FEBRUARY 1993

NOVELL® RESEARCH

Mapping Between UNIX Permissions and NetWare Rights in NetWare NFS

Mala Ranganathan
Technical Marketing Manager
Connectivity Product Marketing

This Application Note provides a basic explanation of how mapping is done between UNIX permissions and NetWare Rights for files and directories in NetWare NFS. It includes a brief discussion on how UNIX permissions and NetWare's access control work, followed by details on how permissions are mapped between NFS and NetWare. The explanation is intended for both novice and veteran users of UNIX and NetWare.

Contents

Acknowledgements

My sincere thanks to Jeremy Liang (architect of the NFS permissions / NetWare rights mapping design) for his valuable input. My special thanks to Auston Erwin, Henry Leung, Albert Leung, Mark Painter, Ram Ravuri, Keith Brown, Milton Howard, Mike Kirsch, and Jack Thomasson for taking the time to review this AppNote and provide valuable feedback.

Introduction

This AppNote provides a basic explanation of the translation between NFS permissions and NetWare rights (and vice-versa) for files and directories, in providing cross-platform access support in NetWare NFS. We start with a brief discussion of how UNIX permissions and NetWare access control work. This is followed by details on how permissions are mapped between NFS and NetWare. The terms UNIX permissions and NFS permissions are used interchangeably throughout the AppNote.

Overview of UNIX Permissions

NetWare NFS employs a UNIX-style access control mechanism based on user (owner), group, and world (other) permissions for file and directory access. The permissions specify who can do what to a file or directory, and they are set by the owner.

For each file or directory, you can set read (r), write (w), and execute/search (x) permissions for:

- The owner of the file or directory

- The <u>group</u> associated with the file or directory
- All <u>others</u> on the system

Permission Mask

The access to a file or directory is determined solely by the user's identification code (UID), the user's primary group identification code (GID), and the specific permission mask on the file or directory.

A UNIX permission mask (in absolute mode) is a three-digit octal number that defines how the owner, group, and others can access the file or directory. (An octal number contains only digits 0-7.) The permission mask is constructed by taking the logical OR (addition) of the modes shown in Figure 1.

Figure 1: UNIX permission bits for files and directories.

<u>UNIX Permission Mask</u>		<u>Permissions</u>			<u>Explanation</u>
Octal	Binary	Owner	Group	Others	
400	100 000 000	r--	---	---	Read by owner
200	010 000 000	-w-	---	---	Write by owner
100	001 000 000	--x	---	---	Execute file or search by owner
040	000 100 000	---	r--	---	Read by group
020	000 010 000	---	-w-	---	Write by group
010	000 001 000	---	--x	---	Execute file or search by group
004	000 000 100	---	---	r--	Read by others
002	000 000 010	---	---	-w-	Write by others
001	000 000 001	---	---	--x	Execute file or search by others

File Permissions

For files, the UNIX permissions have the following definitions:

r Allows the user to see the contents of a file and make copies of the file.

w Allows the user to modify the contents of a file or delete the file.

x Allows the user to execute a file (either a compiled program or a shell script).

Figure 2 shows an example of how the permission mask for a file is obtained by taking the logical sum of the permissions for the owner, group, and others.

Figure 2: The permission mask for a file is obtained by taking the logical sum of owner, group, and others permissions.

<u>UNIX Permission Mask</u>		<u>Permissions</u>				<u>Explanation</u>
Octal	Binary	Owner	Group	Others		
400	100 000 000	r--	---	---		Read by owner
200	010 000 000	-w-	---	---		Write by owner
100	001 000 000	--x	---	---		Execute/search by owner
040	000 100 000	---	r--	---		Read by group
020	000 010 000	---	-w-	---		Write by group
004	000 000 100	---	---	r--		Read by others
ÄÄ						
764	111 110 100	rwx	rw-	r--		Permission mask for file

As shown in Figure 2, permission mask 0764 (octal) for a file owned by user mala and group staff translates to the following:

- User mala can read, write (modify/delete), and execute the file.

- Members of the group "staff" can read and write to (modify/ delete) the file.
- All other users (that is, all users other than "mala" and members of group "staff") can only read the file.

The UNIX ls command lists information about files in a directory. If the file in Figure 2 is named events.txt, the ls -l command displays this information as follows:

```
%ls -lg
        -rwxrw-r--   1 mala   staff  145  Aug 8 17:15 events.txt
```

Directory Permissions

For directories, the UNIX permissions have the following definitions:

r Allows user to list the contents of the directory. (It doesn't allow the user to read the contents of the files, because access to files is controlled by the permissions on the individual files.)

w Allows user to create, modify, delete, or rename files and subdirectories in the directory, provided that the permissions on existing files and subdirectories allow these operations.

x Allows user to change to a directory using the cd command and list the files in the directory using the ls command.

Permission masks apply to directories as well. For example, permission mask 0755 (octal) for a directory named mydir owned by user mala and group staff translates to the following:

- User mala can list, create, modify, delete, rename, cd to directory mydir.
- Members of the group staff can list files and cd to directory mydir.
- All other users can list files and cd to directory mydir.

The ls -l command displays this information as follows:

```
%ls -ldg
        drwxr-xr-x  1 mala    staff     512 Aug 8 17:30 mydir
```

Default Permissions

The default permissions for files or directories created in UNIX by the shell (from the command line) are:

- Files 666 (octal)
- Directories 777 (octal)

Files or directories created by programs do not have the above-mentioned default permissions. The program typically specifies its own set of permissions for any files and directories it creates.

Whether a file/directory is created by a program or by the UNIX shell, the permissions are reduced by the umask, as explained in the next section.

The Userfile Creation Mask (umask)

In UNIX, users can set a umask (userfile creation mask) value that affects the permissions for newly-created files and directories. You view and change the umask value with the umask command. For example:

umask displays the current umask value

umask 000	changes the umask value to 000

The default umask value is 000 (octal), which does not subtract any permissions from the permission mask. When you set the umask value to something other than 000, the corresponding permissions are withheld when new files and directories are created. In other words, if w permission is to be denied to others for newly created files and directories, the user sets the umask to 002 (octal). Another typical umask value is 022 (octal), which denies "w" permission to everyone but the owner.

The final permissions are calculated by logically subtracting the umask value from the default permissions (or from the permissions specified by a program).

For example, suppose the user sets the umask value to 022. A file that is created in UNIX (with default permissions of 666) will have final permission bits given by octal subtraction:

666 - umask value = 666 - 022 = 644, which is -rw-r--r--

Similarly, a directory that is created in UNIX (with default permissions of 777) will have final permission bits given by octal subtraction:

777 - umask value = 777 - 022 = 755, which is drwxr-xr-x

Permissions Required for File/Directory Tasks

The table in Figure 3 lists the permissions a user must have in order to perform various tasks relating to files and directories in the UNIX environment.

Figure 3: Permissions required to perform various file and directory tasks in the UNIX environment.

Task	Permissions Required	
	File	Directory
Read the contents of a file	r	(r)x *
Copy a file from a directory	r	x
Modify a file in a directory	rw	wx
Delete a file in a directory		wx
Execute a file (program)	x	x
List files in a directory		rx
Create a file in a directory		wx
Rename a file in a directory	r	wx
cd to a directory, or use directory as part of a pathname		x

* With no r permission to the directory, the user cannot do ls and list files in it. But the user can still read a file in the directory if he/she has read permission for the file and knows its name. The user should also have x permission for the directory to either cd to it and perform a cat <filename> command, or do a cat <pathname> command from the current directory.

