WHITE PAPER

Using User Access Control Lists (ACLs) to manage file permissions with a StorCenter™ NAS (applies to EMC LifeLine 3.x)

A Detailed Review

iomega
an EMC company

EMC²

SEPTEMBER 2012

EXECUTIVE SUMMARY

In a UNIX or Linux environment, the Network File System (NFS) protocol is used to access file systems. In a Windows environment, the Common Internet File System (CIFS) protocol is used to access file systems. The StorCenter device supports a mixed NFS and CIFS environment by providing multiprotocol access capabilities to enable UNIX and Windows users to share the same file systems.

The StorCenter device is powered by enterprise-class EMC® LifeLine™ software to offer easy-to-use, powerful, and affordable network storage for any small business or remote office. In LifeLine version 3.x, Windows user access control is enforced at two levels: share level and directory level. Share-level access control is configured in the Iomega StorCenter Manager software on individual folders that are shared via CIFS. Directory-level access control is typically done using native Windows® tools and interfaces to configure more granular access control on directories and files inside a share. Whenever there are conflicts between share-level and directory-level access control on the same file system object, the most restrictive permission settings win.

This paper provides in-depth review of the Windows user access control capabilities supported on the StorCenter, the enabling technologies, as well as the known limitations. The goal is to help administrators define and execute a desired level of security in the Windows environment.

INTRODUCTION

This white paper includes the following sections:

- Relevant Windows concepts
- Overview of enabling technologies in LifeLine
- Share-level user access control
- Directory-level user access control
- Multiprotocol access control

To better understand Windows user access control, you may need to understand some relevant Windows concepts. These concepts are introduced first to help readers gain background knowledge.

EMC LifeLine software is a Linux-based operating system specialized for storage management. LifeLine relies on the underlying file system structure and an open source software package together to implement Windows user access control. These enabling technologies are described next to give readers an overview of the mechanism and components of the implementation.

A combination of share-level and directory-level user access control is provided by LifeLine. Whenever there are conflicts between share-level and directory-level access control on the same file system object, the most restrictive permission settings win. The share-level control is configured in the Iomega StorCenter Manager software. The directory-level control is typically configured using native Windows utilities, such as Windows Explorer and the Computer Management Microsoft Management Console (MMC). Capabilities as well as limitations are detailed in these sections. Lastly, in a multiprotocol environment where both NFS users and CIFS users access the same share, security settings on a file system object can be enforced and modified via either protocol. This presents the challenge of synchronizing security settings for both worlds. The policy that determines the behaviors in such an environment is described, including the mappings between UNIX permissions and Windows ACLs.

SCOPE

Most content in this white paper applies to the following conditions only:

- The StorCenter device is running LifeLine software release 3.x.
- Windows ACL on a share is enabled by checking the **Allow users to change file level security** checkbox in Iomega StorCenter Manager, as shown in Figure 1.

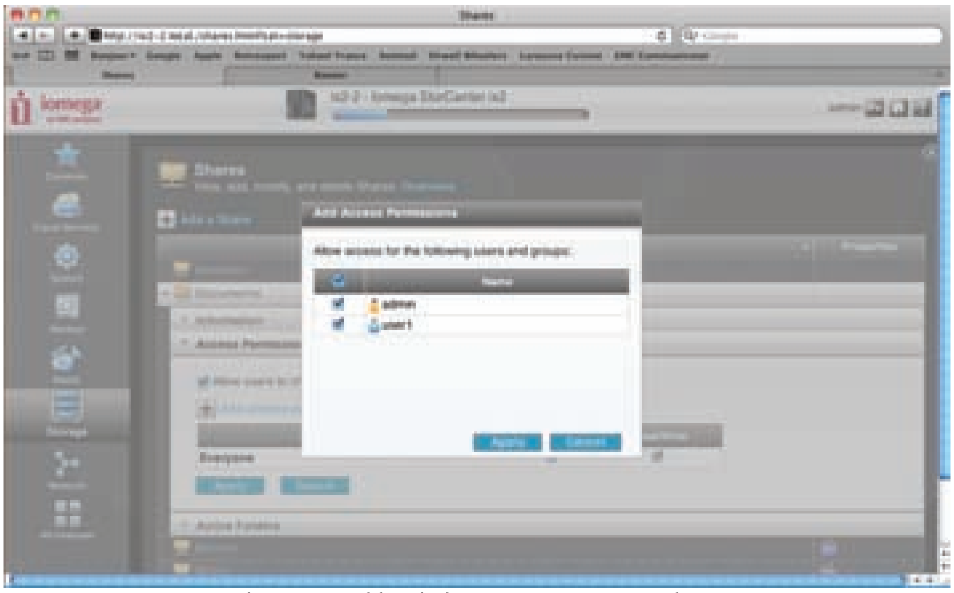


Figure 1. Enable Windows ACL support on a share

If a share is not secured, or it is secured but does not have Allow users to change the file level security enabled, the share can still be accessed from a Windows host. However, the share, and every directory and file inside the share do not have Windows ACL associated. Windows user access control is only enforced according to what is configured on the share using Iomega StorCenter Manager. Hereafter, any mentioning of a secured share is set in the context that the share is already enabled to support Windows ACL unless being called out otherwise as it appears in the paper.

AUDIENCE

This white paper is intended for system administrators responsible for implementing the StorCenter device in their Windows, or multiprotocol mixed Windows and UNIX environment. Readers should be familiar with the concepts of both Windows and UNIX user access control. Windows system administration experience is expected.

TERMINOLOGY

- Folder: on a StorCenter network storage device, a folder is a file system object and shared storage type that is created to be accessed by hosts using file sharing protocols, including NFS and CIFS. A secured folder requires authorized users to authenticate prior to access, while a public folder can be accessed by any user without authentication.
- Share: in this white paper, a folder is also referred to as a share. The terms are equivalent and interchangeable.
- Directory: in this white paper, a directory refers to a sub-level directory that is created within a share.
- Common Internet File System (CIFS): a distributed file system providing transparent access to remote file systems.
- Windows File Sharing: the practice of distributing or providing access to files using the CIFS protocol.

- **Security Identifier (SID)**: an alphanumeric string that is assigned by an authority, such as a Windows domain controller, to identify a Windows object, such as a user.
- **Access Control List (ACL)**: an ACL is a list of permissions attached to an object. An ACL specifies which users or system processes are granted access to an object, including operations which are allowed on the given object. Each entry in an ACL list is called Access Control Entry (ACE).
- **POSIX ACL**: the ACL implementation on Unix-like systems resulting from the POSIX 1003.1e/1003.2c working group.
- **UNIX permissions**: a UNIX-like system manages file-system permissions in three distinct classes known as user, group, and others. In effect, UNIX permissions are a simplified form of ACL.
- **Samba**: a free software re-implementation of the SMB/CIFS networking protocol. It runs mostly on UNIX and Unix-like operating systems, such as Linux, to provide file and print services for Microsoft Windows clients and can integrate with Active Directory as part of a domain.
- **Network File System (NFS)**: a distributed file system providing transparent access to remote file systems. NFS allows all network systems to share a single copy of a directory.
- **User Identifier (UID)**: an integer value that is used in UNIX-like operating systems to identify a user within the kernel.
- **Group Identifier (GID)**: an integer value that is used in UNIX-like operating systems to identify a user group within the kernel.
- **Active Directory (AD)**: a technology created by Microsoft to provide a variety of network services including LDAP directory access, Kerberos authentication, DNS-based naming and network information, information security for user access to networked resources, among others. Active Directory also allows administrators to assign policies, deploy software, and apply critical updates in Windows environments.
- **Domain**: a logical group of computers that share a central directory database that contains the user accounts and security information for the resources in the domain.
- **Workgroup**: a workgroup is Microsoft's terminology for a peer-to-peer Windows computer network. Microsoft operating systems in the same workgroup may allow each other access to their files, printers, or Internet connection.

