IBM Storwize V7000 Unified ACL Modes and Migration Options Whitepaper

Version 1.0
5. Feb. 2015

Document Authors:
Achim Christ / European Storage Competence Center
achim.christ@de.ibm.com
Hannes Bellmer / European Storage Competence Center
hannes.bellmer@de.ibm.com

Table of Contents

1. Purpose of this document

IBM Storwize V7000 Unified version 1.5 introduces various new features. Specifically, administrators have greater flexibility in controlling how access permissions are handled by the system with this new version. This allows for optimizing the logical configuration of Storwize V7000 Unified to provide better compatibility with Windows environments.

Additional configuration steps are necessary when upgrading from version 1.4 to version 1.5 of the product. This is required upon code upgrade to version 1.5, even if the new features are not to be leveraged. The exact configuration steps necessary vary with different environments – they particularly depend on the current configuration which is in place prior to the upgrade. The necessary configuration steps and procedures get more complex if the default configuration has been altered prior to the upgrade to version 1.5 already.

This document first explains the newly introduced configuration options, and explains how they influence handling of access permissions. It is shown how different configuration options result in different behavior for clients. Afterwards, different migration scenarios are outlined on a high level, before concrete guidelines are provided for implementation of such scenarios. External dependencies and risks are explained in detail, where appropriate.

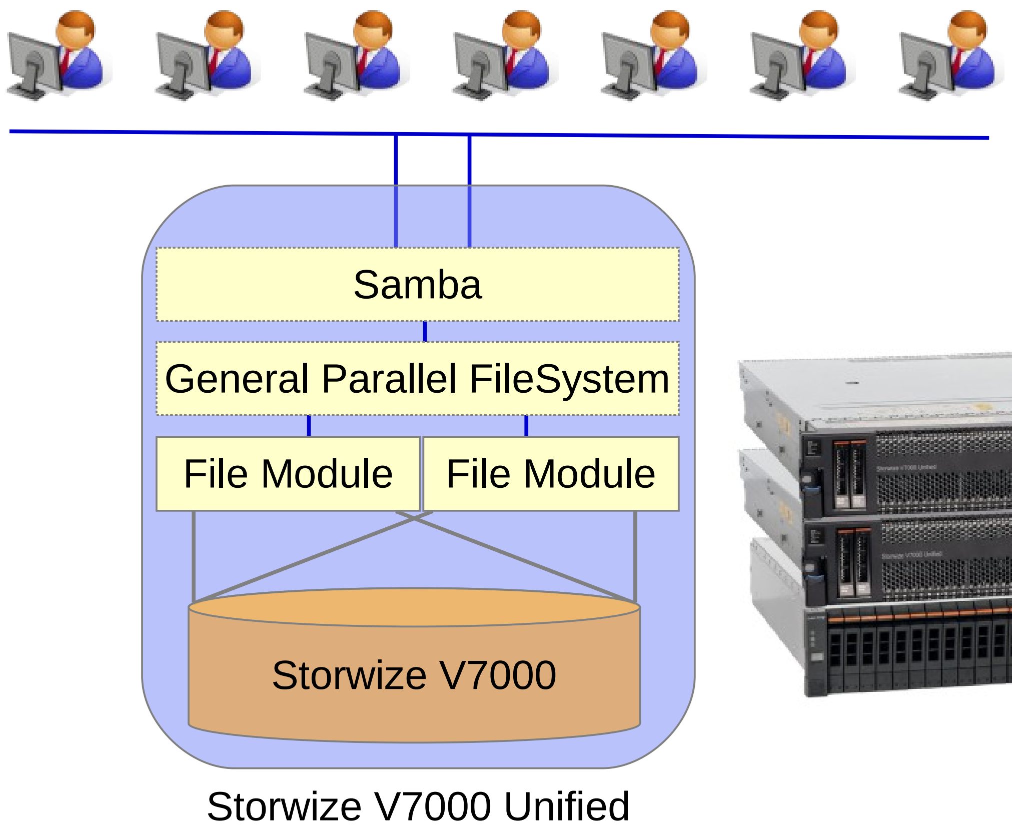
The migration procedures outlined in this paper focus on achieving optimal compatibility in Windows environments. Compatibility with Unix environments, i.e. allowing both, Windows and Unix clients shared access to data, is not covered in detail by the document at hand.

Readers are assumed to have a good understanding of authorization concepts and file system permission management in Windows and Unix environments. Also, they are assumed to have a good understanding of Storwize V7000 Unified, as well as the Software components that make up the Storwize V7000 Unified software stack – in particular this refers to IBM General Parallel FileSystem (GPFS) and Samba. Basic knowledge of GPFS Access Control List (ACL) management is assumed.

2. Overview

This chapter gives a brief introduction into the architecture of IBM Storwize V7000 Unified, lists the Software components that make up the Storwize V7000 Unified software stack, and explains how file access permissions are stored within the product.

2.1. Architecture



Windows clients connect to Samba over an IP network. Samba stores data (files and directories), as well as access permissions for this data in GPFS. Both Samba and GPFS run on File Modules, which provide High Availability clustering. GPFS is a clustered Filesystem, allowing concurrent access to the data. It uses Storwize V7000 Volumes as backend storage (NSDs).

2.2. ACL Modes

IBM Storwize V7000 Unified products leverage the IBM General Parallel FileSystem (GPFS), and use NFSv4 semantics to store Access Control Lists (ACLs) in such Filesystems.

Samba stores file access permissions in such GPFS ACLs. Two different Modes exist, which influence how Samba translates access permissions, being presented to Windows clients, into GPFS ACLs. The following two chapters outline their behavior and show how they influence the mapping of access permissions for Windows clients.

3. ACL Mode Special

The default Samba setting with IBM Storwize V7000 Unified prior to version 1.5 was ACL Mode Special (nfs4:mode = special). This ACL Mode includes certain behavior to provide compatibility with Unix systems connecting via NFS.

In particular, Samba will (implicitly) write **special:owner** and **special:group** Access Control Entries (ACEs) when operating in this Mode. It will do so, if ACEs for the current owner of the object are modified (owning user or owning group).

When reading such Access Control Entries (ACEs), Samba will substitute special:owner entries with the current owning user of the object, and special:group entries with the current owning group of the object.

This behavior allows for a certain level of compatibility with Unix clients. With traditional NFSv3, Unix clients are limited to permissions for owner (owning user), for (owning) group, and for others (everyone).

Achieving compatibility between Windows and Unix environments requires proper planning and thoughtful configuration of access permissions. As this paper focuses on Windows-only environments, all dependencies and implications of shared Windows / Unix access (to the same set of data) are not covered in detail by the document at hand.

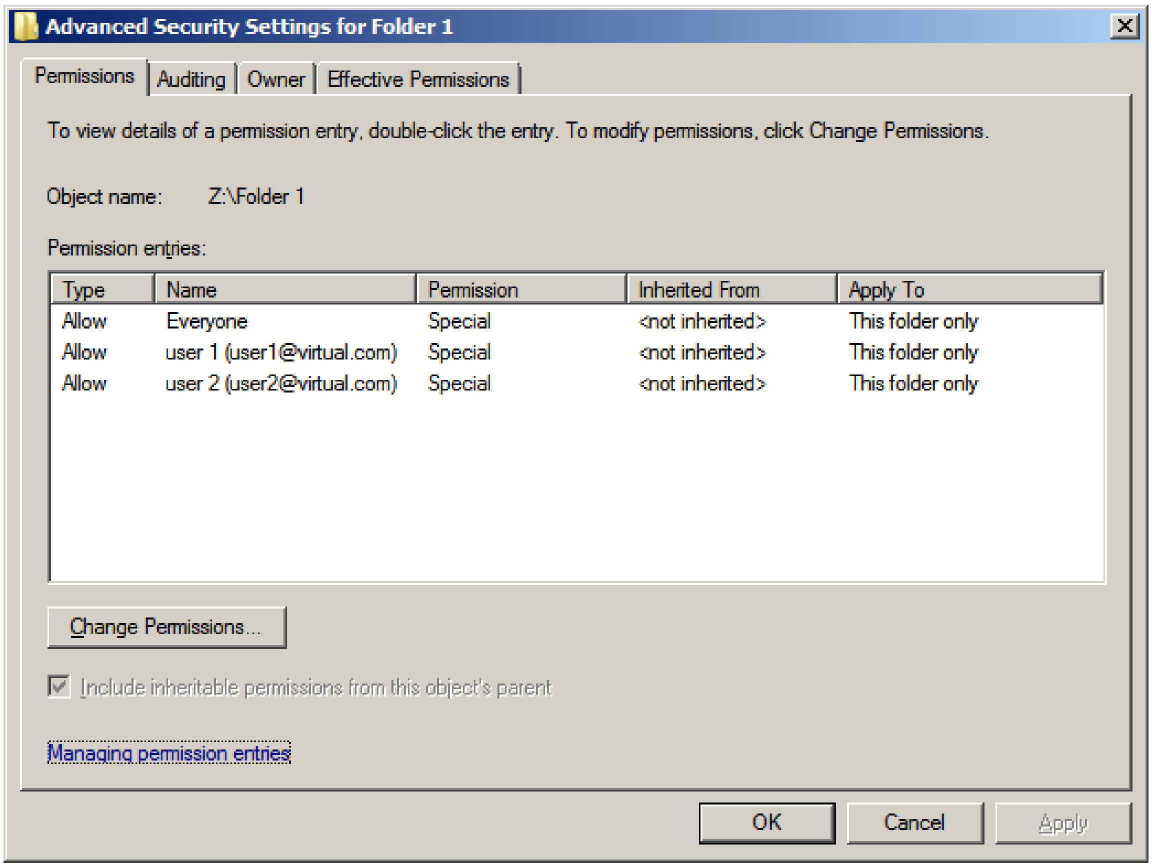
ACL Mode Special does not allow for CREATOR_OWNER and / or CREATOR_GROUP ACEs, due to the fact that such ACEs refer to a special entity (built-in SID) rather than a regular user or group. Refer to the following links for further information on these built-in SIDs:

http://support.microsoft.com/kb/126629
http://networkadminkb.com/KB/a80/creator-owner-explained.aspx

The following examples explain the ACL Mode Special behavior in more detail:

Example 1

Assume that the following ACL is configured for a new object ('Folder 1') from a Windows client:


Advanced Security Settings for Folder 1

Permissions | Auditing | Owner | Effective Permissions

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

Object name: Z:\Folder 1

Permission entries:

Type	Name	Permission	Inherited From	Apply To
Allow	Everyone	Special	<not inherited>	This folder only
Allow	user 1 (user1@virtual.com)	Special	<not inherited>	This folder only
Allow	user 2 (user2@virtual.com)	Special	<not inherited>	This folder only

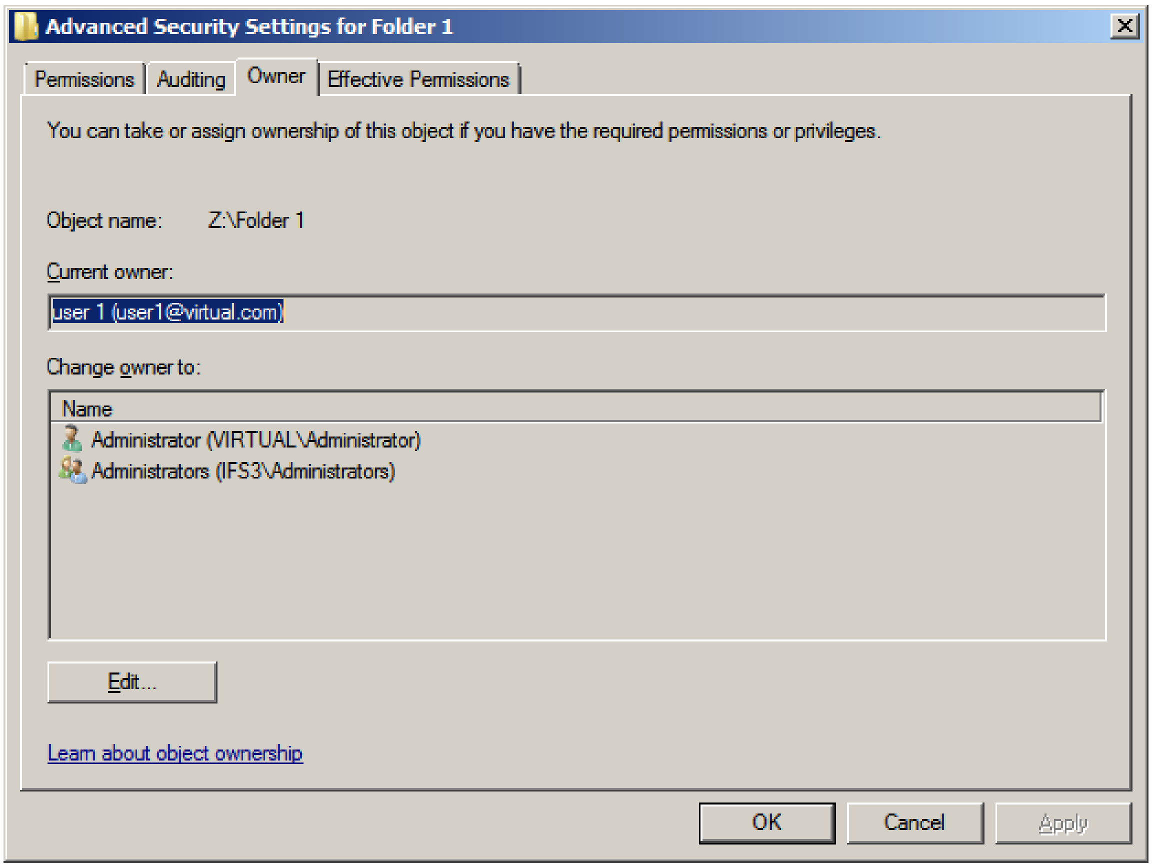
Change Permissions...
Include inheritable permissions from this object's parent
Managing permission entries
OK
Cancel
Apply

One ACE exists for a user ('user 1'), one ACE exists for another user ('user 2'), and one ACE exists for Everyone else. The contents of the ACEs (the actual permissions) are irrelevant for this example, thus will not be discussed any further.