Rules of Thumb for Assigning UNIX Permissions

Most UNIX implementations adopt the following method in determining who can do what on a particular file or directory:

- After a user logs in, for every file access the system of the login user compares the UID with the owner of the file. If he/she is the owner, he/she gets the owner permissions.

- If the user belongs to the same group as the owner (due to membership in primary or secondary groups), he/she gets the group permissions.
- If the user doesn't fall into either of the above two categories, he/she is considered as other and gets world permissions.

This process is illustrated in Figure 4.

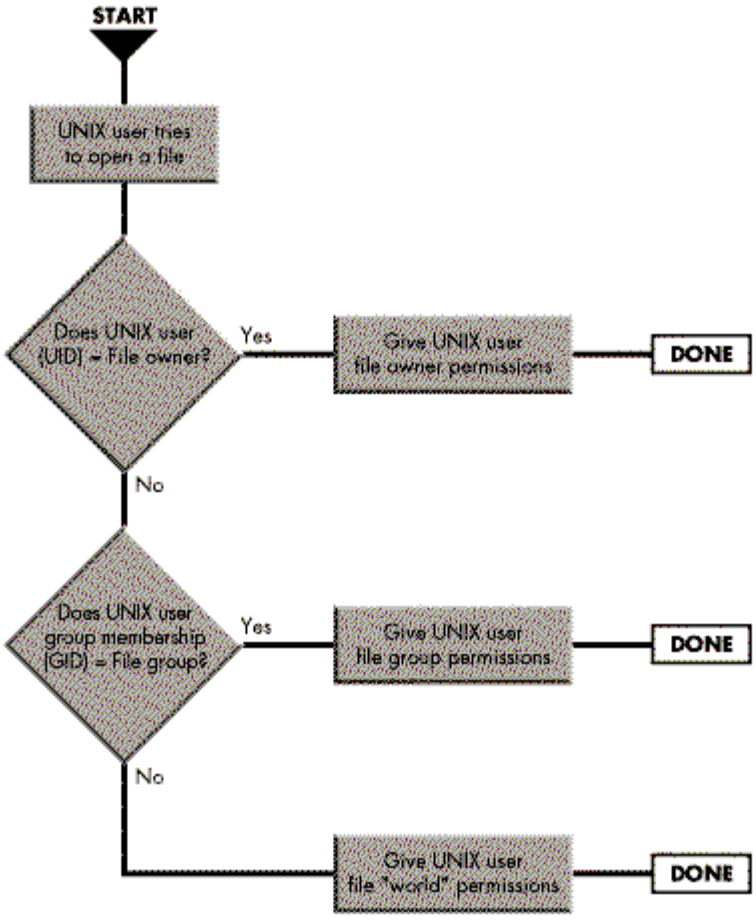


Figure 4: Flowchart showing how UNIX permissions are determined.

It may happen that a user is not the owner of a file, but belongs to the same group as the owner. In this case, the user may not be able to do things with the file that users belonging to others can do. The reason is that UNIX always checks permissions for the smallest category that applies. So even though the world permissions allow the user to do something, if the group permissions do not allow it, the user won't be able to do it.

A few examples might help clarify these concepts. First, consider the file schedule.txt with the following permissions:

```
----rw----   1 mala    staff  90 Aug 8 17:31 schedule.txt
```

The permissions mask allows r and "w" permissions only to members of the group "staff".

- User "mala" cannot use the cat schedule.txt command to copy the file.

- User "notmala" (who is placed as a member of the secondary group "staff" in the /etc/group file) can do cat schedule.txt but cannot do chmod schedule.txt to change the file permissions.

Now consider the directory mydir with the following permissions:

drwx---r-x 1 mala staff 512 Aug 8 17:25 mydir

- User "guest" belonging to group "staff" cannot list files in the directory mydir since "staff" has no permissions.
- User "other" belonging to group "othergroup" can list files in the directory mydir since "world" has r,x permissions.

Here are some other general guidelines to follow when assigning UNIX permissions.

1. It is always desirable to have the "x" permission for a directory since this is needed to cd to it, open it and look for a file, or use it as part of a pathname in accomplishing the given task.
2. Without "wx" permissions to the directory and "w" permission to a file, the file cannot be removed or modified.
3. With "wx" permissions to the directory and no "w" permission to a file, the protection for the file can be overridden and removed.
4. Certain UNIX commands can only be performed by the file owner or the superuser (username root):
    - Only the file owner or superuser can do chmod to change the file's permissions.
    - Only superuser can do chown to change the file's owner.
    - Only superuser (and a file owner who happens to be a member of the group she/he is changing ownership to) can do chgrp to change the file's group ownership.

NetWare NFS Setup Information

UNIX systems maintain a number of files in the /etc directory that are required for system administration. Below is a partial listing of files that contain information for setting up NFS at the NetWare NFS file server.

/etc/passwd	Contains UIDs, GIDs (user account information)
/etc/group	Contains group names, GIDs, members
/etc/hosts	Contains host's Internet addresses
/etc/fstab	Contains file systems mounted during boot time when the UNIX operating system comes up

We'll refer to these files later when we discuss mapping between UNIX/NFS permissions and NetWare access control.

Overview of NetWare Access Control

In NetWare, access control is primarily based on trustee rights assignments which specify who has what rights on the associated object. The objects file and directory are of specific interest to us. The effective rights a user has for a file or directory depends on the combination of the following:

- Trustee rights assignment

- Inherited rights mask
- Security equivalences
- File/directory attributes

Trustee Rights Assignment

A trustee assignment grants rights in a file or a directory to specific users or groups. A trustee assignment automatically grants users the right to see (list files) up to the root of a directory. However, the users cannot see any of the subdirectories, unless they have also been granted rights in the subdirectories.

Unlike UNIX, NetWare does not explicitly assign trustee rights to every file or directory that is created. Trustee rights are propagated down the directory structure until reset by another trustee right assignment at some lower level.

Inherited Rights Mask

An Inherited Rights Mask (IRM) is given to each file and directory when it is created. The default IRM includes all NetWare rights: [SRWCEMFA]. (The meaning of these rights is explained later.) However, users may not be able to exercise all rights since they can only use the rights they have been granted in trustee assignments (from the user's effective rights in the parent directory).

The IRM can be modified later on for files or subdirectories below the original trustee assignment. If the IRM is modified, the only rights the user can inherit for the file or subdirectory are the ones allowed by the IRM.

Security Equivalences

Any user can be made security equivalent to another user or group. This effectively transfers the effective rights (which are valid for the entire directory structure) from one user to another. In general, all users are security equivalent to group EVERYONE by default.

File/Directory Attributes

Attributes can be assigned to individual files or directories. These attributes override rights granted with trustee assignments and can prevent tasks that effective rights would allow. However, if users have the Modify right for the directory or the file, they can change the attributes and perform desired operations.

NetWare's file and directory attributes are listed below (bolded letters indicate abbreviations). Attributes that are applicable to NFS are marked with an asterisk (*).

File Attributes

Archive Needed	**R**ead **A**udit
Copy **I**nhibit	**R**ead **O**nly*/**R**ead **W**rite
Delete **I**nhibit*	**R**ename **I**nhibit*
E**X**ecute Only	**S**hareable
Hidden	**Sy**stem
Indexed	**T**ransactional*
Purge	**W**rite **A**udit

Directory Attributes

Delete **I**nhibit* **R**ename Inhibit

Hidden **S**ystem

Purge

Effective Rights

Effective rights are the ones a user can actually exercise in a given directory or file. The effective rights a user has for a particular directory or file are determined as follows:

Effective rights of user in parent directory — logical AND — Inherited Rights Mask

If a new trustee assignment is made to the directory or file, it overrides the directory's Inherited Rights Mask and the effective rights from the parent directory. In this case, the effective rights of a user in a directory or file are determined as follows:

Trustee assignment for the user — logical OR — Trustee assignment for the group(s) the user belongs to

Note: If the user's effective rights in the parent directory include the S (Supervisory) right, the user gets all rights to all files and directories down the directory tree, irrespective of the trustee assignments at some other level or modifications of Inherited Rights Masks.

