilege, a feature that was added in Windows SharePoint Services 3.0, enables you to programmatically perform actions in code by using an increased level of privilege. The SPSecurity.RunWithElevatedPrivileges method enables you to supply a delegate that runs a subset of code in the context of an account with higher privileges than the current user.

The following is a standard use of RunWithElevatedPrivileges.

```c#
SPSecurity.RunWithElevatedPrivileges(delegate()
{
```
Frequently, to perform actions in SharePoint, you must get a new `SPSite` object to effect the changes. For example:

```csharp
SPSecurity.RunWithElevatedPrivileges(delegate()
{
    using (SPSite site = new SPSite(web.Site.ID))
    {
        // Do things by assuming the permission of the "system account".
    }
});
```

Although elevation of privilege provides a powerful technique for managing security, it should be used with care. You should not expose direct, uncontrolled mechanisms for people with low privileges to circumvent the permissions granted to them.

**Important**

If the method passed to `RunWithElevatedPrivileges` includes any write operations, the call to `RunWithElevatedPrivileges` should be preceded by a call to either `SPUtility.ValidateFormDigest()` or `SPWeb.ValidateFormDigest()`.

## Automatic password changes

The automatic password change feature enables you to update and deploy passwords without performing manual password update tasks across multiple accounts, services, and web applications. This makes managing password in SharePoint 2013 simpler. You can use the automatic password change feature to determine whether a password is about to expire and to reset the password by using a long, cryptographically strong random string.

**Managed account**

You use managed accounts to implement the automatic password change feature. Managed accounts improve security and ensure application isolation. With managed accounts, you can:

- Configure the automatic password change feature to deploy passwords across all services in a farm.
- Configure SharePoint web applications and services, that are running on application servers in a SharePoint farm, to use different domain accounts.
- Map managed accounts to various services and web applications in a farm.
- Create multiple accounts in Active Directory Domain Services (AD DS), and then register each of these accounts in SharePoint.

You can also register managed accounts and enable SharePoint 2013 to control account passwords. Users have to be notified about planned password changes and related service interruptions, but the accounts used by a SharePoint farm, web applications, and various services can be automatically reset and deployed within the farm as necessary, based on individually configured password reset schedules.

Operations that you can use the SPManagedAccount class to perform include:

- Change password
- Set a password change schedule
- Propagate password change
- Find out when a password was last changed
- Enforce minimum length for password

For more information about the managed account API, see the following links:

- SPManagedAccount
- SPManagedAccount.EventProcessingOptions
- SPManagedAccount.EventType
Configuration, administration, and resources in SharePoint 2013

SharePoint 2013
Other Versions

For information about planning, administration, configuration, deployment, migration, upgrades, setting up claims, or security in general, see the IT and administrators documentation and guidelines on Microsoft TechNet.

The following are links to some of the documentation and guidelines on TechNet:

- Plan for authentication methods
- Configure claims authentication
- Security and protection for SharePoint Foundation 2010
- Custom claims providers for People Picker
- People Picker and claims provider planning
- Plan security for sites and content
- Plan security hardening
- Plan sandboxed solutions
- Plan site permissions
- Plan automatic password change
- Security and permissions administration
- User permissions and permission levels
- Business Connectivity Services security overview
- Configure forms-based authentication for a claims-based web application
- Configure the security token service
- Configure Kerberos authentication for the claims to Windows Token Service
- Security cmdlets (SharePoint Foundation 2010)

Note
For information about Windows Identity Foundation (WIF), see the Windows Identity Foundation SDK.

The following tables list blogs, forums, and additional resources about claims-based identity and general security in SharePoint Server.

Table 1. Blogs

<table>
<thead>
<tr>
<th>Blog Name</th>
<th>Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>Share-n-dipity</td>
<td>Blog home page</td>
</tr>
<tr>
<td>Microsoft SharePoint Developer Team Blog</td>
<td>Blog home page</td>
</tr>
</tbody>
</table>

Table 2. Forums

<table>
<thead>
<tr>
<th>Forum Name</th>
<th>Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>SharePoint Products and Technologies Forums</td>
<td>Forum home page</td>
</tr>
</tbody>
</table>

Table 3. Additional resources

<table>
<thead>
<tr>
<th>Additional Resources</th>
<th>Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>Claims and Security: Technical Articles Node</td>
<td>Claims and security: Technical articles and code samples on MSDN</td>
</tr>
<tr>
<td>Security and Authentication for SharePoint Foundation 2010</td>
<td>Security and Authentication for SharePoint Foundation 2010 Resource Center on TechNet</td>
</tr>
<tr>
<td>Microsoft SharePoint Foundation 2010 on TechNet</td>
<td>Microsoft SharePoint Foundation 2010 on TechNet</td>
</tr>
<tr>
<td>Security Resource Center</td>
<td>Security Resource Center on MSDN</td>
</tr>
<tr>
<td>Security Resource Center</td>
<td>Security Resource Center on TechNet</td>
</tr>
<tr>
<td>Windows Identity Foundation (WIF) SDK</td>
<td>Windows Identity Foundation documentation on MSDN</td>
</tr>
<tr>
<td>Microsoft patterns &amp; practices</td>
<td>A guide to claims–based identity and access control</td>
</tr>
<tr>
<td>.NET Framework Class Library</td>
<td>.NET Framework Class Library</td>
</tr>
</tbody>
</table>