When operating in ACL Mode Special, Samba will write the following to GPFS (in NFSv4 semantics) when writing the above ACL:

```
#NFSv4 ACL
#owner:VIRTUAL\user1
#group:VIRTUAL\domain users
special:everyone@:r-x-:allow
 (X)READ/LIST (-)WRITE/CREATE (-)MKDIR (X)SYNCHRONIZE (X)READ_ACL ...
 (-)DELETE    (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL …

special:owner@:rwxc:allow
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL ...
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL ...

user:VIRTUAL\user2:rwxc:allow
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL ...
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL ...
```

Because 'user 1' is the current owning user of the object, Samba will translate the ACE for 'user 1' into an ACE for '`special:owner@`' - this is shown in the next figure. Because 'user 2' is <u>not</u> the current owning user of the object, Samba will <u>not</u> translate the ACE for 'user 2', but instead write an explicit ACE for that user.


Advanced Security Settings for Folder 1

Permissions | Auditing | Owner | Effective Permissions

You can take or assign ownership of this object if you have the required permissions or privileges.

Object name: Z:\Folder 1

Current owner:

user 1 (user1@virtual.com)

Change owner to:

Name
Administrator (VIRTUAL\Administrator)
Administrators (IFS3\Administrators)

When reading the above ACL, Samba will re-translate '`special:owner@`' into the current owning user of the object. As long as the owning user did not change between writing and reading this ACE, Windows clients are presented with what they expect – the translation is transparent to Samba clients.

The same translation would be applied to the owning group of an object. In the above example, an ACE for 'Domain Users' would be translated into an ACE for '`special:group@`'. The owning group of an object is not (directly) visible to Windows clients – it is initially set to the primary group of the user creating the object, and can not be changed from Windows clients.

3.1. Problems in Windows environments

The explained ACL Mode Special behavior presents Windows clients with what they expect, as long as the owner of an object does not change between writing and reading an ACL. Problems arise from the fact that for a Windows client, the difference between explicit user / group ACEs and special:owner / special:group ACEs is not visible. The invisible translation of such ACEs to explicit ones can cause problems in Windows environments.

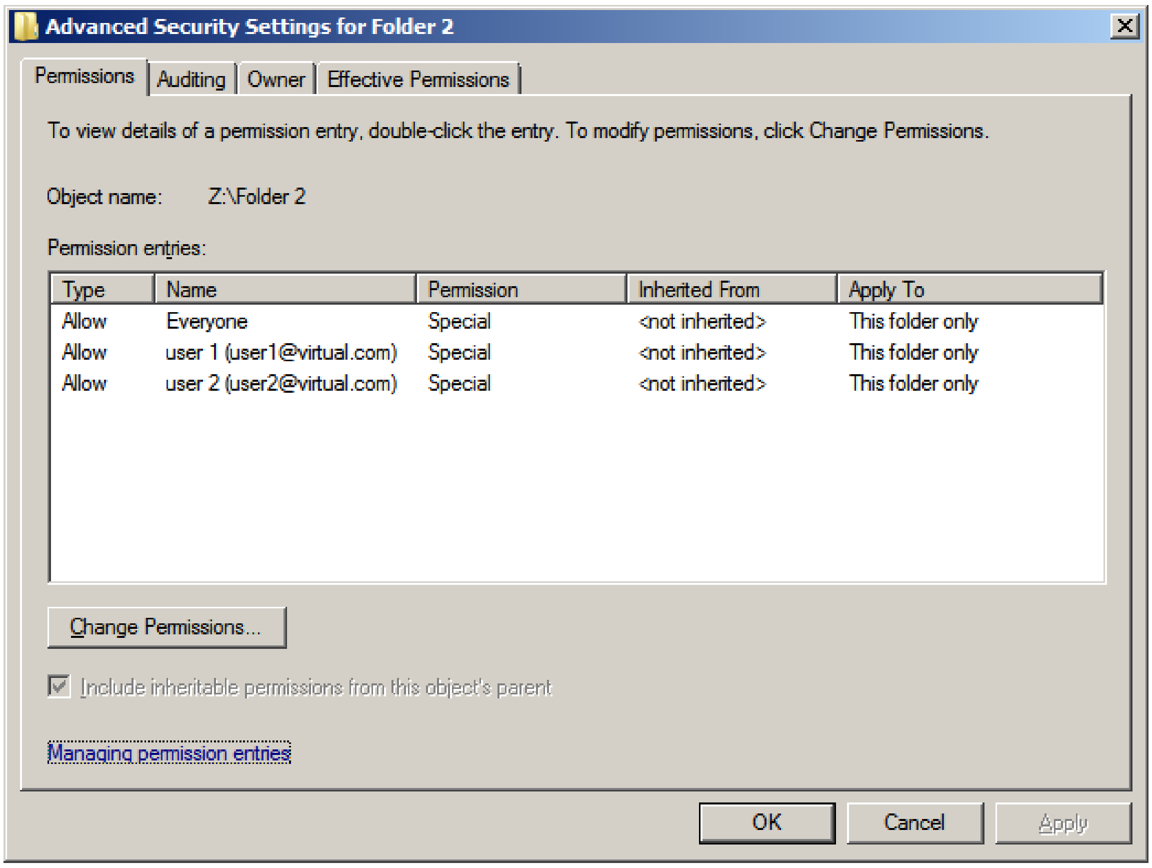
In the above example (Example 1), a Windows client does not notice any difference between the ACE for 'user 1' and the ACE for 'user 2'.

Ownership typically does not play a vital role when managing file permissions in Windows environments – it is not expected to (directly) influence access rights.

The following example outlines how this is different with Samba operating in ACL Mode Special:

Example 2:

Assume that the same ACL as in Example 1 is configured for a new object ('Folder 2') from a Windows client:


Advanced Security Settings for Folder 2

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

Object name: Z:\Folder 2

Permission entries:

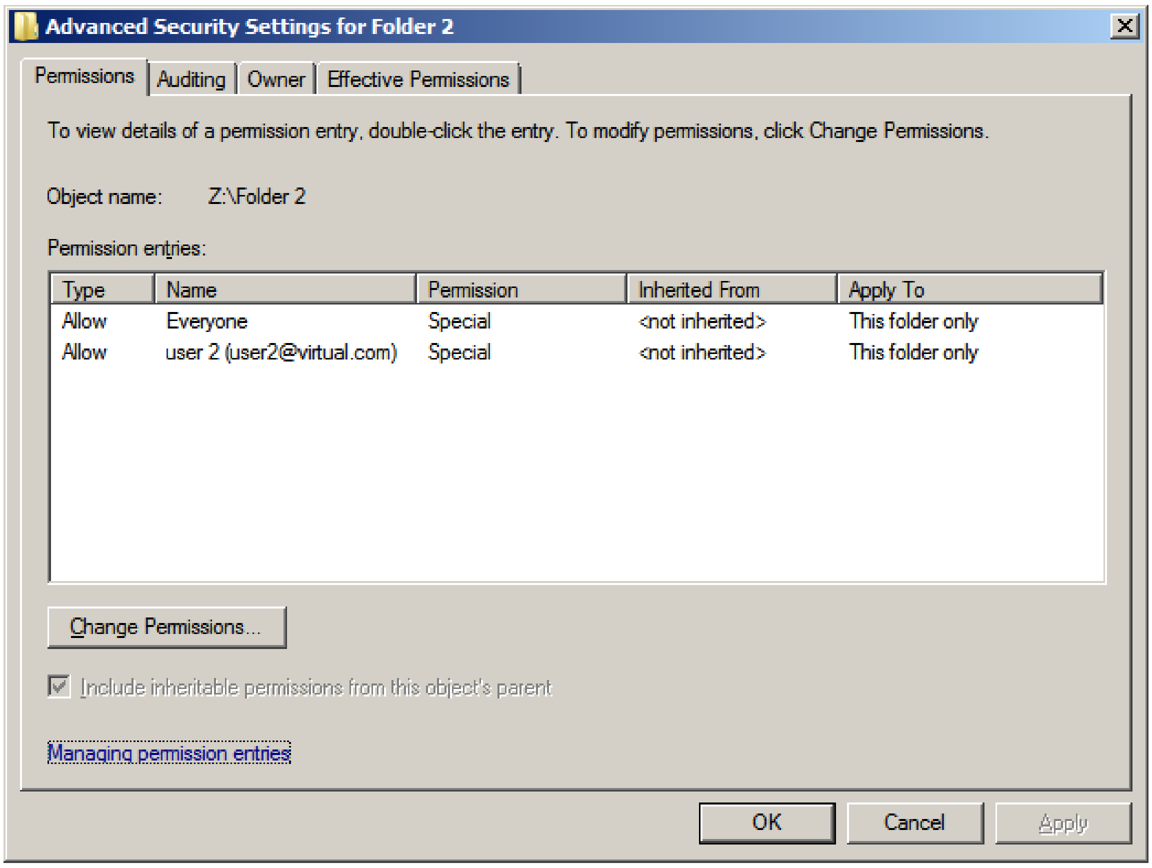
Type	Name	Permission	Inherited From	Apply To
Allow	Everyone	Special	<not inherited>	This folder only
Allow	user 1 (user1@virtual.com)	Special	<not inherited>	This folder only
Allow	user 2 (user2@virtual.com)	Special	<not inherited>	This folder only

☑ Include inheritable permissions from this object's parent

As in the previous example, Samba will write the following ACL to GPFS:

```
#NFSv4 ACL
#owner:VIRTUAL\user1
#group:VIRTUAL\domain users
special:everyone@:r-x-:allow
 (X)READ/LIST (-)WRITE/CREATE (-)MKDIR (X)SYNCHRONIZE (X)READ_ACL ...
 (-)DELETE    (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL ...

special:owner@:rwxc:allow
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL ...
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL ...

user:VIRTUAL\user2:rwxc:allow
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL ...
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL ...
```

Further assume that, for whatever reason, the owner of the object changes – 'user 2' takes ownership of this folder. A Windows client would not expect any influence on access rights from this operation. Instead, Windows administrators would expect to see the exact same ACL as before. But this is not the case:


Advanced Security Settings for Folder 2

Permissions | Auditing | Owner | Effective Permissions

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

Object name: Z:\Folder 2

Permission entries:

Type	Name	Permission	Inherited From	Apply To
Allow	Everyone	Special	<not inherited>	This folder only
Allow	user 2 (user2@virtual.com)	Special	<not inherited>	This folder only

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

The ACE for 'user 1' seems to have vanished – even though the object's permissions have not been altered in any way. In fact, the same ACL is still stored in GPFS:

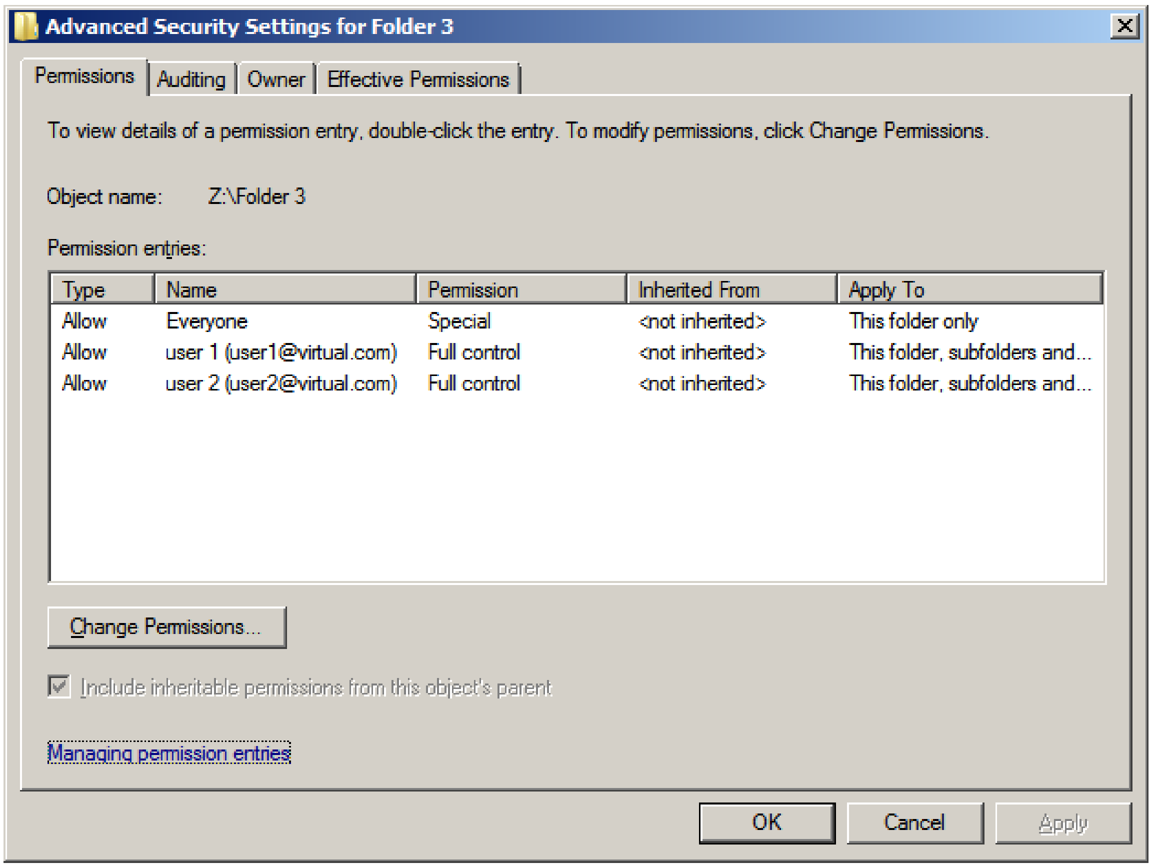
```
#NFSv4 ACL
#owner:VIRTUAL\user2
#group:VIRTUAL\domain users
special:everyone@:r-x-:allow
 (X)READ/LIST (-)WRITE/CREATE (-)MKDIR (X)SYNCHRONIZE (X)READ_ACL ...
 (-)DELETE    (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL ...

special:owner@:rwxc:allow
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL ...
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL ...

user:VIRTUAL\user2:rwxc:allow
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL ...
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL ...
```

The only entry that has changed is the '`owner`' field, showing that 'user 2' is now listed as the owning user of this object. Thus, when reading this ACL, Samba will translate the ACE for '`special:owner@`' into an ACE for the current owning user of the folder, which is now 'user 2'. Where 'user 1', who had access to the object before, has lost all permissions and will not be able to access the folder any longer.

While changing ownership is not a very common use case in Windows environments (as outlined above), the following example shows a more typical scenario in which ACL Mode Special causes unexpected behavior:

Example 3:

Assume that the following ACL is configured for a new object ('Folder 3') from a Windows client:



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

Object name: Z:\Folder 3

Permission entries:

Type	Name	Permission	Inherited From	Apply To
Allow	Everyone	Special	<not inherited>	This folder only
Allow	user 1 (user1@virtual.com)	Full control	<not inherited>	This folder, subfolders and...
Allow	user 2 (user2@virtual.com)	Full control	<not inherited>	This folder, subfolders and...

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

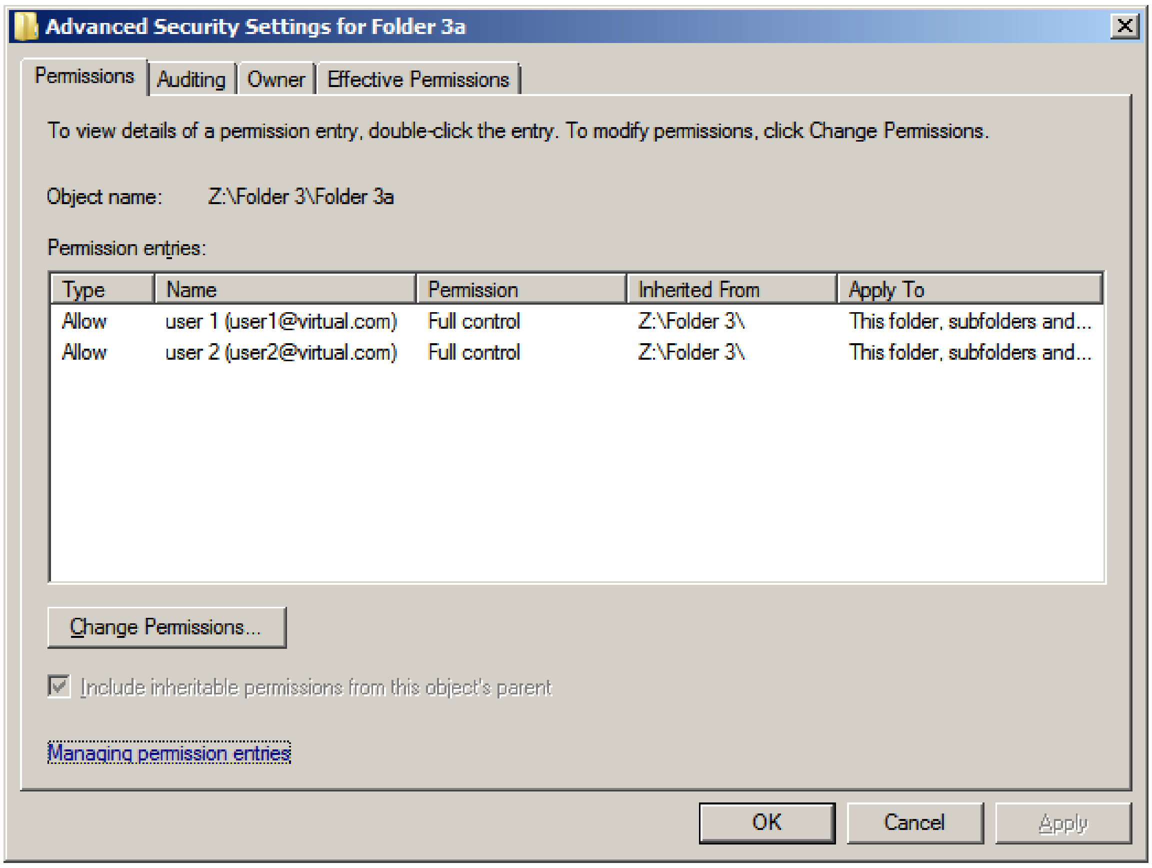
The ACL is similar to the one used in the previous examples, but in this case the ACEs for 'user 1' and 'user 2' are inheriting to child objects ("Apply To This folder, subfolders and files"). Thus, a Windows client would expect to see those two ACEs on all child folders created within 'Folder 3'.

But instead, the actual behavior depends on the owner of these child folders. The ownership of the child folders, in turn, depends on the user who creates them – this is unexpected to Windows users.

Assume that 'Folder 3' (the parent folder) was created by 'user 1'. Thus, 'user 1' is the owning user of that folder, and the ACE for 'user 1' will be translated into a '`special:owner@`' ACE:

```
#NFSv4 ACL
#owner:VIRTUAL\user1
#group:VIRTUAL\domain users
special:everyone@:r-x-:allow
 (X)READ/LIST (-)WRITE/CREATE (-)MKDIR (X)SYNCHRONIZE (X)READ_ACL ...
 (-)DELETE    (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL ...

special:owner@:rwxc:allow:FileInherit:DirInherit
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL ...
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL ...

user:VIRTUAL\user2:rwxc:allow:FileInherit:DirInherit
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL ...
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL ...
```

Assume that 'user 1' creates a child folder named 'Folder 3a' inside 'Folder 3':


Advanced Security Settings for Folder 3a

Permissions | Auditing | Owner | Effective Permissions

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

Object name: Z:\Folder 3\Folder 3a

Permission entries:

Type	Name	Permission	Inherited From	Apply To
Allow	user 1 (user1@virtual.com)	Full control	Z:\Folder 3\	This folder, subfolders and...
Allow	user 2 (user2@virtual.com)	Full control	Z:\Folder 3\	This folder, subfolders and...

Change Permissions...

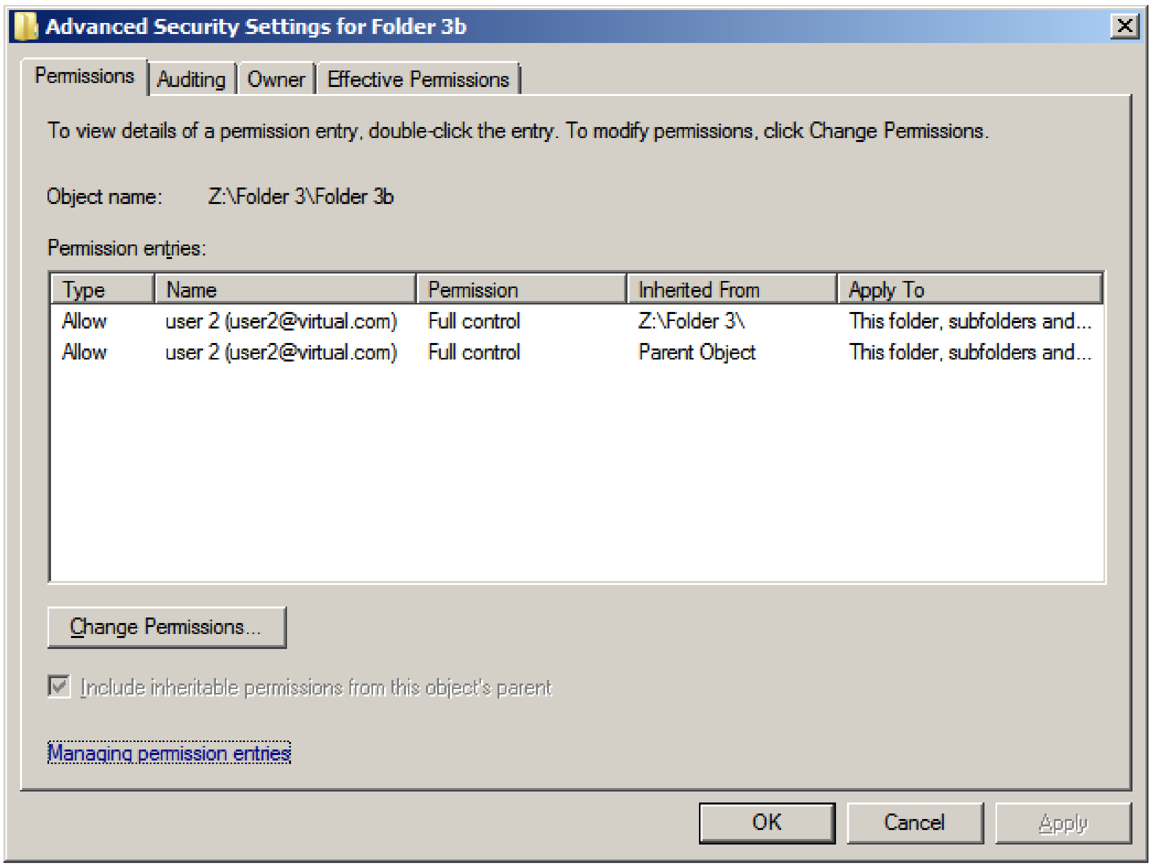
☑ Include inheritable permissions from this object's parent

Managing permission entries

OK | Cancel | Apply

'Folder 3a', which is a child of 'Folder 3', has two ACEs: one for 'user 1', and one for 'user 2'. Both ACEs are inherited from the parent – this behavior is exactly what a Windows client would expect.

Further assume that 'user 2' creates a child folder named 'Folder 3b' inside 'Folder 3':



Advanced Security Settings for Folder 3b

Permissions | Auditing | Owner | Effective Permissions

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

Object name: Z:\Folder 3\Folder 3b

Permission entries:

Type	Name	Permission	Inherited From	Apply To
Allow	user 2 (user2@virtual.com)	Full control	Z:\Folder 3\	This folder, subfolders and...
Allow	user 2 (user2@virtual.com)	Full control	Parent Object	This folder, subfolders and...

Change Permissions...