Figure 5 shows an example of how effective rights are determined for NetWare user MALA, who belongs to the group STAFF, in the TRACK subdirectory.

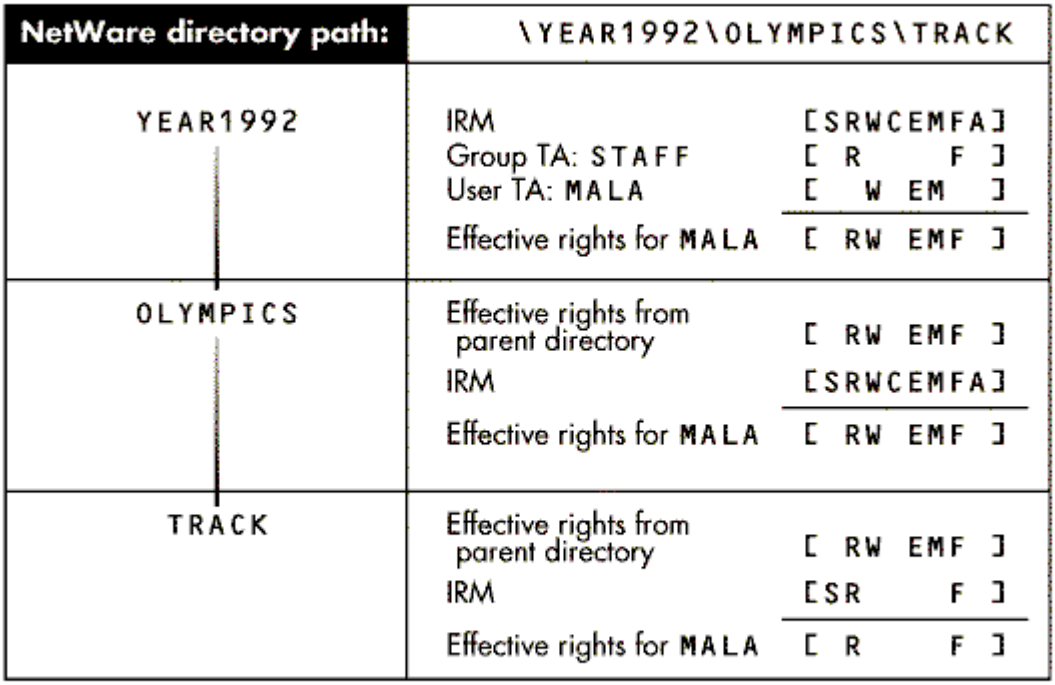


Figure 5: Effective rights are determined from user and group trustee assignments, IRMs, and rights in the parent directory.

NetWare Security Rights

Both trustee assignments and Inherited Rights Masks use the same eight NetWare rights to control access to directories and files. The meaning and effect of each of the trustee rights depends on whether the right is assigned to a directory or a file.

Directory Rights. Directory rights control access to directories, their files, and subdirectories unless redefined at the file or subdirectory level. Following is a list of directory rights:

S (Supervisory) A user with this right to a directory essentially has absolute power from the directory on down its subdirectory tree. The Supervisory right overrides any restrictions placed on subdirectories or files with an IRM. Once the Supervisory right has been granted, it can be revoked only from the directory to which it was granted. It cannot be revoked in a file or subdirectory.

R (Read) Grants the right to open files in a directory and read their contents or execute programs in the directory.

W (Write) Grants the right to open and modify (write to) files in the directory.

C (Create) Grants the right to create files and subdirectories within the directory.

E (Erase) Grants the right to delete a directory, its files, its subdirectories, and its subdirectory's files.

M (Modify) Grants the right to change the attributes (flags) of the directory and the files and subdirectories within it. This also grants the right to rename the directory and its files and subdirectories. It doesn't grant the right to modify the contents of a file.

F (File scan) Grants the right to see filenames within the directory and within its subdirectories.

A (Access control) Grants the right to change a directory's or a file's trustee assignment and Inherited Rights Mask.

File Rights. File rights are used to redefine the rights (access to specific files) that users inherit from parent directory rights. Following is a list of file rights:

S (Supervisory) Grants all rights to the file. A user with this right can grant any right to another user and can modify all rights in the file's Inherited Rights Mask.

R (Read) Grants the right to open and read the file.

C (Create) Grants the right to salvage the file after it has been deleted.

W (Write) Grants the right to open and write to the file.

E (Erase) Grants the right to delete the file.

M (Modify) Grants the right to change the file's attributes and rename the file, but not the right to modify the contents of the file.

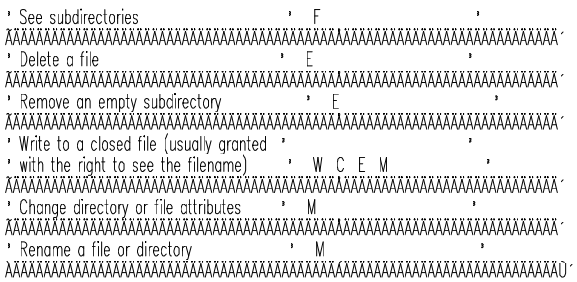
F (File scan) Grants the right to see the filename when viewing the directory. Also grants the right to see the directory structure from the file to the root of the directory.

A (Access control) Grants the right to change the file's trustee assignments and Inherited Rights Mask.

Figure 6 lists the effective rights a user needs to perform various file and directory tasks in the NetWare environment.

Figure 6: Effective rights needed to perform various file and directory tasks in the NetWare environment.

Task	Rights Required
Change trustee assignments	A
Change Inherited Rights Mask	A
Make a new directory	C
Copy a file (or directory) into a directory	C
Create and write to an opened file	C
Read from a closed file (usually granted with the right to see the filename)	R
Modify directory disk space assignments in subdirectories	S
See to the root directory	S / R / C / W / E / M / F / A
Copy a file from a directory	R F
See a filename	F



Mapping Between NetWare and NFS Environments

Directory and file security for NetWare and NFS are similar, although NetWare's security is more flexible and comprehensive. However, at the basic level, NFS and NetWare both assign permissions/trustee rights to a file/directory owner, group, and others (NetWare group EVERYONE).

The NetWare rights [SRWCEMFA] are not an absolute superset of the NFS rwx permissions. The rwx permissions have different meanings in the NFS and NetWare worlds. Hence some method for mapping between NFS permissions and NetWare rights is necessary so that the access rights can have similar meaning in both worlds.

With NetWare NFS, mapping between NFS permissions and NetWare rights is done automatically. When an exact mapping of rights between the two systems is not possible, a conversion is made in favor of tighter control (lesser permissions/rights). The access control information is always NetWare-based.

Whenever a NetWare NFS server is stopped and restarted, NFS permissions for the files and directories are recalculated from NetWare trustee rights assignments stored in the DOS name space. This is done because the NetWare server might still be running while the NetWare NFS server is down, and access control changes could have been made from other desktop platforms (DOS, OS/2, Macintosh, and so on).

Figure 7 illustrates a typical cross-platform scenario in which both DOS and UNIX clients access the same file on a server running NetWare NFS.

