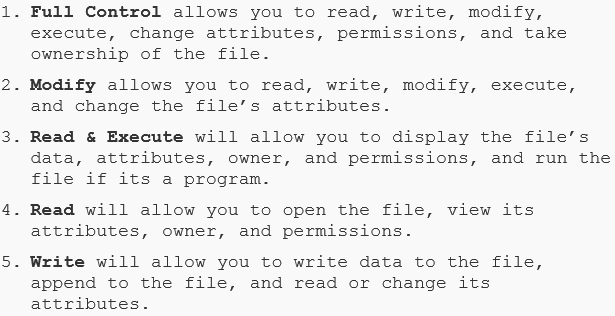
**NTFS and SHARE PERMISSIONS**In a Windows environment, *permissions* are security settings that control access to individual objects, such as files or folders. Permissions determine which specific actions users can perform on a given object. You can assign permissions by modifying an object's properties.

**NTFS FILE PERMISSIONS**

|  |  |
| --- | --- |
| **PERMISSION** | **ENABLES THE USER TO** |
|  |  |
| FULL CONTROL | Change permissions, take ownership, and perform all other tasks |
| MODIFY | Modify and delete the file |
| READ+EXECUTE | Run applications and perform Read tasks |
| READ | Read the file and view file attributes, ownership, and permissions |
| WRITE | Overwrite the file and change file attributes |



**Special Permissions**

Each of the standard NTFS permissions is made up of several more granular permissions called special permissions. Standard permissions are the most frequently assigned groups of permissions; special permissions provide you with a finer degree of control, granular.

For example, the standard Read permission is made up of the following special permissions:

• List Folder/Read Data

• Read Attributes

• Read Extended Attributes

• Read Permissions

**File Permissions in Windows XP Home**

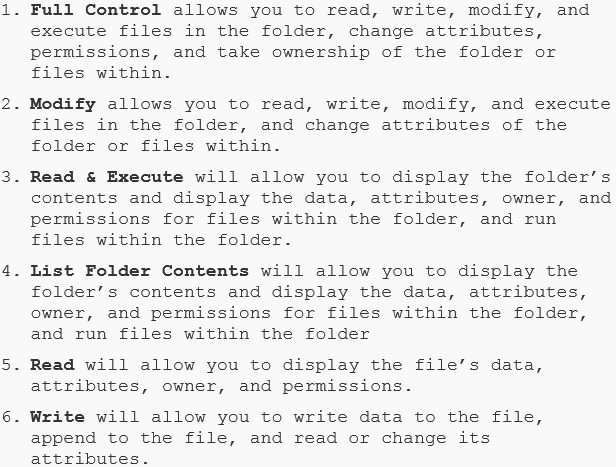
The Classic security model is not available in Windows XP Home, so you will not be able to set individual NTFS permissions on Windows XP Home computers. However, when Simple File Sharing is active, both Windows XP Professional and Windows XP Home provide a rudimentary mechanism for protecting or sharing local files when multiple users use the same computer.

• To protect files, individual users can mark user profile folders such as My Documents as **Private.** Other local users will not be able to access these folders.

• To share files, users can place files and folders in the Shared Documents folder in **My Computer.** All local users will be able to access the contents of Shared Documents. The SharedDocuments folder is also shared on the network and can be accessed by users at othercomputers.

**NTFS FOLDER PERMISSIONS**

|  |  |
| --- | --- |
| **PERMISSION** | **ENABLES THE USER TO** |
|  |  |
| FULL CONTROL | Change permissions, take ownership, delete subfolders and files, and perform all other tasks |
| MODIFY | Delete the folder and perform Write and Read & Execute tasks |
| READ+EXECUTE | Perform the same functions as Read and List Folder Contents tasks, as well as execute files |
| LIST FOLDER CONTENTS | View the names, attributes, and permissions of subfolders in the folder, but only see the names of files within the folder |
| READ | View names, attributes, permissions, and contents of files and subfolders in the folder |
| WRITE | Create new files and subfolders in the folder, and change their attributes |



**Permissions Inheritance**

Permissions that you assign to a folder are inherited by files and folders within that folder. It is generally most efficient to group similar files together in a folder and assign permissions to the folder, rather than to the individual files. Inherited permissions are indicated by gray background check marks in the file or folder's security properties.  
  