☑ Include inheritable permissions from this object's parent

Managing permission entries

OK Cancel Apply

'Folder 3b' also has two ACEs, but both refer to 'user 2' (in fact, they will be merged into a single ACE, depending on the representation). The ACE for 'user 1', though present in the parent folder, seems to have vanished – even though the object's permissions have not been altered in any way. 'user 1', who has access to the parent folder, has lost all permissions to the child folder and will not be able to access it.

```
#NFSv4 ACL
#owner:VIRTUAL\user2
#group:VIRTUAL\domain users
special:owner@:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL ...
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL ...

user:VIRTUAL\user2:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL ...
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL ...
```

Similar to the previous example (Example 2), the only entry that has changed between parent folder and child folder is the '`owner`' field. Thus, when reading the (inherited) ACE for '`special:owner@`', Samba will translate the ACE into one referring to the current owning user of the folder, which is 'user 2'.

3.2. Conclusion

The above examples demonstrate that with ACL Mode Special, user ACEs potentially depend on the owning user of an object. In many cases this is unexpected to Windows clients.

Though not shown in the examples, a similar potential dependency exists between group ACEs and the owning group of an object. In many cases the confusion is even larger in this case, because the owning group of an object is not (directly) visible to Windows clients. Thus, the behavior shown comes as a surprise to many Windows administrators, and it can be a challenging task to identify and correct the root cause.

4. ACL Mode Simple

An alternative to ACL Mode Special (nfs4:mode = special) explained in the previous chapter is ACL Mode Simple (nfs4:mode = simple). This ACL Mode does not include the afore-mentioned behavior to provide compatibility with Unix systems connecting via NFS. Instead, ACL Mode Simple provides more consistent behavior to Windows systems connecting via Samba. Thus, it is often preferred for Windows-only environments.

In particular, Samba will not write special:owner or special:group ACEs implicitly when operating in this Mode. Instead, it will write explicit entries for each user and group. Likewise, when reading such (inheriting) ACEs, Samba will neither substitute special:owner entries with the current owning user of the object, nor replace special:group entries with the current owning group of the object.

There is a fundamental difference between inheriting and non-inheriting ACEs in this regard: Non-inheriting special:owner / special:group entries are still replaced with the current owner, to retain a minimal level of compatibility with Unix clients connecting via NFS. It is important to understand that such non-inheriting special:owner / special:group ACEs will not be written by Samba when operating in ACL Mode Simple. Nevertheless, non-inheriting special:owner / special:group ACEs, written by Unix clients connecting via NFS, are read and interpreted properly by Samba.

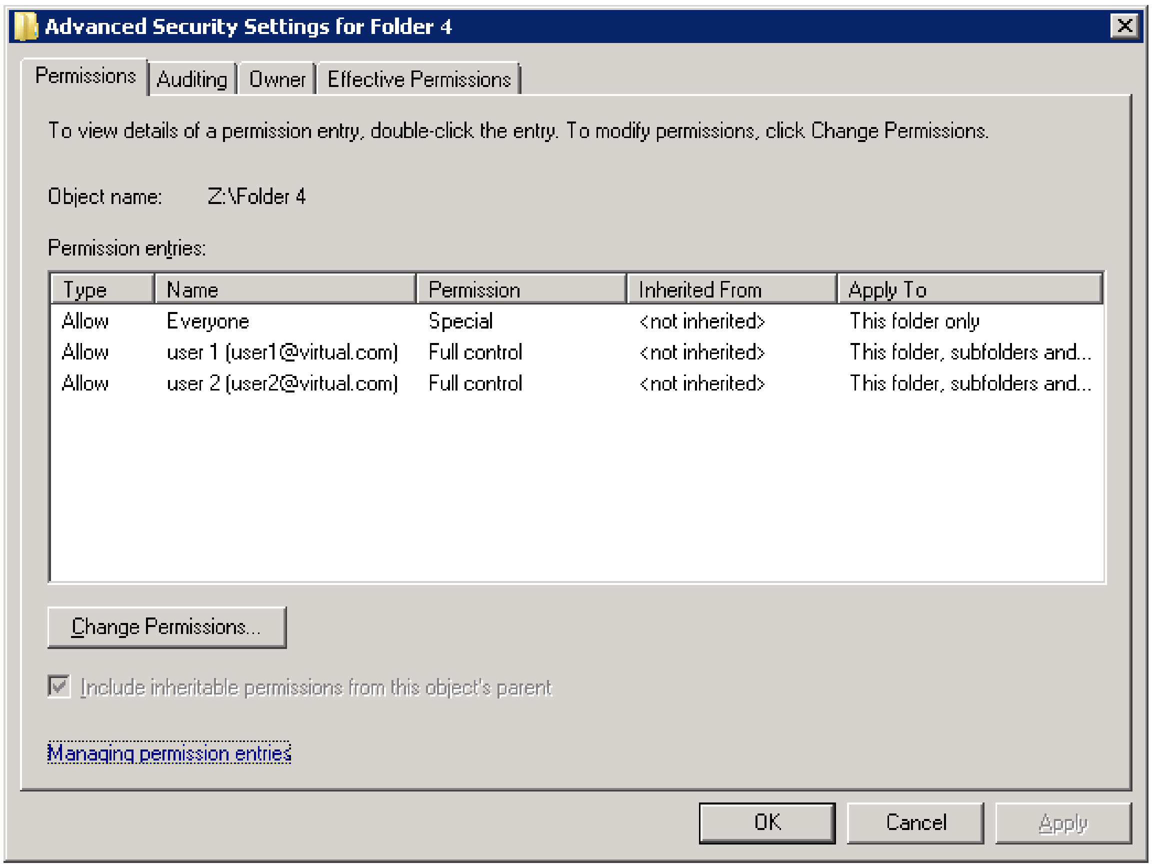
The major disadvantage of ACL Mode Simple results from the fact that explicit user / group ACEs can not be translated into Unix-style permission bits. Hence, Unix clients connecting via NFSv3 are typically not granted any access permissions to files written by Windows clients connecting via Samba. This drastically reduces the options available for administrators to achieve compatibility between Windows and Unix clients, making it a challenging task to achieve shared access to data for both.

When configured for ACL Mode Simple, Samba does not use GPFS special:owner and special:group ACEs for regular operations. This allows for storing CREATOR_OWNER and / or CREATOR_GROUP ACEs by mapping them to (no longer used) inheriting GPFS special:owner and / or special:group entries. As opposed to ACL Mode Special, ACL Mode Simple does allow for usage of CREATOR_OWNER and / or CREATOR_GROUP ACEs from Windows clients.

The following example explains the ACL Mode Simple behavior in more detail:

Example 4:

Assume that the same ACL as in Example 3 from Chapter 3.1 “*ACL Mode Special – Problems in Windows environments*”, page 7, is configured for a new object ('Folder 4'):


Advanced Security Settings for Folder 4

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

Object name: Z:\Folder 4

Permission entries:

Type	Name	Permission	Inherited From	Apply To
Allow	Everyone	Special	<not inherited>	This folder only
Allow	user 1 (user1@virtual.com)	Full control	<not inherited>	This folder, subfolders and...
Allow	user 2 (user2@virtual.com)	Full control	<not inherited>	This folder, subfolders and...

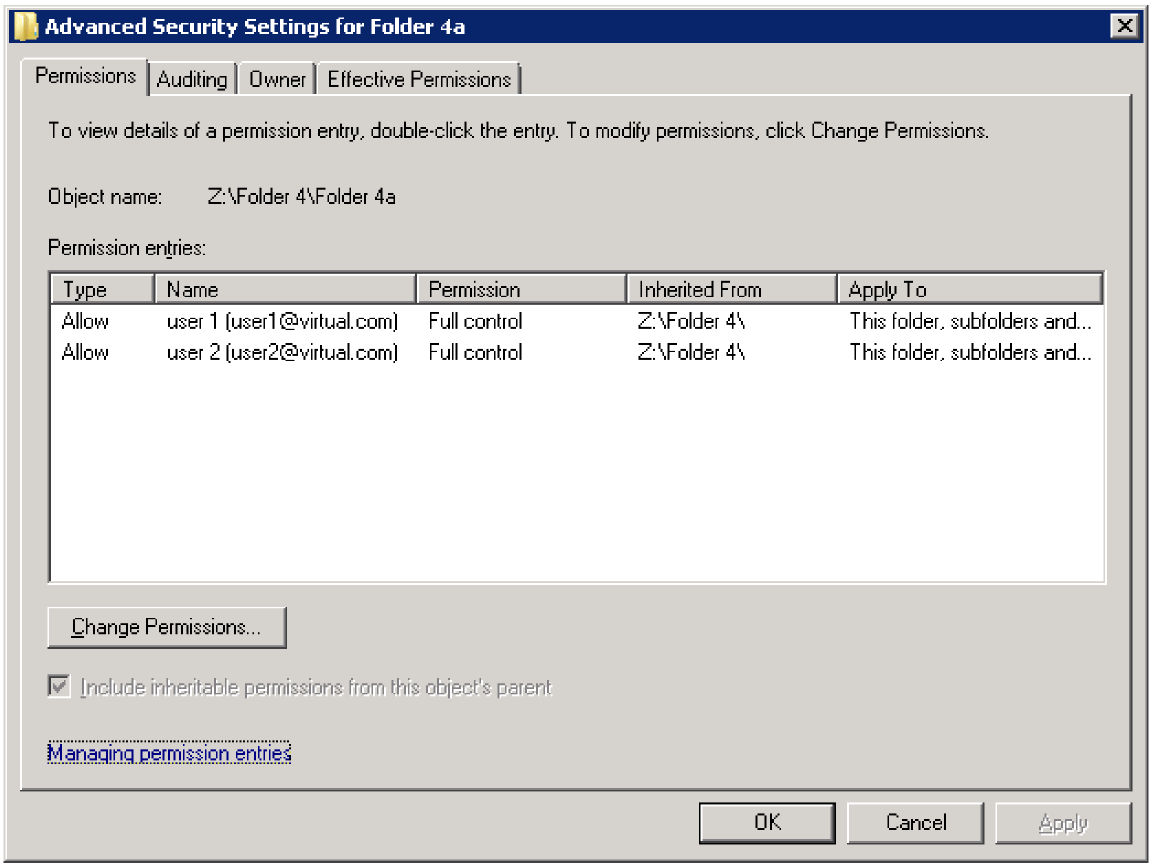
[✓] Include inheritable permissions from this object's parent

On a share which is set to ACL Mode Simple, this will result in the following ACL being written to GPFS:

```
#NFSv4 ACL
#owner:VIRTUAL\user1
#group:VIRTUAL\domain users
special:everyone@:r-x-:allow
 (X)READ/LIST  (-)WRITE/CREATE  (-)MKDIR  (X)SYNCHRONIZE  (X)READ_ACL  ...
 (-)DELETE     (-)DELETE_CHILD  (-)CHOWN  (X)EXEC/SEARCH  (-)WRITE_ACL  ...

user:VIRTUAL\user1:rwxc:allow:FileInherit:DirInherit
 (X)READ/LIST  (X)WRITE/CREATE  (X)MKDIR  (X)SYNCHRONIZE  (X)READ_ACL  ...
 (X)DELETE     (X)DELETE_CHILD  (X)CHOWN  (X)EXEC/SEARCH  (X)WRITE_ACL  ...

user:VIRTUAL\user2:rwxc:allow:FileInherit:DirInherit
 (X)READ/LIST  (X)WRITE/CREATE  (X)MKDIR  (X)SYNCHRONIZE  (X)READ_ACL  ...
 (X)DELETE     (X)DELETE_CHILD  (X)CHOWN  (X)EXEC/SEARCH  (X)WRITE_ACL  ...
```

When comparing this with Example 3 from Chapter 3.1 “*ACL Mode Special – Problems in Windows environments*”, page 7, it becomes obvious that Samba behaves fundamentally different when operating in ACL Mode Simple. Explicit ACL entries are written to GPFS for all ACEs – regardless whether or not the ACE refers to the current owner of the object. Even though 'user 1' is the current owning user of 'Folder 4', an explicit ACE is written for 'user 1'. Similarly, another explicit ACE is written for 'user 2'.