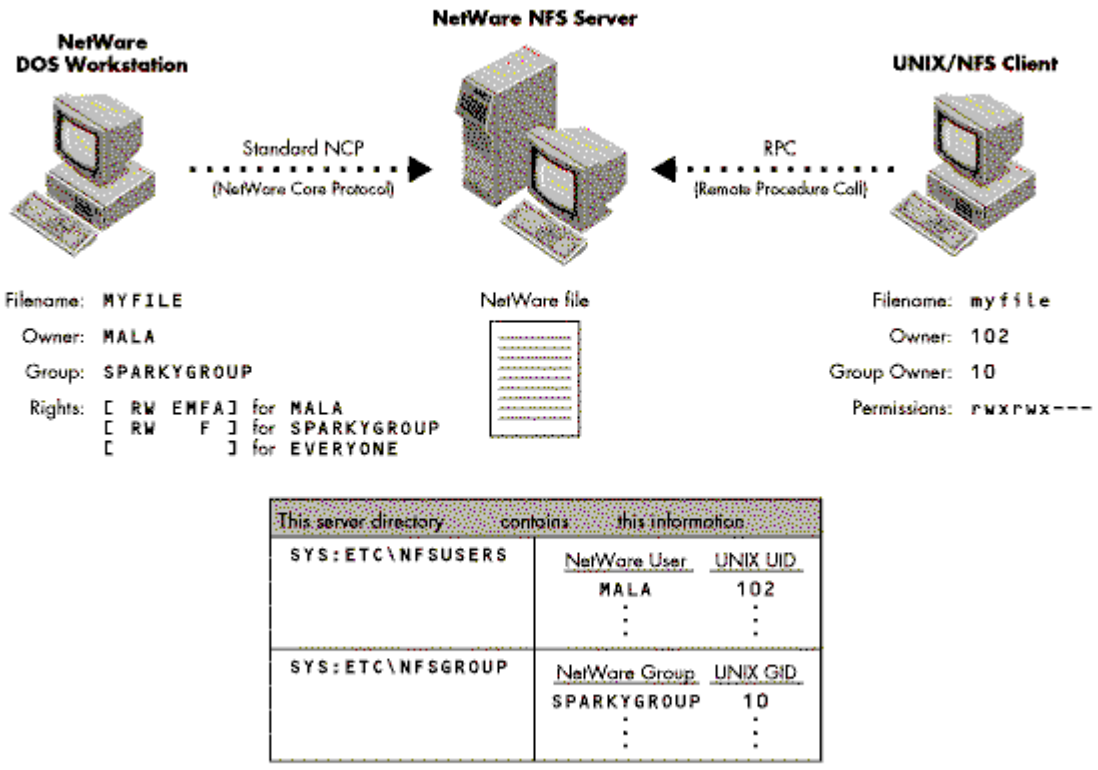


Figure 7: Scenario illustrating cross-platform file access between a NetWare DOS client and a UNIX/NFS client.

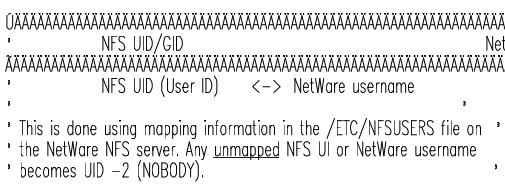
The sections which follow describe the rules (and exceptions to the rules) for direct mapping between the two file systems. Once you understand these rules, you will still need to apply some intuitive knowledge to figure out appropriate rights to make sure the access control is the same on both systems.

Mapping Between UID/GID and NetWare Users/Groups

The first thing NetWare NFS does is attempt a one-to-one mapping between NFS UIDs (user IDs) and GIDs (group IDs) and NetWare users and groups. When you install NetWare NFS, a NetWare user named NOBODY and a NetWare group named NOGROUP are created automatically to correspond to UID -2 and GID -2, respectively.

Figure 8 describes how NFS and NetWare user/group classes are mapped.

Figure 8: Rules for mapping NFS UID to NetWare username and NFS GID to NetWare groupname.



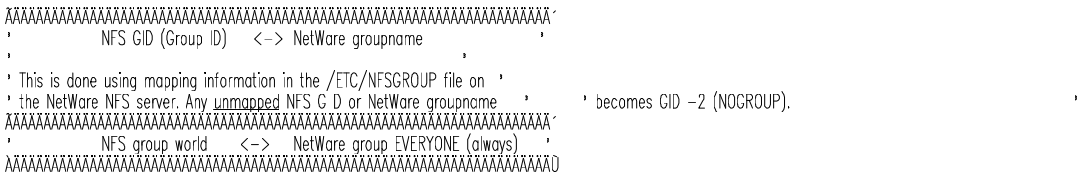


Figure 9 shows an example of how UID/GID mapping works, given the information in the NFSUSERS and NFSGROUP files on the NetWare NFS server.

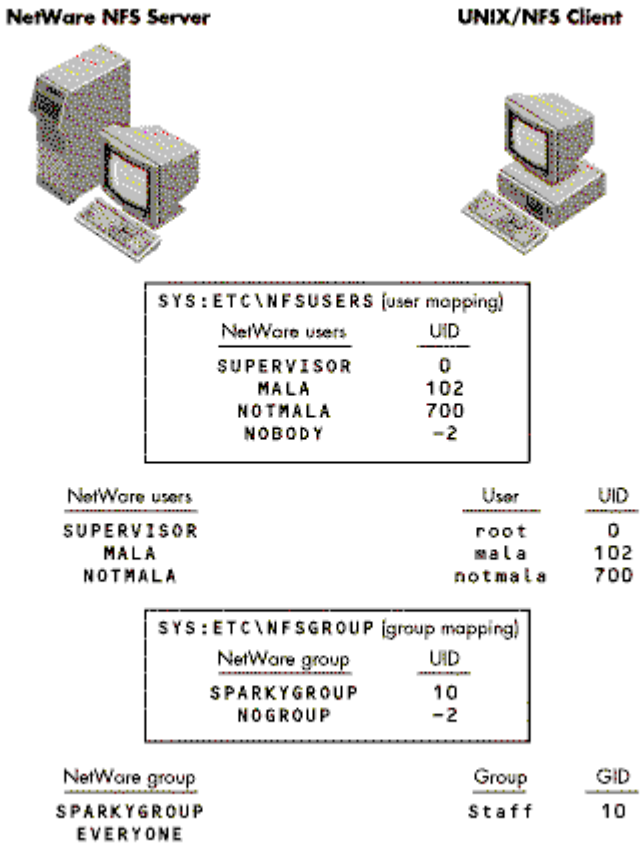


Figure 9: Example of how NetWare NFS maps UIDs and GIDs to NetWare users and groups.

UID/GID Mapping Exceptions. Some exceptions to the UID and GID mapping rules are explained below.

UID = -2

The superuser (root) on an NFS client machine will be mapped to UID -2 if the export option for the mounted file system does not allow remote root access for this client machine.

UID = -3

The owner of a file or directory will be mapped to UID -3 in the following exceptional cases:

- If a file system is exported with the option Do not modify DOS attributes from NFS and

- a <u>file</u> below it has the NetWare **DI** (Delete Inhibit) or **RI** (Rename Inhibit) or **T** (Transactional) attribute associated with it,

  or

- a <u>directory</u> below it has the NetWare **DI** (Delete Inhibit) attribute associated with it,

the owner of the parent directory will be mapped to UID -3.