WINDOWS CONCEPTS

CIFS USER ID RESOLUTION

Every user of the StorCenter network storage, who is running either Windows or a UNIX-like operating system, must be identified by a unique user identifier (UID). Windows, however, does not use numeric IDs to identify users but instead uses strings called security identifiers (SIDs).

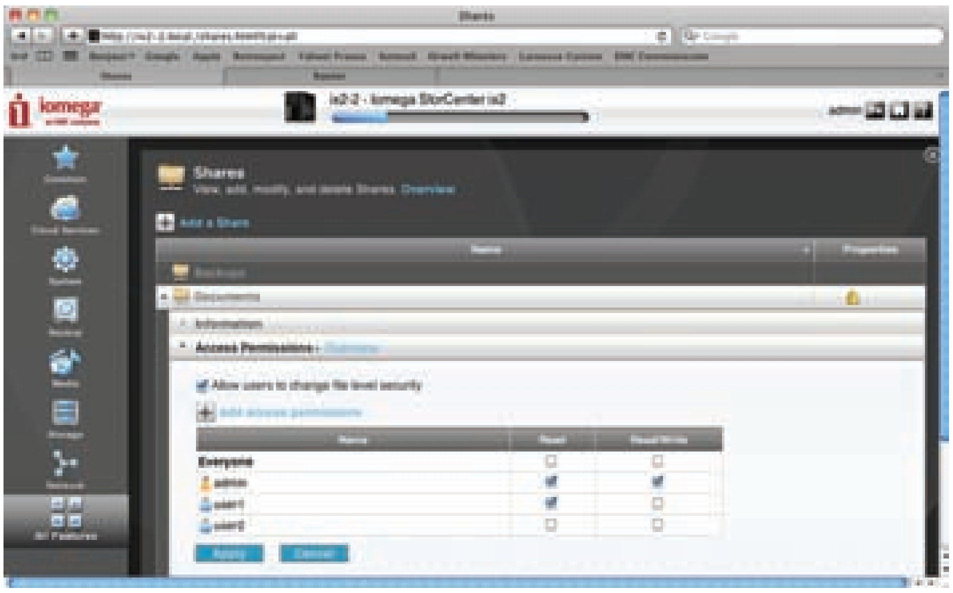
In a multiprotocol environment, a method of mapping Windows SIDs to UIDs is typically needed to synchronize user access. When a StorCenter device is configured to be in the Windows Workgroup mode, only users that are created on the NAS can access secured shares via CIFS. Each user is automatically assigned a UID during creation, and the UID can be modified. UNIX users that have matching UIDs as those assigned on the device can access secured shares with their access permissions being checked against their UIDs. Therefore, in the Windows Workgroup mode, a simple internal user mapping mechanism is applied.

When a StorCenter device is joined into a Windows Active Directory (AD) domain mode, users are imported from the AD to the storage device. No UIDs will be created for the imported users, and no user mapping is provided. All UNIX users accessing via NFS are treated as anonymous users, typically the nobody user, and can access secured shares with their access permissions being checked only against the anonymous user UID.

SECURITY ON FILE SYSTEM OBJECTS

In a multiprotocol environment, the StorCenter device uses its security policies to manage the access control of its file systems. The administrator can set user access to secured shares using the Iomega StorCenter Manager as shown in Figure 2.

Figure 2. Set user access to a secured folder UNIX security model


ix2-2 - Iomega StorCenter ix2
Shares
View, add, modify, and delete Shares. Overview
Add a Share
Documents
Information
Access Permissions
Apply
Cancel

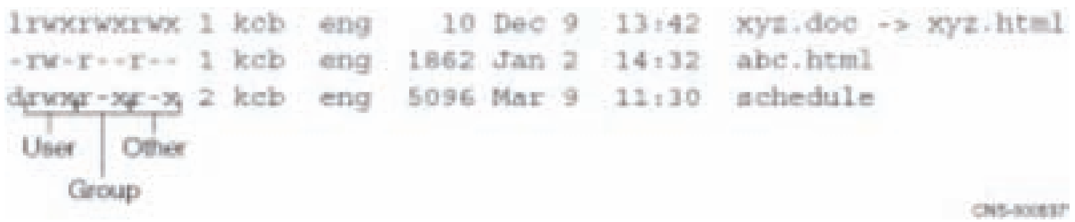
Allow users to change file level security

Name	Read	Read/Write
Everyone	☐	☐
admin	✓	✓
user1	✓	☐
user2	☐	☐

unix security model

UNIX access rights are referred to as the mode bits of a file system object. They are represented by a bit string in which each bit represents an access mode or privilege granted to the user owning the file, the group associated with the file system object, and all other users. UNIX mode bits are represented as three sets of concatenated rwx (read, write, execute) triplets for each category of users (user, or group, or other), as shown in Figure 3.

Figure 3. UNIX security model



```
lrwxrwxrwx 1 kcb  eng    10 Dec 9  13:42  xyz.doc -> xyz.html
-rw-r--r-- 1 kcb  eng  1862 Jan 2  14:32  abc.html
drwxr-xr-x 2 kcb  eng  5096 Mar 9  11:30  schedule
```

The example illustrates the following:

- The first character of each line indicates the file type: d for a directory, l for a symbolic link, or dash (-) for a regular file.
- The next nine characters of each line are the read/write/execute permission sets for user or group or other.
- kcb is the user and eng is the group.
- xyz.doc is a symbolic link that anyone can traverse to retrieve the xyz.html file.
- abc.html is a regular file that anyone can read but only the user kcb can write to it.
- Schedule is a directory that anyone can search and read, but only user kcb can insert files into it and delete files

from it.

UNIX also uses *umask* (UNIX shorthand for “user file-creation mode mask”), a four-digit octal number, to determine the file permission for newly created files. Every process has its own umask. The umask specifies the permissions you do not want given by default to newly created files and directories by doing a bitwise AND with the bitwise complement of the umask.

Windows security model

The Windows security model is based primarily on per-object rights, which involve the use of a security descriptor (SD) and an access control list (ACL). Access to a file system object is based on whether permissions have been set to Allow or Deny through the use of an SD. The SD describes the owner of the object and group SIDs for the object along with its ACLs. An ACL is part of SD for each object. Each ACL contains access control entries (ACEs). Each ACE in turn contains a single SID that identifies a user, group, or computer, and a list of rights that are denied or allowed for that SID. You can learn more about the Windows security model at http://www.microsoft.com/whdc/driver/security/drvsecure.mspx. The StorCenter device supports Windows ACLs at the share, directory, and file level. Share-level user permissions can be configured using the Iomega StorCenter Manager and a Windows host. Directory and file level ACLs can only be configured on a Windows host.

DETERMINING THE GID FOR FILE SYSTEM OBJECTS

In a file system, every object (such as a file, directory, link, and shortcut) has an associated owner and owner group that are identified by a UID and GID. NFS uses the UID and GID to control access to the file system object. Since a user can be a member of many groups, the StorCenter device needs a way to determine the group that should be associated with a newly created object. The user primary group setting determines which GID gets assigned to the file system object. NFS and CIFS have the concept of a primary group for a user. In NFS, the primary group is required; however, the primary group is optional on Windows platforms and defaults to the Domain Users group. Table 1 describes how the primary group mapping is determined for file system objects on the StorCenter device.