Assume that 'user 1' creates a child folder named 'Folder 4a' inside 'Folder 4', and 'user 2' creates a child folder named 'Folder 4b' inside 'Folder 4'. When Samba operates in ACL Mode Simple, both child folders will have identical ACLs – regardless of the owner of the child object, and whether or not this differs from the owner of the parent object:


Advanced Security Settings for Folder 4a

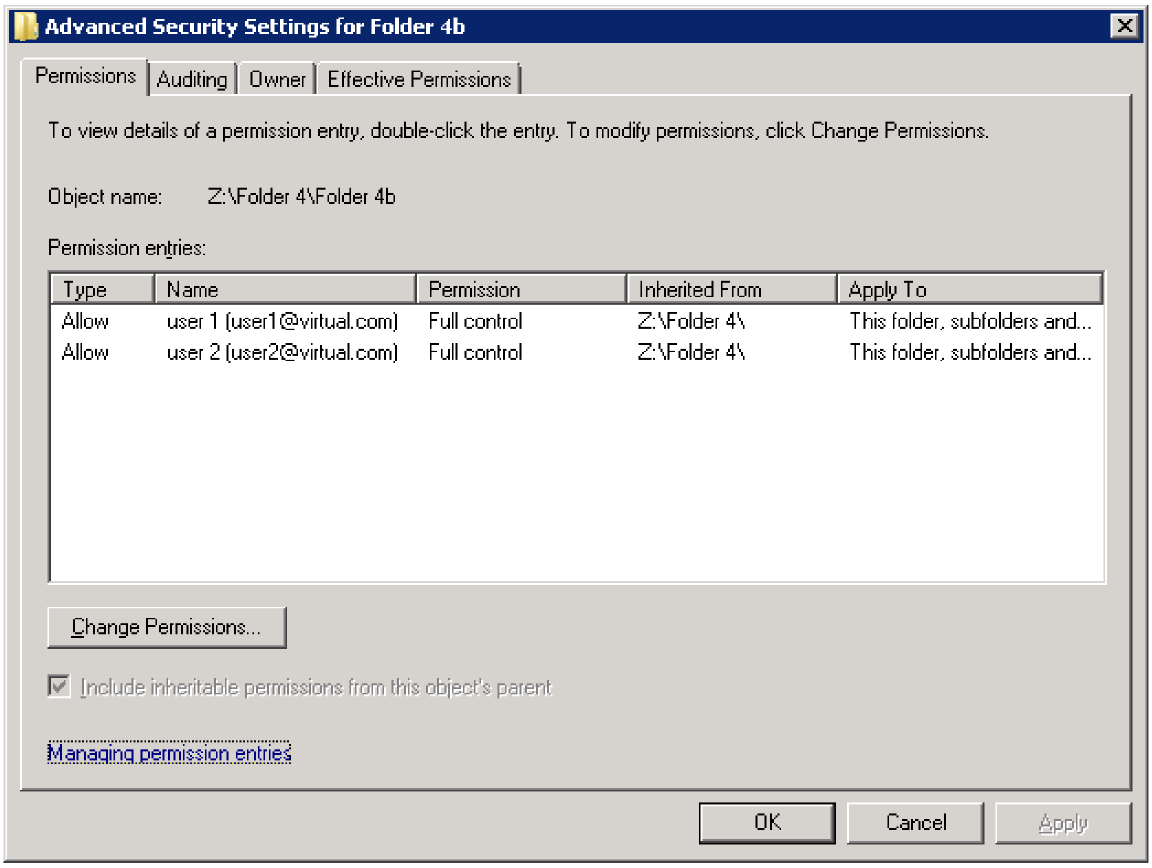
To view details of a permission entry, double-click the entry. To modify permissions, click Change Permissions.

Object name: Z:\Folder 4\Folder 4a

Permission entries:

Type	Name	Permission	Inherited From	Apply To
Allow	user 1 (user1@virtual.com)	Full control	Z:\Folder 4\	This folder, subfolders and...
Allow	user 2 (user2@virtual.com)	Full control	Z:\Folder 4\	This folder, subfolders and...

[✓] Include inheritable permissions from this object's parent


Advanced Security Settings for Folder 4b

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

Object name: Z:\Folder 4\Folder 4b

Permission entries:

Type	Name	Permission	Inherited From	Apply To
Allow	user 1 (user1@virtual.com)	Full control	Z:\Folder 4\	This folder, subfolders and...
Allow	user 2 (user2@virtual.com)	Full control	Z:\Folder 4\	This folder, subfolders and...

[✓] Include inheritable permissions from this object's parent

This behavior is consistent with what Windows administrators would expect, as in Windows environments the owner of an object does not directly affect access permissions.

5. Migration procedure

As outlined in the previous chapters, the semantics of ACL Mode Simple provide significant benefits over ACL Mode Special in Windows-only environments. Thus, ACL Mode Simple is often preferred over ACL Mode Special. Since IBM Storwize V7000 Unified versions prior to 1.5 included a default configuration which applied the ACL Mode Special semantics to all data, a significant amount of administrators would prefer switching the ACL Mode settings of existing Storwize V7000 Unified installations.

To benefit from the new settings on existing data it is mandatory to migrate / re-write existing GPFS ACLs according to the new format to avoid ACL inconsistency.

The following chapter describes how this can be achieved.

<u>Important</u>

IBM is assuming all Storwize V7000 Unified users to be using ACL Mode Simple consistently, starting with release 1.5.

All of IBM's internal testing is based on ACL Mode Simple configurations.

However, IBM is not aware of any issues related to mixed configurations of ACL Mode Special and ACL Mode Simple.

IBM does support (though IBM does not recommend) running long-term with mixed mode configurations of ACL Mode Special and ACL Mode Simple.

5.1. High-level migration procedure

This chapter gives a high-level overview of the migration options available, before the following chapter gives more specific guidance on how to implement such procedures.

a) General considerations

With Storwize V7000 Unified, the ACL Mode can either be set for individual shares, or globally for the entire system. Likewise, the ACL Mode can be changed by applying share-specific (local) settings, or by applying a global setting. The global ACL Mode applies to all shares which do not have a share-specific (local) setting – the local (share-specific) setting supersedes the global setting. Hence, the global ACL Mode applies to all new shares, unless they receive a share-specific (local) setting post creation.

In a default Storwize V7000 Unified configuration only the global setting is present. Thus, by default, there are no share-specific (local) settings, which means the global setting applies to all shares consistently.

Re-configuring Samba to change the ACL Mode of a share is possible online, without any downtime. The changed ACL Mode will be applied to new Samba sessions, only. Existing Samba sessions will continue to use the previous ACL Mode, which was in effect when the session was initially established. Thus, all clients need to disconnect and re-connect for the newly set ACL Mode to be applied.

Samba clients which have not yet re-connected <u>must not</u> modify any ACLs – otherwise invalid /

inconsistent settings may get applied.

When changing the ACL Mode for a share which already contains data, the external representation of certain ACEs for this data will change – which is undesired in most cases. When switching from ACL Mode Special to ACL Mode Simple, CREATOR_OWNER and / or CREATOR_GROUP ACEs may show up on data which did not have such entries previously.

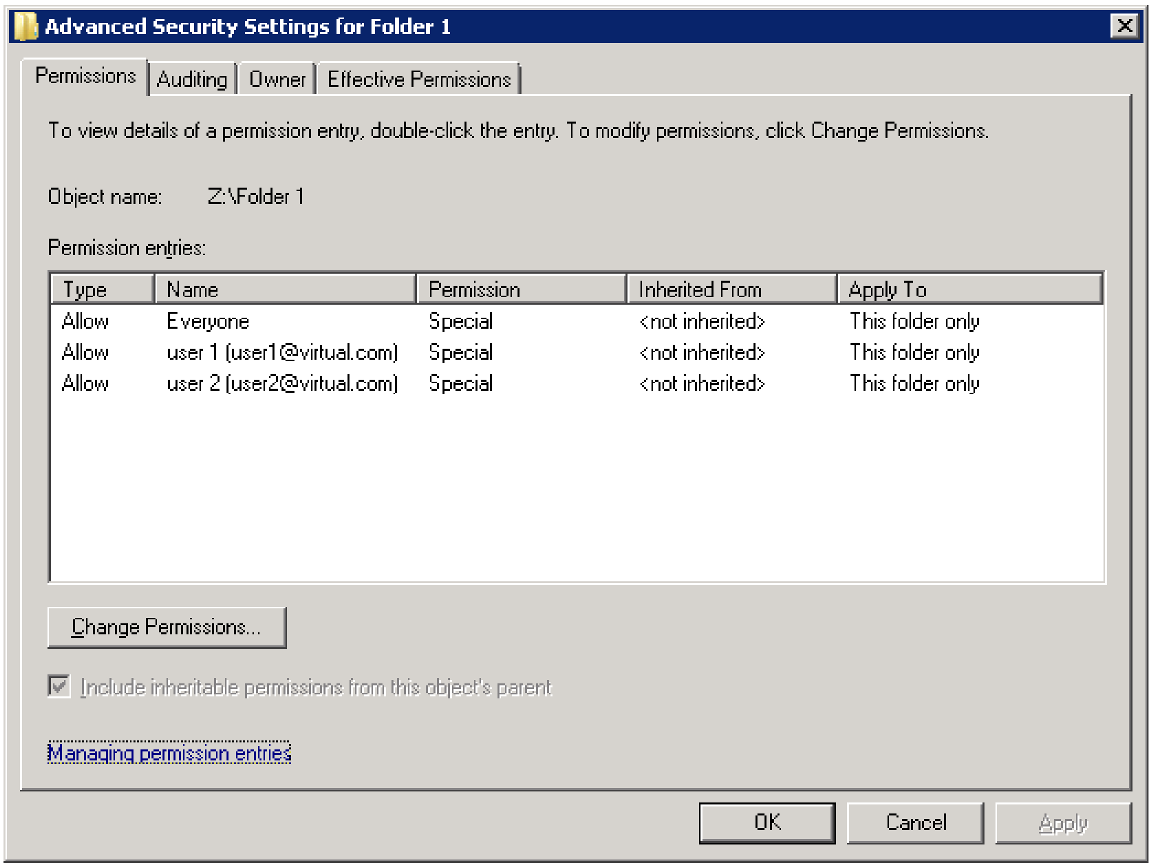
The following example demonstrates this undesired misinterpretation of existing ACLs:

Example 5:

The share which was used in the previous examples (Example 1 – 3 from Chapter 3 "*ACL Mode Special*", page 5) was set to ACL Mode Special when initially writing the data. Assume that the Samba configuration is altered with regards to the ACL Mode of this share – it is now being switched to ACL Mode Simple, without modifying anything else.

The data contained in the share is not altered in any way – the ACLs written to GPFS are not being modified when switching the ACL Mode of the share.

After Windows clients re-connect, 'Folder 1' from Example 1 from Chapter 3 "*ACL Mode Special*", page 5, is now being presented with the following ACL:



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

Object name: Z:\Folder 1

Permission entries:

Type	Name	Permission	Inherited From	Apply To
Allow	Everyone	Special	<not inherited>	This folder only
Allow	user 1 (user1@virtual.com)	Special	<not inherited>	This folder only
Allow	user 2 (user2@virtual.com)	Special	<not inherited>	This folder only

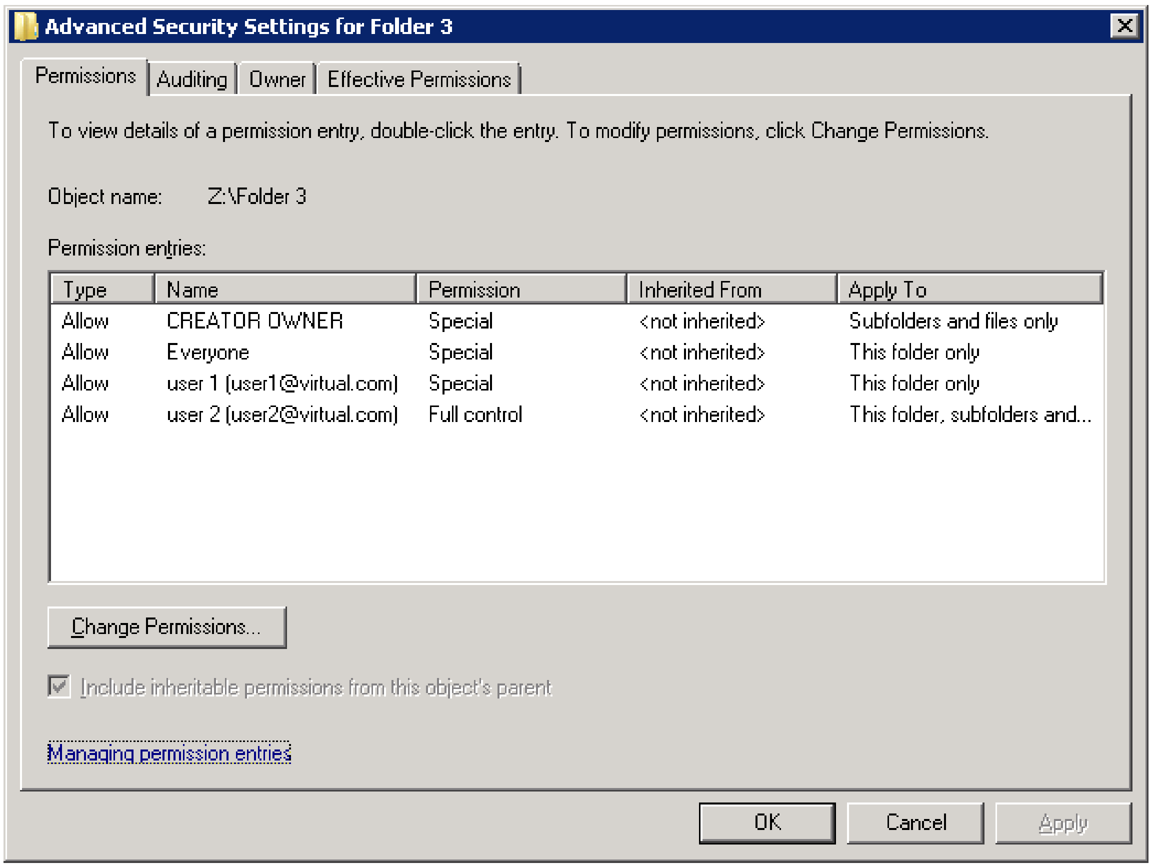
☑ Include inheritable permissions from this object's parent

Managing permission entries

The non-inheriting '`special:owner@`' ACEs are still being translated into the current owner of the object – this is the same behavior with ACL Mode Simple and with ACL Mode Special.