- If a file system is exported with the option Do not modify DOS attributes from NFS and
  - a file below it has the NetWare **RO** (Read Only) attribute associated with it,

  the owner of both the file and its parent directory will be mapped to UID -3.

- If a file system is exported with the option Modify DOS attributes from NFS and
  - a file below it has the NetWare **T** (Transactional) attribute associated with it,

  the owner of its parent directory will be mapped to UID -3.

- If the owner of a file or directory does not have the Access Control right on the NetWare side, the owner will be mapped to UID -3.
- If a mapping results in NOBODY, but the default NetWare user NOBODY does not exist and cannot be created, the user will be mapped to UID -3.

Note: Refer to the section on "Exporting NetWare Directories to NFS Clients" for information on export option settings for exported file systems.

NFS-to-NetWare Mapping

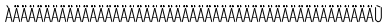
A new permissions mode is mapped to corresponding NetWare trustee assignments on the DOS side whenever one of the following actions takes place in an NFS mounted file system residing in a NetWare NFS server:

- A file or directory is created
- A file or directory's permissions are changed
- Permissions are generated (on access)

The entire set of NFS permissions is stored in the directory entries of the <u>NFS name space</u> for future use. This includes the entire mode field: set uid/gid bit (a bit used in UNIX to set user ID/group ID on execution), the sticky bit (a bit set in UNIX to save program text between processes), and so on. This information might be needed for reverse mapping, when permissions have to be recalculated for files and directories from NetWare trustee rights. One-to-one mapping between the two systems is not always possible, as in the case of the x (execute) permission, which doesn't have an equivalent NetWare right.

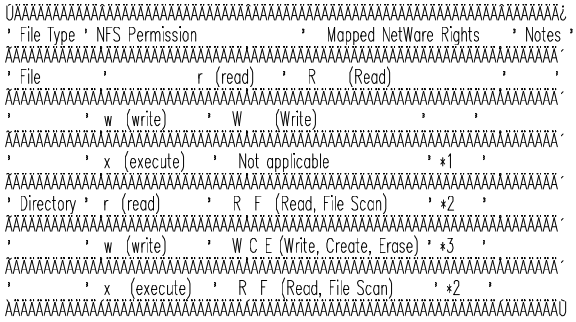
The NFS permissions for user, group, and others are converted to NetWare trustee rights assignments as follows:

NFS Permissions -->	NetWare Trustee Rights Assignment
user	owner
group	group
world	EVERYONE



The converted trustee rights assignments are stored in the directory entries of the DOS name space. Figure 10 shows how specific NFS permissions translate to NetWare rights.

Figure 10: Translation from NFS access permissions to NetWare trustee rights.



Notes:

*1 Since there isn't an equivalent NetWare right for the x permission, it is not used in mapping. The NetWare Execute-Only file attribute makes a file nonreadable and nonwritable, which is not what the NFS x permission stands for. The "x" permission is kept in the NFS name space only. It is not reflected in the NetWare trustee assignments.

*2 The File Scan right allows users to see directory or file listings. The Read right is given along with File Scan for UNIX-DOS interoperability, so that the user will be able to perform UNIX-like operations (for example, the user can copy files from the parent directory with r,x permissions).

NFS	NetWare
read permission with search bit on -->	[R F]
(r,x for a directory)	
or	
search permission with read bit on -->	[R F]
(x,r for a directory)	

The File Scan right is granted for a directory only if the user has the combination of read and search (r,x) permissions (necessary to be able to do ls and cd commands on the UNIX side) in the directory.

*3 The NFS "w" permission for a directory allows you to modify, create, delete, and rename files and directories below it. The corresponding NetWare rights are Write, Create, Erase, and Modify. However, the NetWare Modify right also allows renaming as well as changing of file and directory attributes. The user is not automatically granted the NetWare Modify right, which would give the user the ability to rename NFS directories and files.

NFS Permissions-to-NetWare Rights Mapping Rules

There are four handy rules you need to remember about the mapping of NFS permissions to NetWare rights. These rules are listed below.

Rule 1:

The owner of a file or directory is assigned the **A**ccess Control right on the NetWare side, because this right

is a prerequisite for owning a file in NFS.

On the NFS side, only the owner of a file/directory or the superuser (root) can use the chmod command to change the access permissions. In NetWare, the owner of a file/directory doesn't have any special privileges for the file/directory. **A**ccess Control rights have to be explicitly specified via trustee right assignments.

The denial of the **A**ccess Control right from the NetWare side revokes ownership as seen from NFS, and the owner gets changed to an UID of þ3.

Example. The following example may help clarify these concepts. From a DOS workstation, user MALA uses the NetWare TLIST command to see what rights she has to the file MYFILE in the directory MYDIR.

```
Q:\MYDIR>TLIST MYFILE

Owner:  MALA
        Name            Type            Rights
        --------        ------          ----------
        MALA            (User)          [ RW EMFA]
        ----
        SPARKYGROUP     (Group)         [ R     ]
        EVERYONE        (Group)         [ R     ]
```

First, MALA revokes the **A**ccess Control right to the file from herself. (This renders her incapable of re-granting any access rights to the file.)

```
Q:\MYDIR>REVOKE A FOR MYFILE FROM MALA
        Trustee's access rights set to [ RW  EMF ]
        Rights for 1 file(s) were changed for MALA.
```

If you run the NFSADMIN utility at the NetWare NFS server, the Get file info option will show the following for myfile:

```
Filename                    /userdata/mydir/myfile
        File Permission     644
        File owner              -3
        File group              sparkygroup
        Recursion                  No
```

Note: The Supervisor can issue a GRANT A FOR MYFILE TO MALA command and the owner of the file will revert to mala on the UNIX side.

Now, from a UNIX workstation, user "mala" sees the following output from the ls command:

```
%cd /mnt
        %ls -lg myfile
        -rw-r--r--   1   65533   staff   1414 Aug 12 00:17 myfile
```

Note: The superuser can issue a chown mala myfile command and the rights for MALA will revert to [RW EMFA] for MYFILE on the NetWare side.

Rule 2:

For a directory, **E**rase and **F**ile Scan rights need to be propagated to all the files below, but not to subdirectories since they have their own Erase and File Scan rights.

Rule 3:

Remember these important points regarding first-time creation of a file/directory.

- The Inherited Rights Mask for a newly-created file/directory is set to all 1's (inherit all rights - [SRWCEMFA]) if both the UID and GID of the file/directory are the same as its parent directory. Otherwise, only Erase, Modify, and File Scan bits are set in the Inherited Rights Mask ([EMF]).
- The Modify right is granted for a newly-created file if the user has write and search (w,x) permissions in the parent directory, or if the user already has the Modify right through inheritance from the parent directory or security equivalence.
- The Erase right is granted for a newly-created file/directory if the user has the "w" (write) permission in the parent directory on the UNIX side.
- The File Scan right is granted for a newly-created file if the user has read and search (r,x) permissions in the parent directory on the UNIX side.

Rule 4:

Keep in mind the general effect that subsequent mappings of NFS permissions to NetWare rights has (this is the effect of doing chmod type of operations on the UNIX side):

- For a file, the Supervisor, Create, Erase, Modify, and File Scan rights are preserved in the mapping.
- For a directory, the Supervisor, Read, Write, and Modify rights are preserved in the mapping.
- The Erase right is granted for a file or directory if the user has the "w" (write) permission in the parent directory on the UNIX side.
- The File Scan right is granted for a file if the user has read and search (r,x) permissions in the parent directory on the UNIX side.

Summary: Translating NFS Permissions to NetWare Rights

NetWare NFS follows two basic steps to translate NFS permissions to NetWare rights. These steps are summarized below.