Protocol	Description
NFS	When a file system object (FSO) is created from a UNIX client, the GID can be determined in these ways: • In Domain mode, the FSO GID is automatically assigned to be the primary group of user “nobody”, which is group users. • In Workgroup mode, the FSO GID is taken from the GID supplied by the UNIX client based on the primary group of the creator.
CIFS	When an FSO is created from a Windows client, the GID can be determined in these ways: • In Domain mode, the FSO GID is taken from the GID associated with the primary group of the creator as defined in the AD. • In Workgroup mode, the FSO GID is taken from the UNIX primary group of the user as defined in the passwd file on the Iomega storage device.

Table 1. Determining the file system object GID

USER ACCESS CONTROL OF FILE SYSTEM OBJECTS

A windows user credential is built and cached when a user first connects to a StorCenter device through CIFS protocol. The credential contains the user SID and all the SIDs of the groups in which the user is a member. When using regular NFS authentication (AUTH_SYS), a UNIX user credential is sent along with the Remote Procedure

Calling (RPC) protocol request and consists of a UID and up to 16 GIDs to which the user belongs. When a user requests access to a file system object, the StorCenter device compares the user credentials with the permissions on that file system object.

FILE NAMING

NFS and CIFS use different file naming conventions. Table 2 explains these differences

NFS file names	CIFS file names
Case-sensitive.	Not case-sensitive, but they are case-preserving.
Allows a directory to hold files that have the same names, but differ in case.	Does not allow a directory to have two files with the same name and different cases, and identifies these names as duplicate names.

Table 2. NFS and CIFS file naming conventions

FILE LOCKING

File locking ensures file integrity when multiple users attempt to access the same file concurrently. File locks manage attempts to read, write, or lock a file that is in use by another user. Table 3 describes the different ways the NFS and CIFS protocols implement file locking.

NFSv2 or NFSv3 environment	CIFS or NFSv4 environment
Uses read locks and exclusive (write) locks	CIFS uses opportunistic locks (oplocks) and deny modes. The NFSv4 equivalent of oplocks are called delegations.
NFS locks are advisory but not mandatory. An advisory lock is not an enforced lock and therefore does not affect read and write access to the file. However, it advises other clients that the file is already in use.	The CIFS and NFSv4 protocols enforce strict file locking, as well as unlocked access to files. The CIFS and NFSv4 locks are mandatory. When a CIFS or NFSv4 process locks a file, other users are denied certain types of access to the file, depending on the type of lock imposed. A CIFS or NFSv4 client can lock a file by using: • A deny read/write access on the whole file. • A lock range on a portion of the file.

Table 3. File locking in multiprotocol environment

The StorCenter device only supports NFSv3. In a multiprotocol environment, a file might have locks set by CIFS and NFS users. Because NFSv3 locks and CIFS deny modes and oplocks are not directly equivalent, the StorCenter device must provide some control over the interaction of CIFS and NFS file locking:

- CIFS share modes, oplocks, and level 2 oplocks are supported.
- CIFS blocking lock is supported so that a CIFS client can choose to receive notification from the Iomega StorCenter device when a currently locked file becomes available for the client to obtain a lock.
- If a CIFS or NFSv3 client locks a file, no other client can lock that file.
- NFSv3 exclusive locks are enforced for CIFS client access — CIFS clients can only open a file read-only if an NFSv3 client has an exclusive lock on it.
- CIFS write locks are enforced for NFSv3 client access — NFSv3 clients can only open a file read-only if a CIFS client

has a write lock on it.

- NFSv3 clients can delete files locked by CIFS. Additionally, CIFS clients can delete files locked by NFSv3.

Note: The StorCenter device enforces file locks only when the client application is using locks. Some simple applications, such as Windows Notepad or WordPad, UNIX more, and Linux VI do not use file locking. Files created with these applications are not locked; they can concurrently be opened and edited by another application.

DISTRIBUTED FILE SYSTEM (DFS)

Microsoft Distributed File System (DFS) allows you to group shares located on different servers into a logical DFS namespace. A DFS namespace is a virtual view of these shares shown in a directory tree structure. By using DFS, you can group shares into a logical namespace and make shares that are distributed across multiple servers appear to users as if they reside in one place on the network. Users can navigate through the namespace without needing to know server names or the actual shares hosting the data.

Each DFS tree structure has a root target, which is the host server running the DFS service and hosting the name space. A DFS root contains DFS links that point to the shares on the network. The shares are referred to as DFS targets. Microsoft offers standalone and domain-based DFS root servers: the domain DFS root server and the standalone DFS root server. The domain-based DFS server stores the DFS hierarchy in the Active Directory. The standalone DFS root server stores the DFS hierarchy locally. The StorCenter device provides the same functionality as a Windows Server standalone DFS root server.

SYMBOLIC LINKS

Symbolic links are special nodes created by NFS clients that point to another node (a file or directory called the target node). A symbolic link is, in some respects, similar to a shortcut in the Windows environment. NFS symbolic links usually have no meaning to Windows clients, because the client must resolve the symbolic link to its target. However, the StorCenter device resolves symbolic links for Windows clients so that these clients can access the same files and directories as NFS clients through a symbolic link.By using symbolic links, CIFS clients can access multiple shares on a StorCenter device from a single share. This gives the appearance of one large namespace when it actually consists of individual shares linked together with symbolic links. If the Iomega NAS is able to access the target on behalf of CIFS users, the users see the target of the symbolic link rather than the link itself, and do not know that they have followed a symbolic link. If the target is not accessible, the user sees the symbolic link as a file, but cannot access that file.

The StorCenter device has the following rules regarding symbolic links:

- Both soft symbolic links and hard symbolic links are supported (hard symbolic links cannot be created for directories).
- When the target of a symbolic link resides in the same share, or in another share, the symbolic link is listed by Windows clients and can be followed to its target.
- When the target of a symbolic link does not reside in the same share or any other share on the Iomega storage device, the symbolic link cannot be seen by Windows clients.
- If a symbolic link refers to a target that does not actually exist, the symbolic link cannot be seen by Windows clients.
- If a symbolic link refers to a directory and a Windows client attempts to delete the symbolic link, only the link itself

is deleted. The contents of the directory referenced by the link remain intact.

Note: Do not use Microsoft Office applications on files represented by symbolic links. When a file is updated, Microsoft Office creates the updated file in the directory containing the symbolic link, instead of the symbolic link target directory. Therefore, the symbolic link is removed and a file with the same name replaces it unless the file is saved with a different name.

OVERVIEW OF ENABLING TECHNOLOGIES IN LIFELINE

SAMBA

The LifeLine software relies on Samba to work in a Windows environment. Samba is a free application, licensed under the GNU General Public License. Samba implements some of the services essential to the Windows domain and Active Directory interoperability so that UNIX and UNIX-like computer systems can participate in a Windows environment to share data and information. The primary component is a reimplementation (via reverse engineering) of the CIFS network protocol for file and printer sharing, authentication and authorization, name resolution, and service announcement.

Samba sets up network shares for chosen UNIX directories, including all contained subdirectories. These appear to Windows users as normal Windows shares accessible on the network. The same directories can also be accessed by NFS users concurrently in a multiprotocol environment. Samba services are implemented as two daemons: the smbd that provides the file and printer sharing services, and the nmbd which provides the Windows NetBIOS to IP address name service.

POSIX ACL

The LifeLine software is Linux-based and uses the XFS file system that supports POSIX ACLs, which are utilized by LifeLine to map to Windows ACLs for user access control.

A POSIX ACL consists of a set of entries. The permissions of each file system object have an ACL with each of the owner/group/other class of users represented by an ACL entry. These are called minimal ACLs. Permissions for additional users or groups occupy additional ACL entries.