Inheriting '`special:owner@`' ACEs, however, are interpreted differently by ACL Mode Simple.

After switching the share to this ACL Mode, 'Folder 3' from Example 3 from Chapter 3.1 “*ACL Mode Special – Problems in Windows environments*“, page 7, is now being presented with the following ACL:


Advanced Security Settings for Folder 3

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

Object name: Z:\Folder 3

Permission entries:

Type	Name	Permission	Inherited From	Apply To
Allow	CREATOR OWNER	Special	<not inherited>	Subfolders and files only
Allow	Everyone	Special	<not inherited>	This folder only
Allow	user 1 (user1@virtual.com)	Special	<not inherited>	This folder only
Allow	user 2 (user2@virtual.com)	Full control	<not inherited>	This folder, subfolders and...

☑ Include inheritable permissions from this object's parent

Inheriting '`special:owner@`' ACEs will now be interpreted as CREATOR_OWNER ACEs. Though not shown in the example, inheriting '`special:group@`' ACEs will be interpreted as CREATOR_GROUP ACEs in a similar fashion.

To avoid this undesired behavior, special migration steps are required as outlined in the following chapters. In particular, existing GPFS ACLs need to be re-written when changing the ACL Mode for a share, so that the interpretation of existing data is not altered by the operation.

The migration options available vary with different software (firmware) versions of Storwize V7000 Unified.

b) Migration options prior to version 1.5

Prior to version 1.5, the default ACL Mode setting for Storwize V7000 Unified is **Special** (nfs4:mode = special). It is possible to change the Samba configuration regarding the ACL Mode with version 1.4, but this requires root access to the File Modules. An approved RPQ / Score Request is required for a supported configuration.

Ideally, the appropriate ACL Mode is configured for a share before any data is written to this share. When changing the ACL Mode for a share which already contains data, the external representation of all (inheriting) special:owner and special:group ACEs for this data will change. Such ACEs are interpreted differently by the different ACL Modes, so without any modifications to the data in GPFS (i.e. ACL re-write) the representation and behavior for Windows clients accessing this data will change. This is undesired in most cases.

All newly written data, and all newly written ACLs will behave consistently as explained before.

To avoid this (undesired) change in the external representation of special:owner and special:group ACEs for existing data, it is necessary to migrate / copy the data using e.g. Windows clients. There exist multiple possible scenarios, which all require downtime – the data will be inaccessible during the migration:

- Apply default ACLs to all existing data. This option is probably undesired if the system contains large amounts of data already, especially if complex access permissions exist for this data.
- While the share is still in ACL Mode Special: Migrate the data from the share to some external storage using a Windows client and e.g. robocopy. This will preserve the ACL representation for Windows clients. Change the share from ACL Mode Special to ACL Mode Simple. Migrate the data back from the external storage, again using a Windows client and e.g. robocopy. As the actual data is already present in the share, only ACLs need to be copied in this last migration step.
- Instead of using external storage for the migration, it is possible to create a new (empty) share on Storwize V7000 Unified operating in ACL Mode Simple. The data needs to be migrated using a Windows client and e.g. robocopy nonetheless, in order to preserve the ACL representation for Windows clients. The original share can then either be deleted while the new share takes it's place (which is probably the preferred option), or the old share can be changed to ACL Mode Simple before the data is being migrated back.

After all shares have been migrated following the procedures above, the global ACL Mode setting of Storwize V7000 Unified can be changed from ACL Mode Special to ACL Mode Simple. The global setting will then be applied to all newly created shares. All share-specific (local) settings can now be deleted to ensure that the global setting gets applied to all shares consistently, in the same fashion as in the default configuration.

c) Migration options with version 1.5 and above

The migration procedure for software (firmware) version 1.5 and above is documented in the IBM Knowledge Center:

> http://www-01.ibm.com/support/knowledgecenter/ST5Q4U_1.5.0/com.ibm.storwize.v7000.unified.150.doc/upgradingacl.html

Storwize V7000 Unified version 1.5 introduces various changes with regards to ACL Mode settings, and in particular introduces tools to migrate existing data without requiring external clients (using e.g. robocopy):

- For new installations, the default ACL Mode is **Simple (**nfs4:mode = simple).
- For upgrades of existing installations, an **Intermediate** ACL Mode is introduced

(nfs4:readmode = special, nfs4:mode = simplenocreator).

- The software (firmware) upgrade procedure from version 1.4 to version 1.5 will automatically modify the ACL Mode setting for existing shares, if certain conditions are met.
- A set of CLI commands was introduced to support migrations (including ACL re-write), which can (optionally) be performed online.

The default ACL Mode for new installations is Simple starting with version 1.5. It is still possible to modify the ACL Mode for individual shares, or globally – one may choose to revert back to the old behavior (ACL Mode Special semantics) by changing the corresponding settings, in case the described compatibility with Unix clients connecting via NFS is desired. Ideally, the desired ACL Mode is set before any data is written to a share.

When upgrading existing installations from version 1.4 to version 1.5, the ACL Mode settings are automatically modified. Existing shares will be switched to ACL Mode Intermediate, under certain conditions (as outlined below). When operating in this Intermediate Mode, Samba will interpret existing data as it would do in ACL Mode Special, but it will write new data as it would do in ACL Mode Simple. Thus, both existing data (written in ACL Mode Special) and new data (written in ACL Mode Intermediate) will be interpreted identically. This allows for modification of the GPFS ACLs, without any impact on Samba clients connected. ACLs can be re-written transparently for data in shares which are set to ACL Mode Intermediate.

If this automatic modification of the ACL Mode setting of existing shares is undesired, the global ACL Mode can be set to Simple prior to applying the software (firmware) upgrade. If the global ACL Mode is set to Simple, then the version 1.5 upgrade process will not make any modifications to the Samba configuration with regards to the ACL Mode settings. Individual shares can still be migrated from ACL Mode Special to ACL Mode Simple (or vice versa) after the upgrade has completed, but this approach requires additional manual steps.

Before changing Samba's global ACL Mode, it is important to understand the implications of this operation. As mentioned earlier, the global ACL Mode gets applied to all shares which do not have a share-specific (local) setting. Assuming that the effective behavior of all existing shares should not be altered by the operation, all existing shares require a share-specific (local) setting to override the global ACL Mode. Before changing the global ACL Mode, it is recommended to apply a local setting to all shares without an existing local setting (using the same mode as the current global setting). Thus, when switching the global ACL Mode setting, all existing shares will continue to operate with the same behavior as before...

It is <u>not</u> recommended to use shares set to ACL Mode Intermediate for an extended period of time. The purpose of this ACL Mode is to allow for transparent modifications of the ACLs – once this modification (re-writing of GPFS ACLs) has completed, the ACL Mode of the respective shares needs to be set to Simple. This will finalize the ACL migration. All clients need to disconnect and re-connect at this stage for the newly set ACL Mode to be applied (for all new sessions).

As mentioned earlier, Samba clients which have not yet re-connected <u>must not</u> modify any ACLs – otherwise invalid / inconsistent settings may get applied.

Only after the migration has been finalized, the ACL Mode has been set to Simple, and all connected clients have re-connected, CREATOR_OWNER and / or CREATOR_GROUP ACEs can be used on a specific share. Such ACEs can not be stored in ACL Mode Special or Intermediate, which is why the above mentioned finalization of the ACL migration needs to be carried out after all GPFS ACLs have been re-written.

Risks

Storwize V7000 Unified version 1.5 introduces a set of CLI commands to transparently modify (re-write) GPFS ACLs for existing data. This migration can be performed online, while Samba clients remain connected and continue to access the data which is being modified.

The migration tools, which can be controlled using CLI commands, are designed to migrate existing data (written in ACL Mode Special) to be interpreted correctly by ACL Mode Simple. For this, the migration tools will replace all special:owner ACEs with explicit entries for the current owning user of the object, and all special:group ACEs with explicit entries for the current owning group of the object.

All data which was written in ACL Mode Special needs to be migrated using the tools above, before it is being accessed by Samba clients operating in ACL Mode Simple. Data written in ACL Mode Special which is <u>not</u> being migrated in this way, will show extraneous / invalid CREATOR_OWNER and / or CREATOR_GROUP ACEs when Samba is switched to ACL Mode Simple.

On the other hand, data which was written in ACL Mode Simple may contain legitimate CREATOR_OWNER and / or CREATOR_GROUP ACEs. Running the mentioned migration tools on data written in ACL Mode Simple will delete such legitimate CREATOR_OWNER and / or CREATOR_GROUP ACEs.

The migration operation can not be reverted, because legitimate CREATOR_OWNER / CREATOR_GROUP ACEs can not be distinguished from extraneous / invalid entries. Thus, it is critically important to understand which share (and which directory) contains data written in ACL Mode Special, and which share contains data written in ACL Mode Simple (potentially containing legitimate CREATOR_OWNER and / or CREATOR_GROUP ACEs). Failure to identify the proper procedure, or application of the wrong procedure for a given set of data, will result in invalid / inconsistent access permissions being applied to the data.

Upgrading environments with Snapshots and / or Backups

If automatic Snapshots or automatic Backups (via TSM or NDMP) are configured for the Filesystem which is being migrated, or the Filesystem is part of an Asynchronous Replication relationship, then it is recommended to wait for these scheduled processes to pick up the changes after modification (re-writing) of the GPFS ACLs.

Ideally, switching the ACL mode for all shares in a given Filesystem is performed after all previous Snapshots and all previous Backups were expired and / or deleted. This is to avoid having to restore data with the previous (pre-update) GPFS ACLs.

Since modification of the GPFS ACLs does not affect files in Snapshots and / or files in Backups, such copies of the data still contain the previous (pre-update) ACLs. If such Snapshot / Backup copies of data have to be restored, it is necessary to re-run the GPFS ACL migration procedures on this data – it is critically important to limit the scope of this run of the migration tools to the files which were being restored with previous (pre-update) ACLs. Failure to do so will result in invalid / inconsistent access permissions being applied to the data.

Upgrading environments with File Copy Services

When migrating systems in an Asynchronous Replication relationship, all systems need to be upgraded to software (firmware) version 1.5 before any further modifications are applied. Only after all systems in the Asynchronous Replication relationship have been upgraded to version 1.5, which will make ACL Mode Intermediate available, modification of the GPFS ACLs can take place transparently, without any impact on connected clients.

In general it should be avoided to run systems with different code levels in an Asynchronous Replication relationship for an extended period of time. In case of a disaster failover users are required to switch from the primary- to the secondary system in the relationship. If this implies switching from one software (firmware) code level to another, this imposes additional risk to the failover process. If the different code levels behave differently in certain aspects, then users are exposed to unforeseen changes, which is undesired in most cases.

However, the above recommendation is not specific to the ACL migration scenarios outlined in this document. After the primary system was upgraded to software (firmware) version 1.5, and before any additional modifications are applied to the GPFS ACLs or Samba configuration, a disaster failover can still be performed to a secondary system running code level 1.4. If the secondary system in a Replication relationship is upgraded before the primary system is upgraded, then the disaster failover capabilities are not negatively affected either.

One needs to ensure that a share is set to ACL Mode Intermediate on all systems in a Replication relationship, before any modification to the GPFS ACLs for data inside this share is applied. However, this modification (re-writing GPFS ACLs) needs to be carried out on the primary system in a Replication relationship, only.

The same is true when migrating systems in a Remote Caching (Active Cloud Engine) configuration. All systems that participate in Remote Cache relations need to be updated to software (firmware) version 1.5, and a share needs to be set to ACL Mode Intermediate on all systems (sites). Modification of the GPFS ACLs needs to be carried out on Cache sites with Writer roles first, before the GPFS ACLs on the Home site are modified.