Step 1 NetWare NFS deals with only UIDs, GIDs, NetWare users, and NetWare groups. The NFS user id, group id, and the class world are mapped to the corresponding NetWare user class (see Figure 11). Keep in mind the exceptions to this translation given in the previous sections.

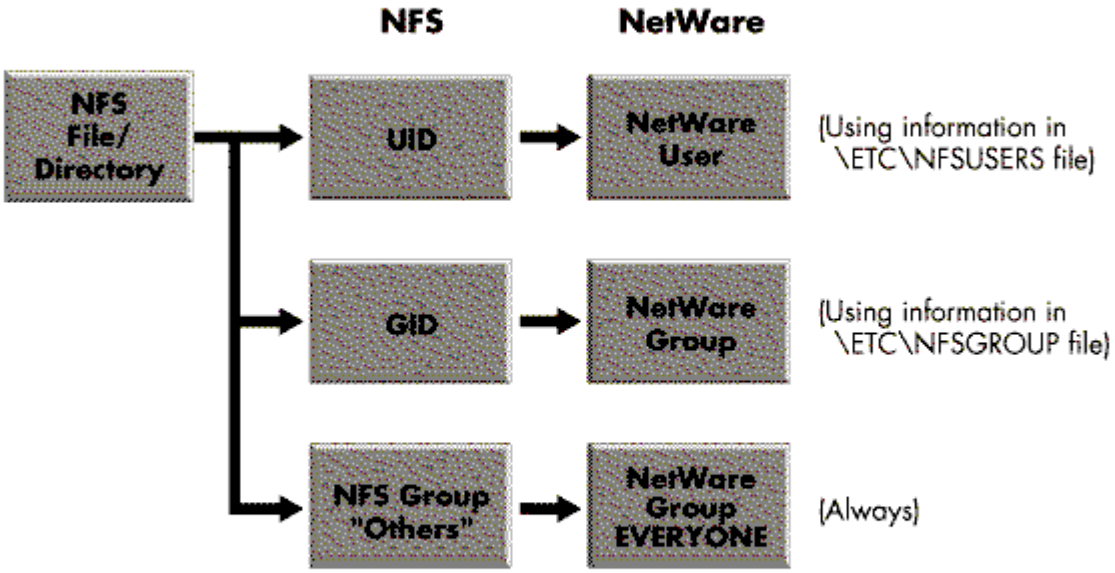


Figure 11: NetWare NFS first maps NFS UIDs and GIDs to corresponding NetWare users and groups.

Step 2 NFS permissions for directory/file owner, group owner, and world are mapped to NetWare trustee rights for the appropriate NetWare user and groups (see Figure 12).

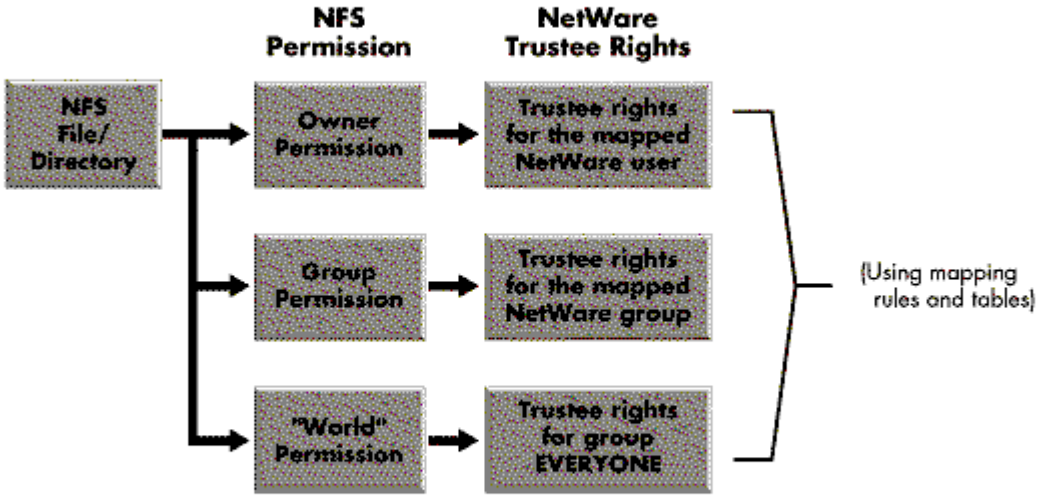


Figure 12: NFS permissions are mapped to NetWare rights according to the rules outlined in this AppNote.

Each time the NetWare NFS server goes down and comes back up, NFS permissions are recalculated (except the ones which are not mapped, like the x permission) for files and directories from the trustee rights assignment and NetWare file attributes in the DOS name space. This is done to ensure consistency with the latest changes that have been made to the file/directory's trustee assignments and attributes from other desktop platforms.

Recalculation of permissions is done on the fly when the file or directory is accessed from the UNIX client.

NetWare-to-NFS Mapping

When mapping NetWare rights to NFS permissions, the effective rights for the NetWare file or directory owner, NetWare group, and the NetWare group EVERYONE are translated to appropriate NFS permissions for user, group, and world respectively. (All other NetWare trustee assignments for the file or directory are discarded.) This translation occurs whenever one of the following takes place in a NetWare directory exported for access from NFS clients:

- A file or directory is created in DOS or imported from NetWare.
- A file or directory's access control is changed (by one of the NetWare desktop clients).

The NetWare effective rights for owner, group, and EVERYONE are converted to NFS permissions as

follows:

NetWare Effective Rights	--> NFS Permissions
owner	user
group	group
EVERYONE	others

The converted NFS permissions are stored in the directory entries of the NFS name space. Figure 13 shows how specific NetWare file and directory rights translate to NFS permissions.

Figure 13: Translation from NetWare rights to NFS access permissions.

File Type	NetWare Rights	Mapped NFS Permissions	Notes
File	R (Read)	r (read)	
	W (Write)	w (write)	
	C (Create)	Not meaningful for file	
	E (Erase)	Parent's write	*1
	F (File Scan)	Parent's read, execute, (search)	*1
	A (Access Control)	No direct match	*2
	M (Modify)	Not applicable	
	S (Supervisor)	Not applicable	
Directory	R (Read)	Not applicable	
	W (Write)	Not applicable	
	C (Create)	w (write) permission granted only if NetWare directory also has the Erase right	*3
	E (Erase)	w (write) permission granted only if NetWare directory also has the Create right	*3
	F (File Scan)	r (read), x (execute/search)	*4
	A (Access Control)	No direct match	
	M (Modify)	Not applicable	
	S (Supervisor)	Not applicable	

Notes:

*1 These apply to the parent directory under the following circumstances:

- If all the files and subdirectories have **E**rase rights and the parent directory has **C**reate and **E**rase rights, they are converted to the w (write) permission for the parent directory on the NFS side.

NetWare NFS

E (all files and subdirectories)
and --> w (for parent
C E (parent directory) directory)

- If all the files and subdirectories in a directory have **F**ile Scan rights, the **F**ile Scan right for the

parent directory is mapped to r (read) and x (execute) permissions on the NFS side.

NetWare		NFS
F (all files and subdirectories) --> F (parent directory)		r,x (for parent directory)

For a directory with a large number of files and subdirectories, these translations can be very time consuming.

*2 The **A**ccess Control right is a prerequisite for file ownership in NFS. These are mapped only when they are associated with the file owner.

- The NetWare owner of the file has the same rights on the NFS side as the NFS owner of the file.
- If the NetWare owner of the file does not have the **A**ccess Control right, the NetWare owner's ID is mapped to NFS UID= -3, which doesn't allow the permissions to be changed from the NFS side.