The type of ACLs that define the current access permissions of file system objects is called **access ACL**. A second type called default ACL is also defined. The **default ACLs** define the permissions a file system object inherits from its parent directory at the time of its creation. Only directories can be associated with default ACLs. Default ACLs play no direct role in access checks.

When a directory is created inside a directory that has a default ACL, the new directory inherits the parent directory's default ACL both as its access ACL and default ACL. Objects that are not directories inherit the default ACL of the parent directory as their access ACL only.

The permissions of inherited access ACLs are further modified by the mode parameter that each system call creating file system object has. The mode parameter contains nine permission bits that stand for the permissions of the owner, group, and other class permissions. The effective permissions of each class are set to the intersection of the permissions defined for this class in the ACL and specified in the mode parameter. If the parent directory has no default ACL, the permissions of the new file are determined as defined in POSIX.1. The effective permissions are set

iomega
an EMC company

to the permissions defined in the mode parameter, minus the permissions set in the current umask. The umask has no effect if a default ACL exists.

SHARE-LEVEL USER ACCESS CONTROL

When a share is created on the StorCenter device, it can either be secured or public. A public share requires no user access control; every user can access it and gain full control using both CIFS and NFS. On the other hand, a secured share allows Windows user access control to be configured by checking the **Allow users to change file level security option as shown in Figure 5.**

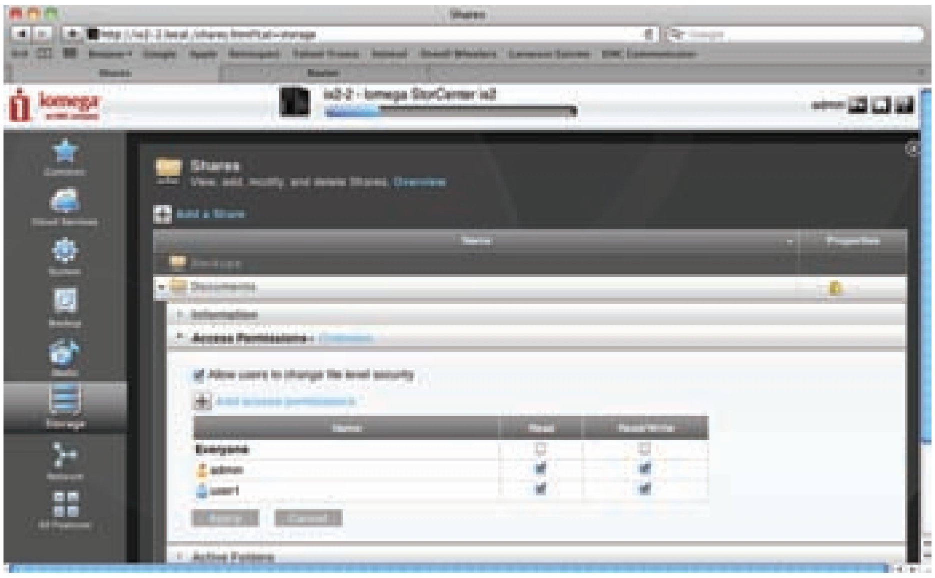
If a share is public, or if the share is secured but does not have the **Allow users to change file level security** option enabled, the share can still be accessed by a Windows host. However, the share, and every directory and file inside the share, do not have the **Security** tab in their **Properties**. Therefore, there is no Windows ACL associated with each object, and Windows permissions cannot be created or modified.

Access control using Iomega StorCenter Manager

This is a three-step process using Iomega StorCenter Manager to secure a folder and configure access control.

1. Enable security on the folder.

Figure 4 Secure a share

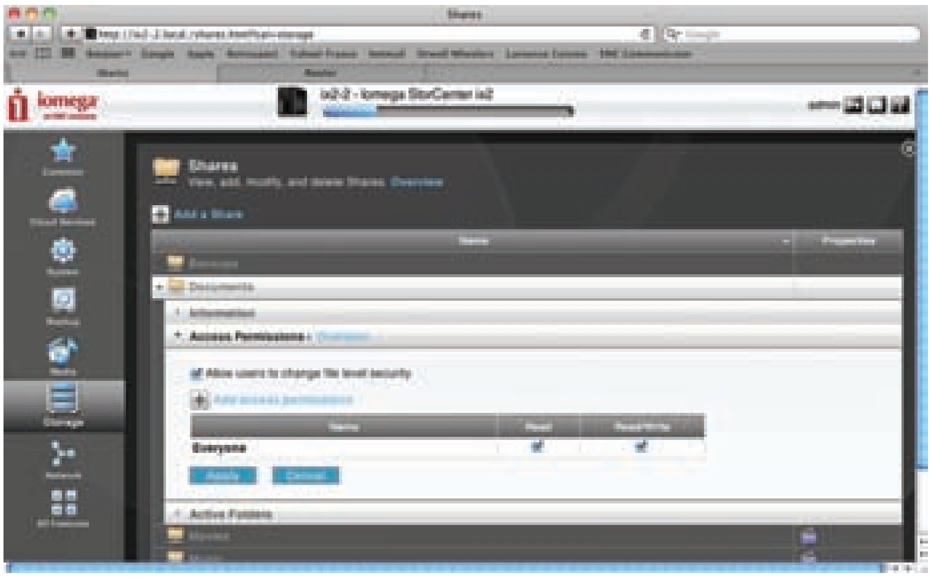


☑ Allow users to change file level security

Name	Read	Read/Write
Everyone	☐	☐
admin	✓	✓
user1	✓	✓

2. Enable Windows user access control on files and directories within the share, and configure user access at the share level.

Figure 5 Enable Windows user access control



By checking the **Allow users to change file level security** option, the following Samba parameters are set:
nt acl support = yes
dos filemode = yes

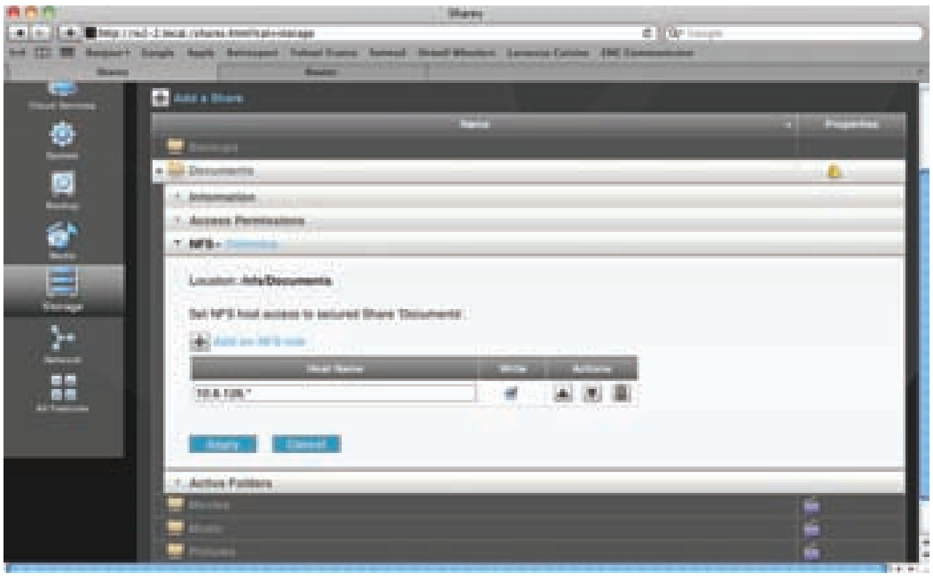
The *nt acl* support parameter controls whether the *smbd* daemon will attempt to map UNIX permissions into Windows ACLs. The dos *filemode* parameter determines if a user having write access to a file can modify the Windows permissions.

Additionally, the configured user access rights are saved in and enforced by Samba.