5.2. Low-level migration procedure

After discussing the general migration procedures and dependencies in the previous chapter, the following chapters give more specific guidance on how to implement such procedures.

a) Migration options prior to version 1.5

Specific instructions for changing the ACL Mode settings with Storwize V7000 Unified version 1.4 are provided as part of the approved RPQ / Score Request mentioned earlier, which needs to be in place for a supported configuration. Note that software (firmware) versions prior to 1.5 do not allow for migrating existing data from ACL Mode Special to ACL Mode Simple semantics directly. Instead, external clients (such as Windows clients running e.g. robocopy) need to be used to copy all data at least once. Thus, it is strongly recommended to configure the appropriate ACL Mode for a share before any data is written to this share.

All available migration options prior to software (firmware) version 1.5 require downtime – the data will be inaccessible during the migration. An upgrade to Storwize V7000 Unified version 1.5 is generally preferred, before the ACL Mode settings of existing shares are altered.

b) Migration options with version 1.5 and above

If File Copy Services are being used (Asynchronous Replication and / or Remote Caching), then all Storwize V7000 Unified systems which share data through these File Copy Services need to be upgraded to software (firmware) version 1.5 before any further steps are performed. This is to ensure that all systems understand the new ACL semantics when Samba configuration settings are changed, or GPFS ACLs are modified.

During the upgrade process to version 1.5, the ACL Mode settings for existing shares will be modified under the following conditions:

If the global ACL Mode is set to Simple, then the upgrade procedure will not make any modifications to the Samba configuration regarding the ACL Mode settings. After the upgrade has completed, individual Filesystems can manually be marked as 'intermediate' to start the upgrade procedure as explained below. This approach may be chosen if only a small subset of the existing data is to be migrated, while the majority of the existing data should remain untouched.

Use the following command to manually mark individual Filesystems as 'intermediate':

```
updatefs <filesystem> --aclmode forceintermediate
```

If the global ACL Mode is set to Special (which was the default prior to version 1.5), then the upgrade procedure will make the following modifications to the Samba configuration regarding the ACL Mode settings:

- The global ACL Mode setting is switched to Simple.
- All shares which are set to ACL Mode Special will be changed to ACL Mode Intermediate. This is true for both, shares with a share-specific (local) setting of Special, as well as shares without a local setting, for which the global ACL Mode setting got applied before the upgrade.
- All shares which are set to ACL Mode Simple will not be modified – they will remain with a share-specific (local) ACL Mode setting of Simple.
- All Filesystems with all shares in these Filesystems set to ACL Mode Intermediate will be marked as '**intermediate**'.
- All Filesystems with all shares in these Filesystems set to ACL Mode Simple will be marked as '**simple'**
- All Filesystems with shares in different ACL Modes, i.e. Filesystems with at least one share set to ACL Mode Simple and at least one share set to ACL Mode Intermediate, will be marked as '**inconsistent'**

After the upgrade has completed, all existing shares will either be set to ACL Mode Intermediate or to ACL Mode Simple. Each share will have a share-specific (local) ACL Mode setting.

In addition to the ACL Mode for shares, Filesystems will now also be marked with a certain Mode. Filesystems will be in one of the following states:

- Current ACL mode is 'intermediate' and requires cacheupdate.

- Current ACL mode is 'simple' and up to date.
- Current ACL mode is inconsistent for filesystem shares.

Use the following command to query the current Mode of a Filesystem:

```
# updatefs <filesystem> --aclmode show
```

The following sections provide detailed instructions on how to migrate Filesystems (and the shares contained therein), based on their current Mode. The appropriate procedure needs to be applied to each Filesystem – unfortunately there is no single process to address all possible scenarios.

All procedures can be performed online, while clients connected continue to operate on the data being migrated. However, clients need to disconnect and re-connect after the ACL Mode was switched to pick up the new Samba configuration settings.

Migrating Filesystems marked as 'intermediate'

Filesystems marked as 'intermediate' only contain shares with an ACL Mode setting of Intermediate. The GPFS ACLs need to be modified for all data in these shares. This modification needs to be carried out while the share is set to Intermediate Mode. After the GPFS ACLs have been modified accordingly (re-written), the Samba configuration can be switched to ACL Mode Simple for all shares in these Filesystems.

1. Verify that the Filesystem is marked as 'intermediate' on all systems which share access to the Filesystem through File Copy Services (Asynchronous Replication and / or Remote Caching):

   ```
   # updatefs <filesystem> --aclmode show
   Current ACL mode is 'intermediate' and requires cacheupdate.
   EFSSG1000I The command completed successfully.
   ```

2. Filesystems in a Remote Caching configuration require an additional step – skip this step if the Filesystem is not shared through Remote Caching (Active Cloud Engine).
   Update the GPFS ACLs for all data in the Filesystem on the Writer Cache site in a Remote Caching Configuration:

   ```
   # updatefs <filesystem> --aclmode updatecaches
   ```

   This could be an iterative approach, as explained below. Repeatedly run this command until no further updates are applied to the Filesystem data.

3. Update the GPFS ACLs for all data in the Filesystem on the primary system in a Replication relationship, or on the Home site in a Remote Caching configuration:

   - To update GPFS ACLs for all data in all shares in the Filesystem:

     ```
     # updatefs <filesystem> --aclmode update
     start processing of /ibm/...
     traversed ... files, processed ... ACL's and changed ....
     Hit 0 issues during this update run.
     Multipe runs are recommended for online updates.
     processing completed.
     EFSSG1000I The command completed successfully.
     ```

- Alternatively, to update individual shares or directories, only:

  ```
  # updatefs <filesystem> --aclmode update --aclpath <path>
  start processing of /ibm/...
  traversed ... files, processed ... ACL's and changed ....
  Hit 0 issues during this update run.
  Multipe runs are recommended for online updates.
  processing completed.
  EFSSG1000I The command completed successfully.
  ```

Performance tests in lab environments have shown that the above command is capable of processing 100 mio. files in roughly 5 Minutes. Of course, the actual performance will depend on the overall workload of the system, among other factors.

4. Repeat the above step until no further updates are applied to the Filesystem data:

   - To update GPFS ACLs for all data in all shares in the Filesystem:

     ```
     # updatefs <filesystem> --aclmode update
     start processing of /ibm/...
     traversed ... files, processed ... ACL's and changed 0.
     Hit 0 issues during this update run.
     Multipe runs are recommended for online updates.
     processing completed.
     EFSSG1000I The command completed successfully.
     ```

   - Alternatively, to update individual shares or directories, only:

     ```
     # updatefs <filesystem> --aclmode update --aclpath <path>
     start processing of /ibm/...
     traversed ... files, processed ... ACL's and changed 0.
     Hit 0 issues during this update run.
     Multipe runs are recommended for online updates.
     processing completed.
     EFSSG1000I The command completed successfully.
     ```

   Specifically note the line that reads “`... and changed 0.`” in the above output. If, during the second run of the above steps, further updates are made to the Filesystem data, then additional runs of the '`updatefs <filesystem> --aclmode update`' command are necessary. Run the command repeatedly until no further updates are applied.

5. If, in the previous steps, the '`--aclpath`' parameter was used to modify individual shares or directories only, then it is recommended to perform an additional run of the '`updatefs <filesystem> --aclmode update`' command without the '`--aclpath`' parameter. This is to ensure that all data in all shares in the Filesystem has been migrated consistently, and to avoid situations where certain directories were accidentally overlooked and skipped.

6. If the Filesystem is configured for automatic Snapshots, for automatic Backups (via TSM or NDMP), or the Filesystem is part of an Asynchronous Replication relationship, then it is recommended to wait for these scheduled tasks to pick up the changes before continuing. Ideally, the following steps are being carried out after all previous Snapshots and all previous Backups were expired and / or deleted, to avoid restoring data with the previous (pre-update) GPFS ACLs.

7. Use the following command to finalize migration of the Filesystem:

   ```
   # updatefs <filesystem> --aclmode finalize
   EFSSG1000I The command completed successfully.
   ```

   This will remove the share-specific (local) ACL Mode setting for all shares in the Filesystem. After this step, the global Samba ACL Mode gets applied to all shares in the Filesystem, which is consistent with the default Storwize V7000 Unified configuration.

   Perform the finalization of the migration on the secondary system in a Replication relationship first, before finalizing the migration on the primary system. Likewise, finalize the migration on all Reader Cache sites in a Remote Caching configuration first, then on the Home site, and then on all Writer Cache sites.

   The migration should be finalized on all systems at roughly the same time – there should not be a significant amount of time during which a particular share was already finalized on one system, but not on all other systems which share data through File Copy Services.

8. Verify that the Filesystem is now marked as 'simple' on all systems:

   ```
   # updatefs <filesystem> --aclmode show
   Current ACL mode is 'simple' and up to date.
   EFSSG1000I The command completed successfully.
   ```

9. Re-connect all Samba clients. After all clients were re-connected, CREATOR_OWNER and / or CREATOR GROUP ACEs can now be used.

If not all Snapshots and / or not all Backups were expired and / or deleted during the above migration procedure, then there is a risk of having to restore data with the previous (pre-update) GPFS ACLs. If this is the case, then the following command can be used to manually migrate the restored data – which will make it consistent with the data already migrated according to the above procedure:

```
updatefs <filesystem> --aclmode update --aclpath <path>
```

It is critically important to limit the scope of this command to the path of files which were being restored with previous (pre-update) ACLs. Failure to do so will result in invalid / inconsistent access permissions being applied to the data.

Migrating Filesystems marked as 'simple'

Filesystems marked as 'simple' only contain shares with an ACL Mode setting of Simple. No modification of the GPFS ACLs is required for data in these shares. However, the shares in such Filesystems will still have a share-specific (local) ACL Mode setting. As the global Samba setting is changed to Simple by the automatic upgrade process, such share-specific (local) ACL Mode settings are redundant and can be removed.

1. Verify that the Filesystem is marked as “simple” on all systems which share access to the Filesystem through File Copy Services (Asynchronous Replication and / or Remote Caching):

   ```
   # updatefs <filesystem> --aclmode show
   Current ACL mode is 'simple' and up to date.
   EFSSG1000I The command completed successfully.
   ```

2. To remove the share-specific (local) ACL Mode setting for all shares in the Filesystem:

   ```
   # updatefs <filesystem> --aclmode forcesimple
   EFSSG1000I The command completed successfully.
   ```

   After this step, the global Samba ACL Mode gets applied to all shares in the Filesystem, which is consistent with the default Storwize V7000 Unified configuration.

   Perform the above step on the secondary system in a Replication relationship first, before performing it on the primary system. Likewise, perform the above step on all Reader Cache sites in a Remote Caching configuration first, then on the Home site, and then on all Writer Cache sites.

   The above step should be performed on all systems at roughly the same time – there should not be a significant amount of time during which a particular share was already processed on one system, but not on all other systems which share data through File Copy Services.

As the effective ACL Mode for shares in these Filesystems is not altered by the above procedure, Samba clients do <u>not</u> need to re-connect after these steps have been carried out.

Migrating Filesystems marked as 'inconsistent'

Filesystems marked as 'inconsistent' contain at least one share with an ACL Mode setting of Intermediate, and at least one share with an ACL Mode setting of Simple.

The GPFS ACLs need to be modified for all data in shares set to Intermediate Mode, but data in shares set to Simple Mode <u>must not</u> be modified. If GPFS ACLs for data in shares already set to Simple Mode is modified, then legitimate CREATOR_OWNER / CREATOR_GROUP ACEs may get deleted.

After the GPFS ACLs have been modified accordingly (re-written), the Samba configuration can be switched to ACL Mode Simple for all shares in these Filesystems. This includes removing the now redundant share-specific (local) ACL Mode settings for shares already set to Simple Mode.

1. Verify that the Filesystem is marked as 'inconsistent' on all systems which share access to the Filesystem through Asynchronous Replication:

   ```
   # updatefs <filesystem> --aclmode show
   Current ACL mode is inconsistent for filesystem shares.
   EFSSG1000I The command completed successfully.
   ```

2. Identify the shares which need to be migrated. Only shares which were set to ACL Mode Intermediate need to be migrated, while shares already set to ACL Mode Simple <u>must not</u> be modified.

   1. Identify the mountpoint of the Filesystem:

      ```
      # lsfs -v -d <filesystem>
      ```

      Specifically note the **mountpoint** which is listed in the column titled '`Mount point`' in the output of the above command. In IBM Storwize V7000 Unified systems, mountpoints begin with '`/ibm/`'.

   2. Identify all shares inside this mountpoint:

```
# lsexport | grep <mountpoint>
```

Specifically note the **share name** which is listed in the first column of the output of the above command, as well as the **path** which is listed in the second column of the output.