*3 If a directory has both **C**reate and **E**rase rights, and all its files and directories have **E**rase rights, they are mapped to the NFS w (write) permission for the directory. Otherwise either the **C**reate or **E**rase right for the directory will be dropped, as seen from NFS.

NetWare		NFS
E (all files and subdirectories) and C E (parent directory)	-->	w (for parent directory)

For an example of this, refer to Example 1 below.

*4 **F**ile Scan is mapped to the r and x permissions for the directory, if and only if all the files and subdirectories in the directory have the **F**ile Scan right.

NetWare		NFS
F (all files and subdirectories) F (for parent directory)	-->	r,x (for parent directory)

For an example of this, refer to Example 2 below.

Example 1. From a UNIX workstation, user mala uses the UNIX ls -ldg command to see what permissions she has to the /mnt directory.

```
%cd  /mnt
        %ls  -ldg  /mnt
         drwx------   2 mala     staff 512 Aug 14 01:05 /mnt
```

Now, from a DOS workstation, suppose user MALA revokes the **E**rase right from herself.

```
Q:\MYDIR>REVOKE E FROM MALA
        Trustee's access rights set to [   RWC   MFA]
```

Back at the UNIX workstation, user þmalaþ sees that she no longer has the w permission:

```
%cd  /mnt
        %ls  -ldg  /mnt
        dr-x------   2 mala     staff 512 Aug 14 01:07 /mnt
```

Example 2. At the UNIX workstation, user þmalaþ uses the UNIX ls -ldg command to see what permissions she has in the /mnt directory.

```
%cd  /mnt
        %ls  -ldg  /mnt
         drwx------   2 mala     staff 512 Aug 14 01:05 /mnt
```

Now, she uses the chmod command to change the permission mode for all files to 755 octal:

```
%chmod  755  *
        %ls  -lg
        -rwxr-xr-x   1 mala     staff   25 Aug 14 01:10 file1
        -rwxr-xr-x   1 mala     staff   20 Aug 14 01:15 file2
```

From a DOS workstation, user MALA has the following rights in the MYDIR directory.

```
Q:\MYDIR>TLIST

Name          Type          Rights
        --------      ------        ----------
        MALA          (User)        [ RWCEMFA]
        ----
        SPARKYGROUP     (Group)       [ RW    ]
        EVERYONE       (Group)       [ RW    ]
```

For FILE1 in that directory, MALA has the following rights:

```
Q:\MYDIR>TLIST FILE1

Owner:  MALA
        Name          Type          Rights
        --------      ------        ----------
        MALA          (User)        [ RW EMFA]
        ----
        SPARKYGROUP     (Group)       [ R     ]
        EVERYONE       (Group)       [ R     ]
```

Now suppose MALA revokes the **F**ile Scan right to FILE1 from herself.

```
Q:\MYDIR>REVOKE F FOR FILE1 FROM MALA
        Trustee's access rights set to [ RW EM A]
        Rights for 1 file(s) were changed for MALA.
```

Note: MALA cannot list "FILE1" after this due to lack of **F**ile Scan rights.

On the UNIX side, permissions r and x are removed for "mala" on /mnt because one of the files in this directory doesn't have **F**ile Scan rights on the NetWare side.

```
%ls  -ldg  /mnt
        d-w-------   2 mala     staff 512 Aug 14 01:17 /mnt
```

Permissions r and x can be added back for owner on /mnt by using chmod to change the permissions from 200 to 700.

```
%chmod 700 /mnt
        %ls  -ldg  /mnt
        drwx------   2 mala     staff 512 Aug 14 01:20 /mnt
```

At this point on the DOS workstation, the **F**ile Scan right is added back to "FILE1" since the parent directory has r,x permissions for the owner on the NFS side.

```
Q:\MYDIR>TLIST FILE1

Owner:  MALA
        Name            Type            Rights
        --------        ------          ----------
        MALA            (User)          [ RW EMFA]
        ----
        SPARKYGROUP     (Group)         [ R     ]
        EVERYONE        (Group)         [ R     ]
```

Summary: Translating NetWare Rights to NFS Permissions

NetWare NFS follows two steps to translate NetWare rights to NFS permissions. These steps are summarized below.

Step 1 The NetWare user classes are converted to their equivalent NFS classes (see Figure 14). Keep in mind the exceptions to the translation given in previous sections.

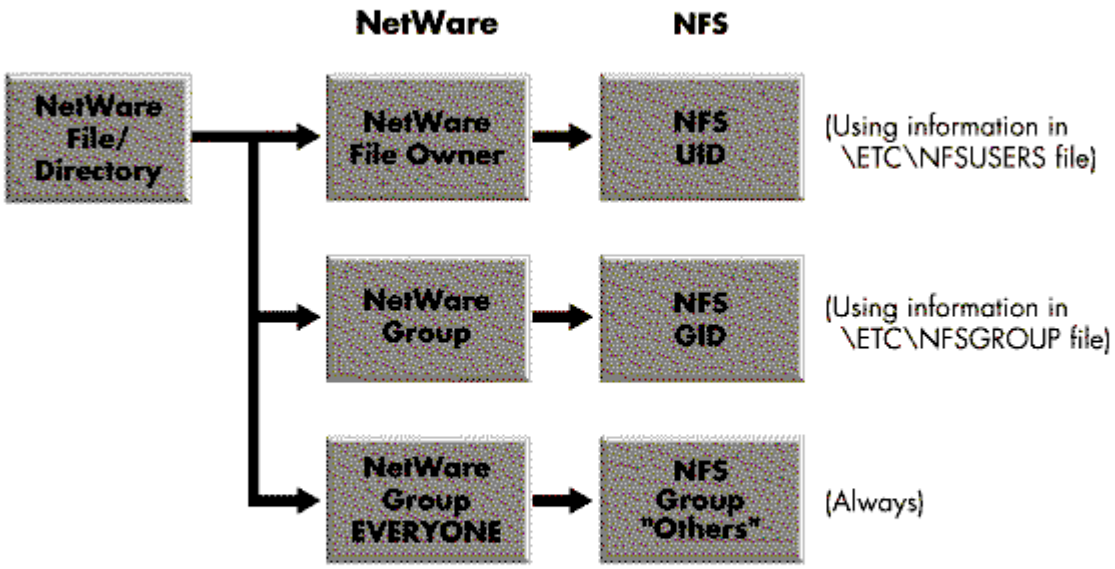


Figure 14: NetWare NFS first converts NetWare user classes to their equivalent NFS UIDs and GIDs.

Step 2 The NetWare effective rights of the owner, group, and the NetWare group EVERYONE for the file or directory are mapped to NFS permissions for the appropriate NFS user, group, and world (see Figure 15).