If the **Allow unsecured read access** option is selected, the share can be accessed by any user without authentication. However, all those users are authorized to only read the folder.

3. Enable NFS secured access to the share and configure host access.

Figure 6 Allow secured NFS access



NFS access to secured shares is normally determined by host access rights. NFS controls who can mount an exported file system based on the host making the mount request, not the user that actually uses the file system. Hosts must be given explicit rights to mount the exported file system. Access control is not possible for users, other than through file and directory permissions.

UNIX USER SQUASHING

When an exported file system is mounted on a host via NFS, any user on the host can access the shared data. To limit the potential security risks, UNIX administrators often allow read-only access, or squash user permissions to a common user and group ID. There are two squashing options when exporting a file system:

- *all_squash*: this option allows the NFS server to treat all client users as anonymous users by mapping all user and group IDs to the anonymous user and group ID.
- *root_squash*: this option allows the NFS server to treat the root user as the anonymous user by mapping its user and group ID to the anonymous user and group ID.

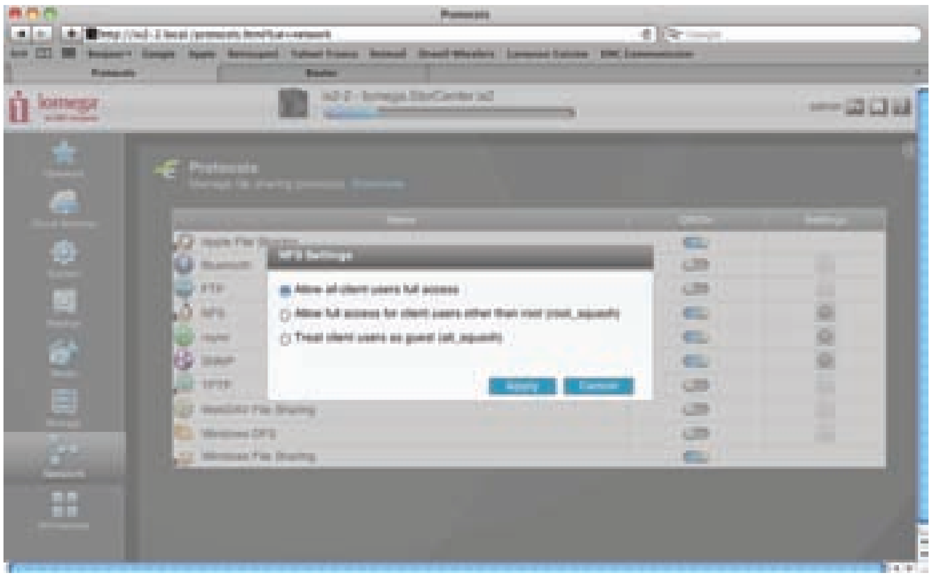
The UNIX anonymous user is typically the *nobody* user in the nobody group. By mapping a super user root or other users into the anonymous user, the NFS server restricts access and operations.

The StorCenter device supports both these user squashing options when the NFS service is enabled on the NAS, as shown in Figure 7.

- Allow all client users full access: no user squashing.
- Allow full access for client users other than root: root_squash.
- Treat client users as guest: all_squash.

Note: NFS settings cannot be modified when the NAS is joined to a domain

Figure 7 Set NFS user squashing



NFS Settings

[✓ Allow all client users full access]
Allow full access for client users other than root (root_squash)
Treat client users as guest (all_squash)

DEFAULT SECURITY SETTINGS

The StorCenter device assigns a default user and group to each share during share creation. The default user is nobody with UID 99, and the default group is users with GID 100. The default permission bits on the share are 0777. When the share is connected on a Windows client, the Windows security settings can be seen in the **Security** tab of the share's **Properties** on a Windows client. The Windows security settings include:

Everyone – Special permissions
nobody (Unix User\nobody) – Special permissions
users (Unix Group\users) – Special permissions

Windows special permissions are customizable sets of permissions. By default, all Windows permissions are allowed for every user on the share.

MODIFICATION OF WINDOWS PERMISSIONS

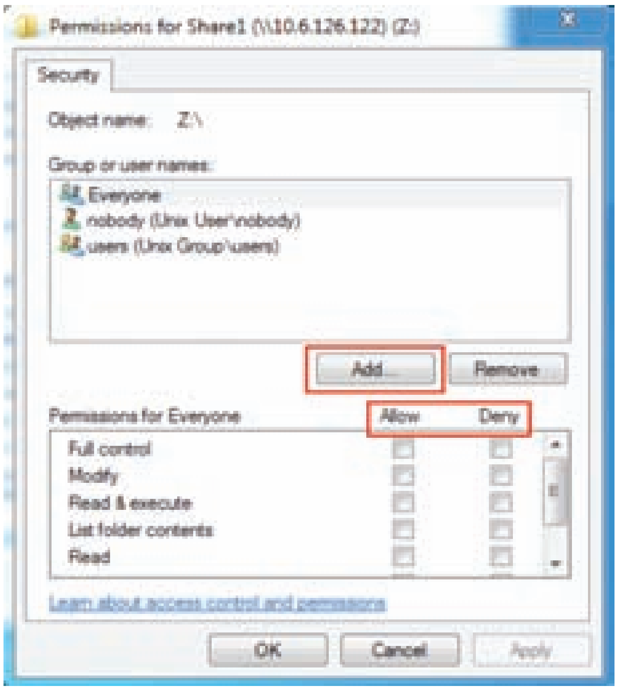
The values of modifying the Windows ACL settings on the share are:

- Customization of Windows permissions that can be automatically inherited by directories and files inside the share.
- Control of more Windows specific permissions that cannot be configured in Iomega StorCenter Manager.

To modify Windows permissions, use the following procedure in Windows Explorer.

1. After the share is connected on the Windows client, right-click the share and open Properties.
2. In the Security tab, click Edit to change permissions.
3. Select desired permissions to allow or deny for a user. Alternatively, to add new permissions, click Add.

Figure 8 Modify Windows permissions


Permissions for Share1 (\\10.6.126.122) (Z:)

Security

Object name: Z:\

Group or user names:

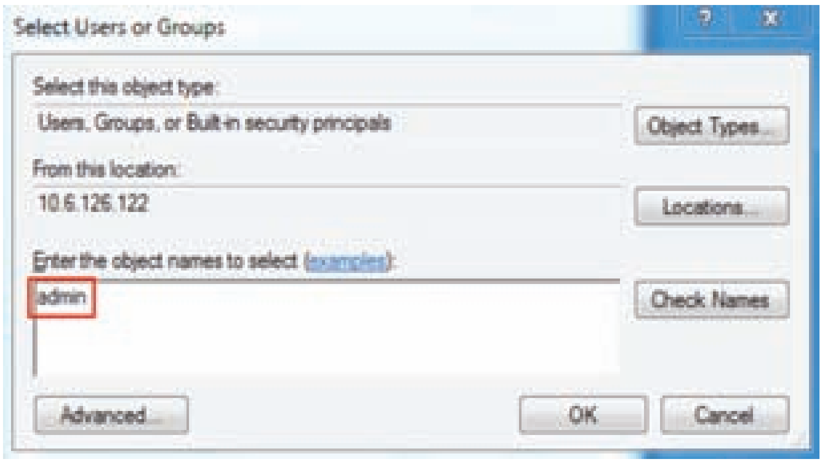
Everyone
nobody (Unix User\nobody)
users (Unix Group\users)

Permissions for Everyone

Permissions for Everyone	Allow	Deny
Full control	☐	☐
Modify	☐	☐
Read & execute	☐	☐
List folder contents	☐	☐
Read	☐	☐