3. Use the following command to query the current ACL Mode of a particular share:

   ```
   # servicecmd net conf getparm <share name> nfs4:mode
   ```

   Use this command to query the current ACL Mode of all shares identified in the previous step. Shares which have an ACL Mode setting of '`simple`' must be ignored in the following step. All shares which have a current ACL Mode setting of '`simplenocreator`' need to be migrated, as explained in the next step(s).

   The above command requires administrative user privileges – the user which runs the command needs to be a member of the '`Privileged`' group. To add the default 'admin' user account to this group, run the following command:

   ```
   # chuser admin -a Privileged
   ```

3. Update the GPFS ACLs for all data which needs to be migrated, as identified in the previous step. Specify the path (not the share name) of each share with a current ACL Mode setting of '`simplenocreator`'. Run the command for each share, or for each directory inside each share identified in the previous step.

   Migrate the data on the primary system in a Replication relationship:

   ```
   # updatefs <filesystem> --aclmode update --aclpath <path>
   start processing of /ibm/...
   traversed ... files, processed ... ACL's and changed ....
   Hit 0 issues during this update run.
   Multipe runs are recommended for online updates.
   processing completed.
   EFSSG1000I The command completed successfully.
   ```

   Performance tests in lab environments have shown that the above command is capable of processing 100 mio. files in roughly 5 Minutes. The actual performance will depend on the overall workload of the system, among other factors.

4. Repeat the step above until no further updates are applied to the Filesystem data. Run the command for each share, or for each directory inside each share identified in the previous step:

   ```
   # updatefs <filesystem> --aclmode update --aclpath <path>
   start processing of /ibm/...
   traversed ... files, processed ... ACL's and changed 0.
   Hit 0 issues during this update run.
   Multipe runs are recommended for online updates.
   processing completed.
   EFSSG1000I The command completed successfully.
   ```

   Specifically note the line that reads "`... and changed 0.`" in the above output. If, during the second run of the above steps, further updates are made to the Filesystem data, then additional runs of the '`updatefs <filesystem> --aclmode update --aclpath <path>`' command are necessary. Run the command repeatedly until no further updates are applied.

5. If the Filesystem is configured for automatic Snapshots, for automatic Backups (via TSM or NDMP), or the Filesystem is part of an Asynchronous Replication relationship, then it is recommended to wait for these scheduled tasks to pick up the changes before continuing. Ideally, the following steps are being carried out after all previous Snapshots and all previous Backups were expired and / or deleted, to avoid restoring data with the previous (pre-update) GPFS ACLs.

6. Use the following command to finalize migration of the Filesystem:

   ```
   # updatefs <filesystem> --aclmode finalize
   EFSSG1000I The command completed successfully.
   ```

   This will remove the share-specific (local) ACL Mode setting for all shares in the Filesystem. After this step, the global Samba ACL Mode gets applied to all shares in the Filesystem, which is consistent with the default Storwize V7000 Unified configuration.

   Perform the finalization of the migration on the secondary system in a Replication relationship first, before finalizing the migration on the primary system.

   The migration should be finalized on all systems at roughly the same time – there should not be a significant amount of time during which a particular share was already finalized on one system, but not on all other systems which share data through File Copy Services.

7. Verify that the Filesystem is now marked as 'simple' on all systems:

   ```
   # updatefs <filesystem> --aclmode show
   Current ACL mode is 'simple' and up to date.
   EFSSG1000I The command completed successfully.
   ```

8. Re-connect all Samba clients. After all clients were re-connected, CREATOR_OWNER and / or CREATOR GROUP ACEs can now be used.

If not all Snapshots and / or not all Backups were expired and / or deleted during the above migration procedure, then there is a risk of having to restore data with the previous (pre-update) GPFS ACLs. If this is the case, then the following command can be used to manually migrate the restored data – which will make it consistent with the data already migrated according to the above procedure:

```
updatefs <filesystem> --aclmode update --aclpath <path>
```

It is critically important to limit the scope of this command to the path of files which were being restored with previous (pre-update) ACLs. Failure to do so will result in invalid / inconsistent access permissions being applied to the data.

6. Test documentation

The IBM Storwize V7000 Unified software (firmware) upgrade procedure from version 1.4 to version 1.5 automatically changes certain ACL Mode settings. The ACL Mode of existing shares will be modified based on their current ACL Mode setting. In addition, Filesystems will also be marked with a certain Mode after the upgrade. The following tests were conducted to determine the specific changes which are applied based on the current configuration.

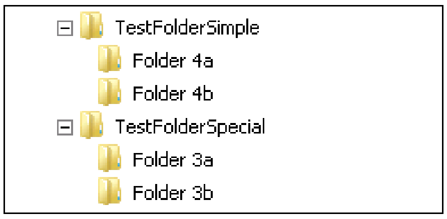
Software (firmware) version before upgrade: 1.4.3.1-4
Software (firmware) version after upgrade: 1.5.0.6-4

A test system was installed and configured with version 1.4, before an upgrade to version 1.5 was performed on this test system. The following table lists the test cases which were configured, in order to address all possible combinations of shares and Filesystems:

Test	Filesystem	Share	ACL Mode		Filesystem Status
			Before	After	
1	*ExclusiveGlobal*	*Ex_Gl*	-	Intermediate	intermediate
2	*ExclusiveSpecial*	*Ex_Sp*	Special	Intermediate	intermediate
3	*ExclusiveSimple*	*Ex_Si*	Simple	Simple	simple
4		*Mu_Gl_1*	-	Intermediate	
5	*MultiGlobal*	*Mu_Gl_2*	-	Intermediate	intermediate
6		*Mu_Gl_3*	-	Intermediate	
7		*Mu_Sp_1*	Special	Intermediate	
8	*MultiSpecial*	*Mu_Sp_2*	Special	Intermediate	intermediate
9		*Mu_Sp_3*	Special	Intermediate	
10		*Mu_Si_1*	Simple	Simple	
11	*MultiSimple*	*Mu_Si_2*	Simple	Simple	simple
12		*Mu_Si_3*	Simple	Simple	
13	*SharedGlobalSpecial*	*Sh_Gl_Sp_1*	-	Intermediate	intermediate
14		*Sh_Gl_Sp_2*	Special	Intermediate	
15	*SharedGlobalSimple*	*Sh_Gl_Si_1*	-	Intermediate	inconsistent
16		*Sh_Gl_Si_2*	Simple	Simple	
17	*SharedSpecialSimple*	*Sh_Sp_Si_1*	Special	Intermediate	inconsistent
18		*Sh_Sp_Si_2*	Simple	Simple	
19		*Sh_Mi_1*	-	Intermediate	
20	*SharedMixed*	*Sh_Mi_2*	Special	Intermediate	inconsistent
21		*Sh_Mi_3*	Simple	Simple	
22		*Mu_Sh_Gl_Sp_1*	-	Intermediate	
23	*MultiSharedGlobalSpecial*	*Mu_Sh_Gl_Sp_2*	-	Intermediate	intermediate
24		*Mu_Sh_Gl_Sp_3*	Special	Intermediate	
25		*Mu_Sh_Gl_Sp_4*	Special	Intermediate	
26		*Mu_Sh_Gl_Si_1*	-	Intermediate	
27	*MultiSharedGlobalSimple*	*Mu_Sh_Gl_Si_2*	-	Intermediate	inconsistent
28		*Mu_Sh_Gl_Si_3*	Simple	Simple	
29		*Mu_Sh_Gl_Si_4*	Simple	Simple	
30		*Mu_Sh_Sp_Si_1*	Special	Intermediate	
31	*MultiSharedSpecialSimple*	*Mu_Sh_Sp_Si_2*	Special	Intermediate	inconsistent
32		*Mu_Sh_Sp_Si_3*	Simple	Simple	
33		*Mu_Sh_Sp_Si_4*	Simple	Simple	
34		*Mu_Sh_Mi_1*	-	Intermediate	
35		*Mu_Sh_Mi_2*	-	Intermediate	
36	*MultiSharedMixed*	*Mu_Sh_Mi_3*	Special	Intermediate	inconsistent
37		*Mu_Sh_Mi_4*	Special	Intermediate	
38		*Mu_Sh_Mi_5*	Simple	Simple	
39		*Mu_Sh_Mi_6*	Simple	Simple	

Each share was populated with sample data, to verify the migration procedures as outlined in the previous chapters. This sample data was discussed in detail in Example 3 from Chapter 3.1 "*ACL Mode Special – Problems in Windows environments*", page 7, as well as in Example 4 from Chapter 4 "*ACL Mode Simple*", page 14.

The following figure shows this sample test data:



Before the upgrade, the sample data has the following ACLs:

```
./TestFolderSpecial
#NFSv4 ACL
#owner:VIRTUAL\user1
#group:VIRTUAL\domain users
special:owner@:rwxc:allow:FileInherit:DirInherit
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

special:everyone@:r-x-:allow
 (X)READ/LIST (-)WRITE/CREATE (-)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (-)DELETE    (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL
(-)WRITE_ATTR (-)WRITE_NAMED

user:VIRTUAL\user2:rwxc:allow:FileInherit:DirInherit
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

./TestFolderSpecial/Folder\ 3a
#NFSv4 ACL
#owner:VIRTUAL\user1
#group:VIRTUAL\domain users
special:owner@:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

user:VIRTUAL\user2:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

./TestFolderSpecial/Folder\ 3b
#NFSv4 ACL
#owner:VIRTUAL\user2
#group:VIRTUAL\domain users
special:owner@:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED
```

```
user:VIRTUAL\user2:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED
```

./TestFolderSimple

```
#NFSv4 ACL
#owner:VIRTUAL\user1
#group:VIRTUAL\domain users
special:everyone@:r-x-:allow
 (X)READ/LIST (-)WRITE/CREATE (-)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (-)DELETE    (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL
(-)WRITE_ATTR (-)WRITE_NAMED

user:VIRTUAL\user1:rwxc:allow:FileInherit:DirInherit
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

user:VIRTUAL\user2:rwxc:allow:FileInherit:DirInherit
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED
```

./TestFolderSimple/Folder\ 4a

```
#NFSv4 ACL
#owner:VIRTUAL\user1
#group:VIRTUAL\domain users
user:VIRTUAL\user1:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

user:VIRTUAL\user2:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED
```

./TestFolderSimple/Folder\ 4b

```
#NFSv4 ACL
#owner:VIRTUAL\user2
#group:VIRTUAL\domain users
user:VIRTUAL\user1:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

user:VIRTUAL\user2:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED
```

After applying the upgrade procedures outlined in this document, the sample data has the following ACLs:

- Sample data which resided in a share previously set to ACL Mode Special:

```
./TestFolderSpecial
#NFSv4 ACL
#owner:VIRTUAL\user1
#group:VIRTUAL\domain users
user:VIRTUAL\user1:rwxc:allow:FileInherit:DirInherit
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

special:everyone@:r-x-:allow
 (X)READ/LIST (-)WRITE/CREATE (-)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (-)DELETE    (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL
(-)WRITE_ATTR (-)WRITE_NAMED

user:VIRTUAL\user2:rwxc:allow:FileInherit:DirInherit
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

./TestFolderSpecial/Folder\ 3b
#NFSv4 ACL
#owner:VIRTUAL\user2
#group:VIRTUAL\domain users
user:VIRTUAL\user2:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

user:VIRTUAL\user2:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

./TestFolderSpecial/Folder\ 3a
#NFSv4 ACL
#owner:VIRTUAL\user1
#group:VIRTUAL\domain users
user:VIRTUAL\user1:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

user:VIRTUAL\user2:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED
```

- Sample data which resided in a share previously set to ACL Mode Simple:

```
./TestFolderSimple
#NFSv4 ACL
#owner:VIRTUAL\user1
#group:VIRTUAL\domain users
special:everyone@:r-x-:allow
 (X)READ/LIST (-)WRITE/CREATE (-)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (-)DELETE    (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL
(-)WRITE_ATTR (-)WRITE_NAMED

user:VIRTUAL\user1:rwxc:allow:FileInherit:DirInherit
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

user:VIRTUAL\user2:rwxc:allow:FileInherit:DirInherit
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

./TestFolderSimple/Folder\ 4a
#NFSv4 ACL
#owner:VIRTUAL\user1
#group:VIRTUAL\domain users
user:VIRTUAL\user1:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

user:VIRTUAL\user2:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

./TestFolderSimple/Folder\ 4b
#NFSv4 ACL
#owner:VIRTUAL\user2
#group:VIRTUAL\domain users
user:VIRTUAL\user1:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED

user:VIRTUAL\user2:rwxc:allow:FileInherit:DirInherit:Inherited
 (X)READ/LIST (X)WRITE/CREATE (X)MKDIR (X)SYNCHRONIZE (X)READ_ACL
(X)READ_ATTR  (X)READ_NAMED
 (X)DELETE    (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL
(X)WRITE_ATTR (X)WRITE_NAMED
```

.