Inherited permissions is a DEFAULT attribute of NTFS permissions. Inherited permissions allow NTFS settings for a folder be applied to its contents and all objects and folders contained within the top folder.

Inheritance is fairly easy to understand when all defaults are used. But when inheritance is blocked, it becomes more difficult to troubleshoot. This difficulty is manifested when a folder deep within another folder has the Inherit Permissions option cleared. In troubleshooting inherited permissions, it is best to start at the root of the problem and work your way up the folder structure.

When you clear inheritance of NTFS permissions from a parent container, you are presented with two options: Copy and Remove. The Copy option will recurse (include) the child objects and write the NTFS permissions from the parent folder. The Remove option removes all default NTFS-created permissions—that of Administrators, Users, Creator Owner, System, etc.—from the list of Group or User Names. Exercise caution when using the Remove option on inheritance blocking!

**Permissions Propagation**

If you modify the permissions for a parent folder, you can choose whether or not to propagate the changes downwards, which means to apply those permissions changes to all of the subfolders within the folder.

**Shared Files and Folders**

A *share* is any network resource that is available to other computers or users on the network. Typical shares include folders, printers, and drives. Because shares enable users to access a computer system from a remote location, you should secure all shared resources against unauthorized access.

There are two kinds of shares: *administrative shares* and *local shares*.

• **Administrative shares** are hidden shares that are created and shared by default on every Windows system. They are displayed with a "$" to indicate that they are hidden files. Although you can delete these administrative shares, the system will re-create them every time the system restarts. Anyone with administrator access to the system can interact with administrative shares.

• **Local shares** are folders that are created on the local network by system users and then shared with other network users by using shared folder permissions. Users, including administrators, can delete local shares, and they are not automatically created upon restart.

**File Sharing with Windows**

On Windows systems, you can share a folder by modifying the folder's properties. When you share a folder, you assign it a share name that can be different from the underlying folder name. You can share the folder more than once using different names.

Users can connect to the shared folder by browsing to the computer in **Network,** or by selecting **Start→Run** and entering the Universal Naming Convention (UNC) path to the folder, in the form **\\computername\sharename.**

**Viewing Shares on the System**

You can see all shares on a system, including administrative shares, by opening **Computer Management,** expanding **Shared Folders,** and selecting the **Shares** node. You’ll see thefollowing administrative shares on every Windows system:

• The root of each drive on the system is shared with its drive letter. (C$, the D drive is shared as D$, and so on)

• The folder where Windows is installed, usually the C:\Windows folder, is shared as ADMIN$.

**SHARE PERMISSIONS**

|  |  |
| --- | --- |
| **PERMISSION** | **ENABLES USERS TO** |
|  |  |
| FULL CONTROL | * Perform all Read and Change tasks |
|  | * Change NTFS permissions on files and folders inside the shared folder |
|  |  |
| CHANGE | * Perform all Read permission tasks |
|  | * Add files and subfolders |
|  | * Change file contents |
|  | * Delete subfolders and files |
|  |  |
| READ | * View file and subfolder names |
|  | * View file contents and file attributes |
|  | * Run program files |
|  | *The Read permission is granted by default to the Everyone group when a folder is shared and to new users when they are added to the Permissions list.* |

**NTFS vs. Share Permissions**

NTFS permissions apply to the actions that users can take on a file or folder either on the network or locally. Share permissions apply only to the folders (and possibly subfolders and files) that have been shared with other users and are being accessed over the network. Using both NTFS and share permissions on the same files and folders may seem like overkill, but they are often used together, and it is important to understand the differences between the two permissions and how they interact with one another.