4. Enter the user name for whom the new permissions are created.

Figure 9 Enter user name


Select Users or Groups

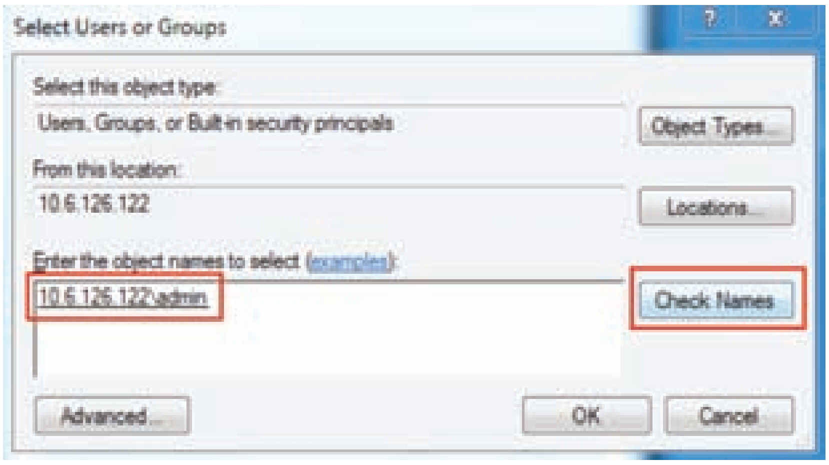
Select this object type:
Users, Groups, or Built-in security principals

From this location:
10.6.126.122

Enter the object names to select (examples):
admin

5. Click Check Names to optionally verify the existence of the user in the workgroup or domain.

Figure 10 Verify user


Select Users or Groups

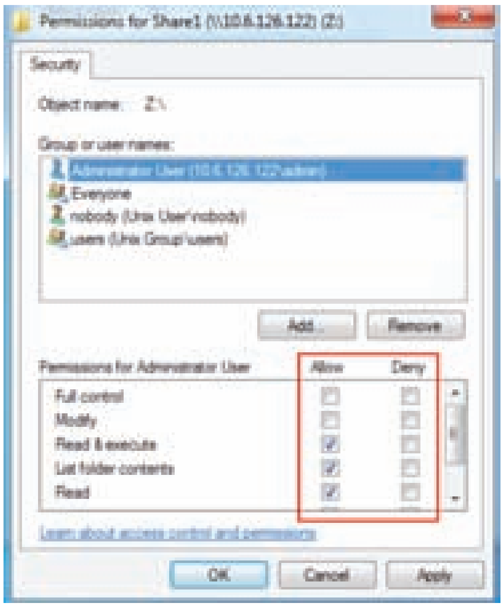
Select this object type:
Users, Groups, or Built-in security principals

From this location:
10.6.126.122

Enter the object names to select (examples):
10.6.126.122\admin

6. Assign appropriate Windows permissions to the user. In this example, the *admin* user is given *Read & execute*, *List* folder contents, and *Read* permissions only.

Figure 11 Assign Windows permissions for the user


Permissions for Share1 (\\10.6.126.122) (Z:)

Security

Object name: Z:\

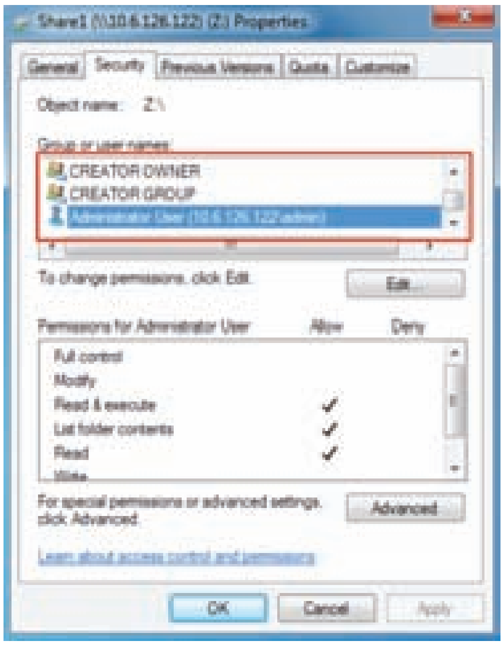
Group or user names:
Administrator User (10.6.126.122\admin)
Everyone
nobody (Unix User\nobody)
users (Unix Group\users)

Permissions for Administrator User	Allow	Deny
Full control	☐	☐
Modify	☐	☐
Read & execute	✓	☐
List folder contents	✓	☐
Read	✓	☐

7. Click **OK** to save the settings. The new set of Windows permissions will be listed. Additionally, if this is the first

time Windows ACL is modified on the share, two new objects named *CREATOR OWNER* and *CREATOR GROUP* with their own Windows permission settings are automatically created. These two new objects are explained in the **Modification of a Windows ACL** section.

Figure 12 Newly added Windows permissions


Share1 (\\10.6.126.122) (Z:) Properties
General
Security
Previous Versions
Quota
Customize
Edit...
Advanced
Learn about access control and permissions
OK
Cancel
Apply

Object name: Z:\

Group or user names:
- CREATOR OWNER
- CREATOR GROUP
- Administrator User (10.6.126.122\admin)

To change permissions, click Edit.

Permissions for Administrator User	Allow	Deny
Full control		
Modify		
Read & execute	✓	
List folder contents	✓	
Read	✓	
Write		

For special permissions or advanced settings, click Advanced.

APPLICATION OF THE MOST RESTRICTIVE SETTINGS

⚠ The default share-level Windows security settings match the default UNIX permissions; both suggest that every CIFS and NFS user has full access to the share. However, these settings alone are not effective. Windows user access is also checked against the access control settings in Iomega StorCenter Manager as shown in Figure 5, which is enforced by Samba rather than Windows ACL. The most restrictive settings apply.

For instances, consider the following scenarios:

- The *admin* user is given read-only access to Share1 in Iomega StorCenter Manager. When you map the share on a Windows client using the *admin* user credential, you will not be able to create or modify files in the share, although the Windows security settings on the share suggest that everyone should have full access.
- The *admin* user is given read-write access to Share1 in Iomega StorCenter Manager. After you map the share on a Windows client using the *admin* user credential, you modify the Windows ACL on the share to grant restrictive permissions to user *admin* as illustrated in Figure 11. As a result, you can browse the folder and read files, but will not be able to create, write or delete files in the share.

Note: Application of the most restrictive settings is the guiding principle in determining a user's effective access rights throughout the entire share. You can grant a user full control over a directory in a share using Windows Explorer, but the user will only have read-only access to the directory if the user is allowed read-only access to the share in Iomega StorCenter Manager. Therefore, if you need to control user access at a more granular level than is possible with Iomega StorCenter Manager, a best practice is to allow read-write access to the share by all users, and then configure more restrictive Windows permissions on a Windows host.

DIRECTORY-LEVEL USER ACCESS CONTROL

You can gain a more granular level of Windows user access control by using native Windows utilities to configure Windows ACLs on individual file system objects within a share. The file system POSIX ACLs on StorCenter device are leveraged to map to Windows ACLs. Again, this is allowed only if the share is secured and has the **Allow users to change file level security** option enabled.

Windows ACL

When a file system object (a directory or a file) is first created inside a share, the object always has the following Windows permissions by default unless the Windows permissions have been previously modified on the share and inheritance is allowed.
Everyone – Special permissions
Administrator User (RTPSOHO8\admin) – Special permissions
users (Unix Group\users) – Special permissions

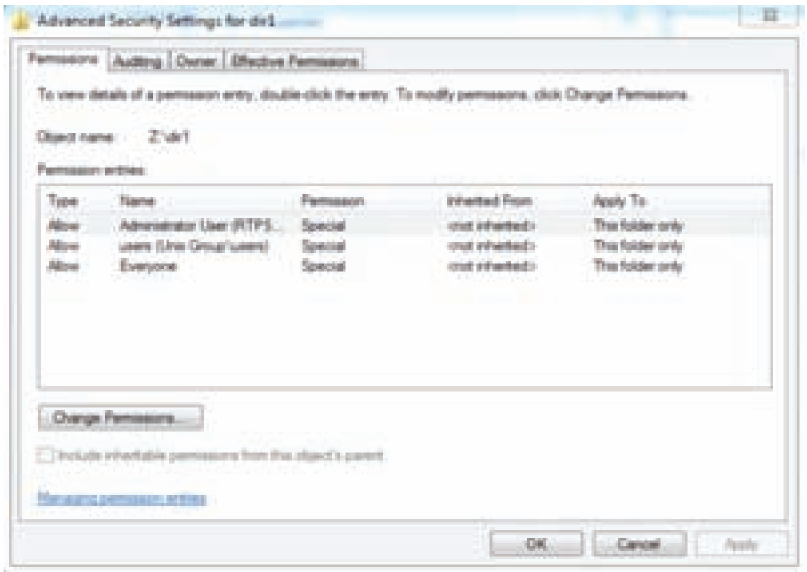
This set of Windows ACL is very similar to that of the share itself. The only difference is that the UNIX nobody user is replaced by the user whose user credentials are used to connect to the share, in this example user admin. When a file system object is first created under a subdirectory inside the share, the object will receive Windows ACL from its parent directory according to rules explained in the Inheritance rules section later in this paper. The Windows ACL for each file system object is automatically translated into POSIX ACL and stored with the object in the file system. On the other hand, if a file system object is created or modified using NFS, the UNIX permissions are set first in POSIX ACL, which is then translated into Windows ACL on the fly when the share is accessed using CIFS. In other words, user permissions are always stored in the file system POSIX ACLs; there is no such entity as Windows ACL in the file system. Translation rules between POSIX ACL and Windows ACL are described in the **Mappings between Windows ACL and UNIX rights** section.

MODIFICATION OF A WINDOWS ACL

Windows ACL can be viewed and modified using native Windows utilities, such as Windows Explorer. The simple procedure described in the Modification of Windows permissions section can be used. Additionally, you can perform more advanced configurations using the same utility.

1. In the **Security** tab of the object **Properties,** click **Advanced** to configure advanced security settings for the object.

Figure 13 Advanced security settings for a file system object



To view details of a permission entry, double-click the entry. To modify permissions, click Change Permissions.

Object name: Z:\dir1

Permission entries:

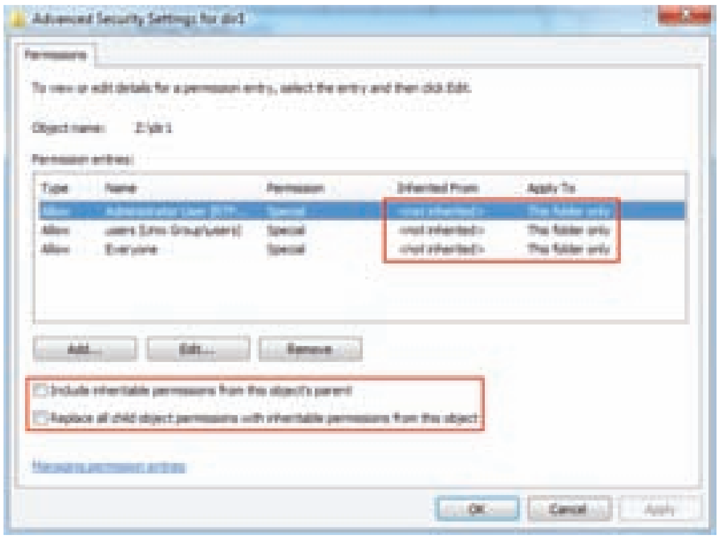
Type	Name	Permission	Inherited From	Apply To
Allow	Administrator User (RTPS...	Special	<not inherited>	This folder only
Allow	users (Unix Group\users)	Special	<not inherited>	This folder only
Allow	Everyone	Special	<not inherited>	This folder only

☐ Include inheritable permissions from this object's parent

2. Click **Change Permissions,** and change the permissions of the selected user.

an EMC company

Figure 14 Change permission entries


Advanced Security Settings for dir1
Permissions
Add...
Edit...
Remove
Managing permission entries
OK
Cancel
Apply

To view or edit details for a permission entry, select the entry and then click Edit.

Object name: Z:\dir1

Permission entries:

Type	Name	Permission	Inherited From	Apply To
Allow	Administrator User (STP...	Special	<not inherited>	This folder only
Allow	users (Unix Group\users)	Special	<not inherited>	This folder only
Allow	Everyone	Special	<not inherited>	This folder only

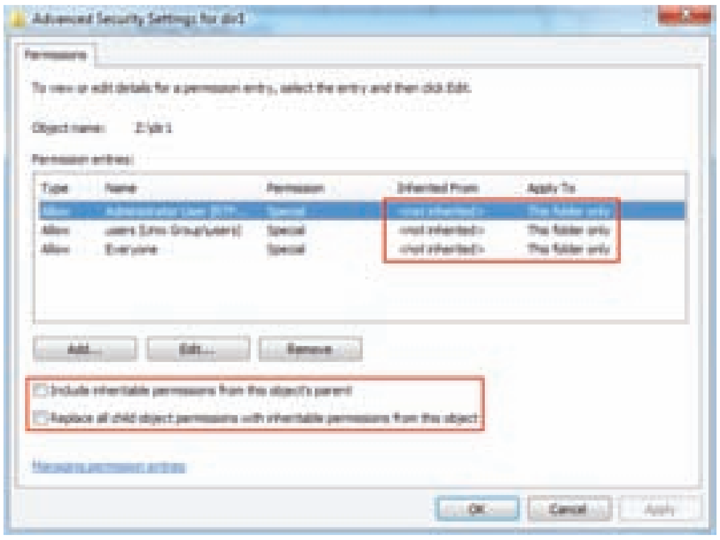
☐ Include inheritable permissions from this object's parent

☐ Replace all child object permissions with inheritable permissions from this object

Notice that the current Windows ACL for the directory does not contain any permission entries inherited from its parent directory. Nor is the ACL inherited by its child objects. These behaviors can be changed by checking **Include inheritable permissions from this object's parent and Replace all child object permissions with inheritable permissions from this object** respectively. The file system will go through all affected objects to update their security entries accordingly, which could be an expensive operation that impacts performance.

3. Customize permission entries by allowing or denying certain rights. You can also determine if inheritance is allowed on the directory and how those permission entries should be inherited.

Figure 15 Change user permission settings


Advanced Security Settings for dir1
Permissions
Add...
Edit...
Remove
Managing permission entries
OK
Cancel
Apply

To view or edit details for a permission entry, select the entry and then click Edit.