In Windows, a shared folder has two sets of permissions: the NTFS permissions (which are on the **Security** tab of that folder's **Properties**) and the share permissions (which are on the **Shared** tab of that folder's **Properties**). The security permissions do not automatically change once a folder is designated as a share, and there is no propagation between the two. A folder can have NTFS permissions assigned, and then be shared and have share permissions assigned. When a user accesses the folder over the network, both the share and NTFS permissions are applicable, and the most restrictive of the two sets of permissions applies. So, if the network user has the Full Control NTFS permission but only the Read share permission, the user will have only the ability to read the contents of the folder.  
  
The best way to distinguish share permissions from NTFS permissions is to consider share permissions as an entry point to the resources (from over the network). Only after the share permissions offer Change and/or Full Control can the NTFS permissions of that type be used.

The combination of share-level and NTFS permissions can seem like administrative overhead, but consider this: Share permissions act as a point of entry for the NTFS permissions over the network. When you enter a network resource through a share, the share permissions dictate what you can do through the share as a whole. The NTFS permissions dictate what you can do to specific files and folders. In the troubleshooting mode, identify whether share-level permissions can be ruled out of the issue.   
  
**Combining Shared Folder Permissions and NTFS Permissions**

When using share permissions and NTFS permissions together, if there is a conflict in the configuration, the most restrictive permission prevails. For example, if a user has NTFS full access to a specific file in a folder that is not shared, the user cannot access the file from the network. In this case, the user can sit down at the computer that contains the file, log in and access the file, because sharing permissions do not affect local access.

One strategy for providing access to resources on an NTFS volume is to share folders with the default shared folder permissions and then control access to shared folders by assigning NTFS permissions. When you share a folder on an NTFS volume, both shared folder permissions and NTFS permissions combine to secure file resources.

Shared folder permissions provide limited security for resources. You gain the greatest flexibility by using NTFS permissions to control access to shared folders. Also, NTFS permissions apply whether the resource is accessed locally or over the network.

When you use shared folder permissions on an NTFS volume, the following rules apply:

* You can apply NTFS permissions to files and subfolders in the shared folder. You can apply different NTFS permissions to each file and subfolder that a shared folder contains.
* In addition to shared folder permissions, users must have NTFS permissions for the files and subfolders that shared folders contain to gain access to those files and subfolders.
* When you combine shared folder permissions and NTFS permissions, the more restrictive permission is always the overriding permission.

**Permissions Considerations**

There are some important considerations that you should keep in mind when applying permissions to files and folders.

* **ALLOW VS DENY** - When choosing whether to allow or deny an action using permissions, you need to choose carefully between the two. Deny is more restrictive than Allow. If the Deny property is applied on either a file or a folder, it will override any Allow permissions that may have been granted to the user. Therefore, use of the Deny permission should be done sparingly. You should deny permissions (using explicit Deny) only to a specific user when it is necessary to override permissions that are otherwise allowed for the group to which this user belongs.

When establishing permissions, administrators can specify whether the entry being added should have access (Allow) or not have access (not Allow) to the resource. It is more practical to clear all the Allow check boxes for a group or a user, in effect denying them access to the resource without using the absolute Deny option. “Not-Allow” access in this way is easier to troubleshoot, manage, and configure.