Object name: Z:\dir1

Permission entries:

Type	Name	Permission	Inherited From	Apply To
Allow	Administrator User (STP...	Special	<not inherited>	This folder only
Allow	users (Unix Group\users)	Special	<not inherited>	This folder only
Allow	Everyone	Special	<not inherited>	This folder only

☐ Include inheritable permissions from this object's parent

☐ Replace all child object permissions with inheritable permissions from this object

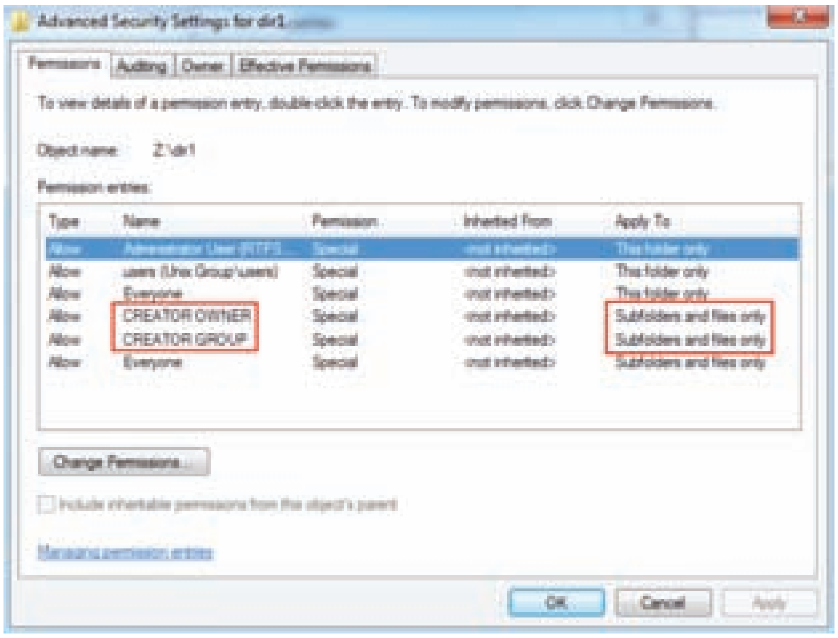
4. Save the modifications. When a directory's Windows ACL is modified and inheritance is enabled for the first time (any option other than **This folder only** in Figure 15), new Windows permission entries are automatically created for the directory. These entries are *CREATOR OWNER* and *CREATOR GROUP*.

CREATOR OWNER is a generic Windows object. It is a placeholder in an inheritable access control entry (ACE). When the ACE is inherited, the system replaces this object's SID with the SID for the object's creator.

CREATOR GROUP is another generic object that is a placeholder in an inheritable ACE. When the ACE is inherited, the system replaces this object's SID with the SID for the primary group of the object's creator. The primary group is used only by the POSIX subsystem.

The *CREATOR OWNER* and *CREATOR GROUP* entries are translated into the POSIX **default ACLs** that control inheritance in the file system. When an object is created inside the directory using NFS, the new object inherits permissions from the **default ACLs**.

Figure 16 Windows objects for inheritance



Advanced Security Settings for dir1

To view details of a permission entry, double-click the entry. To modify permissions, click Change Permissions.

Object name: Z:\dir1

Permission entries:

Type	Name	Permission	Inherited From	Apply To
Allow	Administrator User (RTFS ...	Special	<not inherited>	This folder only
Allow	users (Unix Group\users)	Special	<not inherited>	This folder only
Allow	Everyone	Special	<not inherited>	This folder only
Allow	CREATOR OWNER	Special	<not inherited>	Subfolders and files only
Allow	CREATOR GROUP	Special	<not inherited>	Subfolders and files only
Allow	Everyone	Special	<not inherited>	Subfolders and files only

Permissions
Auditing
Owner
Effective Permissions
Change Permissions...
Include inheritable permissions from this object's parent
Managing permission entries
OK
Cancel
Apply

MULTIPROTOCOL ACCESS CONTROL

IOMEGA ACCESS-CHECKING POLICY

In a multiprotocol environment, the StorCenter device must determine which set of permission attributes on a file or directory to use to grant a user access to a file system object. This process is called user authorization and is controlled through the file system access-checking policy.

When a StorCenter device is in Workgroup mode, access through NFS or CIFS is determined by the protocol which most recently set or modified the permissions on the file system object. Windows ACLs and UNIX permissions are always automatically synchronized. However, the Windows ACLs on the file system object are also compared against the share-level security settings, and the most restrictive settings apply.The same access checking policy applies when the NAS is joined in a Windows domain. However, because in the domain mode every UNIX user is treated as the anonymous user, typically the *nobody* user, UNIX permissions are checked against the *nobody* user and set based on the nobody user. (See section CIFS user ID resolution for more details)

INHERITANCE RULES

When a file system object is created in a secured share using CIFS, the object inherits ACLs from its parent directory if the parent directory allows inheritances. The UNIX permission bits are then translated from the ACLs.

When a file system object is created in a secured share using NFS, the object's UNIX permission bits are determined by the set for the user if the parent directory has no Default ACL. Otherwise, they are determined by the parent directory's Default ACL. The Windows ACLs are then translated from the UNIX permission bits.

MAPPINGS BETWEEN WINDOWS ACL AND UNIX RIGHTS

⚠ There is no perfect mapping between Windows ACL and POSIX ACL. In certain cases, permission translation will be compromised; hence, some security settings will be lost and/or overwritten during the process. For instance, giving a user only the Windows Read attributes permission will result in the UNIX r permission bit being set, which immediately translates back with not only Windows Read attributes, but also List folder/read data, Read extended attributes, and Read permissions settings on the Windows host. You won't be able to assign only the Read attributes permission for an object. While the intended operation may seem like a failure, it is expected behavior due to the fact that security settings are really stored in POSIX ACL and a translation is needed.

Translating a Windows ACL into UNIX rights

When a file system object has its permissions modified using CIFS, the permission entries in the Windows ACL are translated into UNIX permissions as shown in the mappings in Table 4.

	File permissions			Directory permissions		
	R	W	X	R	W	X
Traverse Folder/Execute File			•			•
List Folders				•		
Read Data	•					
Read Attributes	•			•		
Read Extended Attributes	•			•		
Create Files					•	
Write Data		•				
Create Folders					•	
Append Data		•				
Write Attributes		•				
Write Extended Attributes		•				
Delete Subfolder and Files		•				
Delete						
Read Permissions						
Change Permissions						
Take Ownership						

Table 4. Translating a Windows ACL into UNIX rights

Note: there are two special cases with the StorCenter in which a Windows ACL won't translate into UNIX permissions bits. The first case deals with modifying the Windows permissions of the owner of a directory. The modification is always discarded with the UNIX permission bits intact. On the other hand, modification of the UNIX permission bits of the owner of the directory will be effective, and translated accordingly into Windows ACL. The second case deals with modifying the read permission of the owner of a file; the modification is also always discarded.

Translating UNIX rights into a Windows ACL

When a file system object has its permissions modified using NFS, the UNIX permission bits are translated into Windows permissions as shown in the mappings in Table 5.

	R	W	X
Traverse Folder/Execute File			•
List Folders	•		
Read Data	•		
Read Attributes	•		•
Read Extended Attributes	•		
Create Files / Write Data		•	
Create Folders / Append Data		•	
Write Attributes		•	
Write Extended Attributes		•	
Delete Subfolder and Files		•	
Delete			
Read Permissions	•	•	•
Change Permissions		•	
Take Ownership		•	

Table 5. Translating UNIX rights into a Windows ACL

CONCLUSION

The Iomega StorCenter device is a high-performance, easy-to-use, and highly reliable storage device. It is specifically designed to meet the daily storage challenges faced by small- and medium-sized businesses. It is designed to run in a multiprotocol environment so that data can be shared between different platforms and file sharing protocols, including the most popular CIFS and NFS protocols.

A multiprotocol environment presents many challenges to file sharing, the most complicated being user access control. The StorCenter device allows access control in two different ways: share-level access control using Iomega StorCenter Manager, and directory-level access control using native Windows utilities. Windows ACLs and UNIX permissions are synchronized automatically to provide a consistent view of file system object security settings. The combination provides administrators with a simple, flexible, yet protocol-compliant solution for defining and executing the desired level of data security.

iomega®
an EMC company