* **NTFS PERMISSIONS** are cumulative. That means that a user’s effective permissions are the result of combining the user’s assigned permissions and the permissions of any groups the user belongs to.
* **SHARE PERMISSIONS** ONLY apply to users who access the resource over the network. They don’t apply if you log on locally, for example through terminal services.
* **AVOID NESTED SHARES** Avoid having nested shares in your file structures because they can create conflicting behavior for the same network resources if accessed through different shares. This can be asking for trouble, especially when the share permissions are different. A nested share is a shared folder that resides in a separate shared folder. There are, of course, the default hidden shares (C$, D$, etc.), which make all shares nested beneath them, and they're a default. However, if your users use two separate nonhidden shares that are nested, there can be conflicting share permissions.
* **DISTIGUISH BETWEEN BASIC AND SPECIAL NTFS PERMISSIONS** Special permissions give more options to particular access requirements. **NOTE:** Using special permissions will increase the administrative overhead associated with NTFS permissions (more complicated). A best practice would be to use the special permissions only when needed. The standard NTFS permissions provide most of the necessary functionality to offer secure access to shared and local resources. However, there are scenarios where using the special permissions makes sense. **NOTE:** Be sure to rule out special permissions in troubleshooting. Every administrator has at one point not been sure of the application of various permissions—share permissions, NTFS permissions, group memberships, multiple user accounts, etc. Taking a quick look at the special permissions can quickly provide a hint as to whether they're part of the issue you are troubleshooting.
* **KEEP RESOURCES WARRANTING SPECIAL PERMISSIONS SEPARATE** If the scenario permits, it can be a good practice to keep resources requiring special permissions grouped in separate shares or folders with other resources that have special permissions. Having standard permissions intermixed with special permissions in the same location can add administrative overhead.
* **NEVER OVER-PRIVILEGE** A common misstep is to provide too many rights—usually through group memberships—to users for access to resources. Especially if you are using Active Directory, a clearly organized structure with the membership and access requirements defined will lend to a more correctly administered user or group. Take the firewall stance of granting **ONLY** that which is explicitly required.

Group membership is one of the easiest ways to over- or under-privilege access to resources. Especially in domain configurations, the complexity is increased by multiple memberships and/or nested groups. Use the Effective Permissions tool to see what the resultant set of access is, determined by group membership when using Active Directory. Although this is not a direct display of NTFS permissions, you can then examine each group membership for an object as part of troubleshooting NTFS permissions.

* **KNOW WHEN TO COPY AND WHEN TO MOVE** Standard copy and move operations deliver default results that can maintain your configured permissions—or break them. A good way to remember this is that copy operations will create the permissions of the destination container, and move operations will maintain that of the parent container.

**Memorization mechanism:** CC/MM — CopiesCreate/MovesMaintain or CopiesCreate/MovesMake.

Of course, there is also the need to copy resources *and* maintain NTFS permissions that would be difficult to re-create. You can use XCOPY with the /O and /X parameters to perform this type of function. Using XCOPY with these parameters will allow copy operations to copy the files and/or folders to a new location and create them with the NTFS permissions equal to that of the source container.

**File Attributes**

A characteristic that can be associated with a file or folder that provides the operating system with important information about the file or folder and how it is intended to be used by system users. There are several standard attributes that can be enabled for files or folders on Windows systems.

|  |  |
| --- | --- |
| **FILE ATTRIBUTE** | **DESCRIPTION** |
|  |  |
| ARCHIVE (A) | Indicates that a file has not been backed up. Windows automatically sets the Archive attribute on any file you create or modify. When you back up data, you can choose to back up only the files on which the Archive attribute is set. |
| HIDDEN (H) | Hides a file from view in file management tools such as Windows Explorer, Computer in Windows Vista and Windows 7, or My Computer in Windows XP. |
| READ-ONLY (R) | Enables users to read the contents of a file or execute it if it is a program file, but prevents users from changing the contents of a file. |
| SYSTEM (S) | Indicates that a file is used by the operating system. Some applications use this attribute to restrict user access to these files. The System attribute in Windows automatically hides the file or folder. |
| INDEX (I) | This Windows-specific attribute enables the Windows Indexing Service to create an index of the file to speed up the Search function. |

**Viewing and Changing Attributes**

You can view or change most attributes of a file or folder object by opening the properties of the object in Windows Explorer. You can view and manage the System attribute at the command line by using the attrib command. For information on the functions and syntax of the attrib command, see the Windows Help system.

**NTFS and SHARE LINKS**

<https://technet.microsoft.com/en-us/library/2005.11.howitworksntfs.aspx>

<https://technet.microsoft.com/en-us/magazine/2006.01.howitworksntfs.aspx>