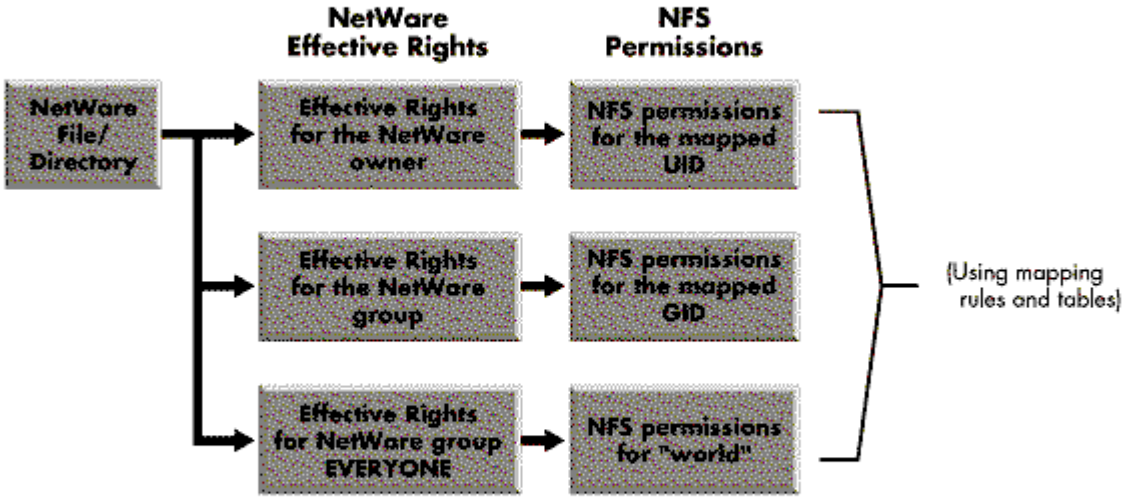


Figure 15: NetWare NFS maps NetWare effective rights to NFS permissions according to the rules given in this AppNote.

Exporting NetWare Directories to NFS Clients

NetWare file systems must be exported or placed in the EXPORTS table before they can be accessed from NFS clients. This can be done using the NFSADMIN utility at the NetWare NFS file server, or by editing the SYS:ETC\EXPORTS file manually.

Exporting subdirectories of paths that are already exported is not recommended. However, the ownership and permissions can be used later to control access to subdirectories of an exported path. For example, if /userdata is already exported as a volume, it is recommended that you not export /userdata/mala also. However, the supervisor can use SYSCON from a NetWare workstation to assign trustee rights and control access for different users (such as shown in the following subdirectories):

/userdata/mala
    /userdata/keith
    /userdata/milton
    /userdata/brian

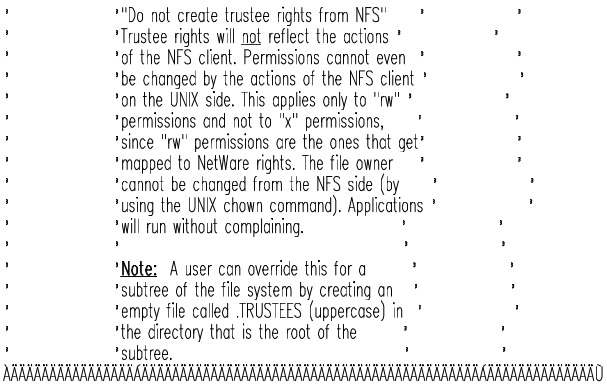
This can also be accomplished by setting permissions on the NFS side. Also, in a multi-workstation UNIX environment, the administrator can export the entire /userdata directory to all machines but only mount individual users' directories on their machines.

Exports File Options

There are a number of Exports file options for a file system, as listed in Figure 16.

Figure 16: Exports file options.

Option	Description	Example
Filename	This is the exported file system to which the options below apply. The root directory is usually exported with a slash before and after the volume name, as shown.	/userdata/
Trusted Hosts	This is a list of hosts that are allowed access to this directory. If this is left blank, all NFS clients can mount the directory.	sparcy, sparky
Root Access	This lists each host that is allowed to have root access to this path (supervisor privileges). If this is left blank, all client systems are denied root access.	sparcy
Read-Only Access	If this is set to Yes, the exported directory is available for reading only unless Read-Write access is explicitly set for the host. If set to No (default), all clients can read and write to this directory.	Yes
Anonymous Access	If this is set to Yes, it allows access to UNIX users whose UID is not found in the NFSUSERS database, by mapping the UID to NetWare user NOBODY. If set to No, unknown UIDs will be rejected.	Yes
Read-Write Access	This specifies hosts which can write to the directory if Read-Only Access has been set to Yes.	sparcy
DOS Attributes	This specifies whether or not NetWare file attributes are set through NFS. Do not modify attributes from NFS (Default) NetWare DOS file attributes will not be affected by changes in NFS permissions. But changes in NetWare file attributes will affect the NFS permissions, which might result in changes in ownership of the file/directory on the NFS client side. Modify attributes from NFS NetWare DOS file attributes will be affected as a result of changes in NFS permissions. Changes in NetWare file attributes will not result in changes to directory ownership or directory permissions.	Do not modify attributes from NFS
Trustee Rights	This specifies whether or not trustee rights are set through NFS. "Create trustee rights from NFS" (default) This causes trustee rights to emulate NFS permissions whenever the client executes the UNIX chmod command, or creates a file or performs any UNIX task that results in the assignment of permissions.	Create trustee rights from NFS



The following Exports file options result in changes in NFS permissions or NetWare rights:

- DOS Attributes
- Trustee Rights

To better understand these file options, let's examine three different scenarios of the exported file system.

Scenario 1. This scenario describes what happens with the default Exports file option settings.

DOS Attributes: Do not modify attributes from NFS (Default)
Trustee Rights: Create trustee rights from NFS (Default)

With the above options set for a file system, the NetWare NFS file server acts as a primary filestore to enable file sharing among all the NFS clients. This setup is used when file systems are manipulated from the UNIX side only, so that NFS permissions are preserved on the UNIX-client side without any kind of interpretation done at the NetWare NFS server.

- Assignment of NFS permissions will not affect NetWare DOS file attributes.
- NetWare rights will emulate NFS permissions and control access to the file system from other NetWare desktop clients.
- The following NetWare attributes:

  - **RO**, **DI**, **RI**, and **T** associated with a file

  - **DI** associated with a directory

  may cause undesirable effects on NFS permissions by revoking **w**rite permission to all users in the parent directory and changing the ownership of the parent directory (and the file in the case of RO attribute for a file) to UID -3.

This is done to ensure similar access control from the UNIX side so that the user can neither change permissions nor perform actions (such as delete/write a file) which are not allowed on the DOS side.

Scenario 2. This scenario describes what happens with the following Exports file option settings:

DOS Attributes: Do not modify attributes from NFS (Default)
Trustee Rights: Do not create trustee rights from NFS

In this setup, access control is truly NetWare-based because trustee rights inheritance from the parent directory on the NetWare side is preserved. It also minimizes creation of trustees (user, groups) as a result of file operations on the UNIX side, thereby making it more manageable.

Trustee rights will not reflect the actions of the NFS client when UNIX commands like chmod or chown are performed. Permissions cannot be changed for files and directories residing in the exported file system, even on the UNIX side. This is unnatural from a UNIX point of view.

This setup is always used in conjunction with an option to override the nocreate setting for a subtree of the file system, by creating an empty file called .TRUSTEES in the directory that is the root of the subtree:

- File operations performed (from an NFS client) above the subtrees with the .TRUSTEES file are still subject to the export option Do not create trustee rights from NFS and will not generate large amounts of bindery information.
- By the same token, users will still be able to perform UNIX operations (chmod, chown) in their subtrees with .TRUSTEES files and control access to their files from all the NetWare desktop clients.

Scenario 3. This scenario describes what happens with the following Exports file option settings:

DOS Attributes: Modify attributes from NFS
Trustee Rights: Create trustee rights from NFS (Default)

This setup is used when file systems are manipulated from the UNIX side. NetWare rights will emulate NFS permissions.

Changes in NFS permissions might affect NetWare DOS file attributes as well, thus playing a role in mapping. For example, the file attribute **R**ead-**O**nly will be set on the DOS side for the files with r-xr-xr-x (no write permission for user, group, and other) permission on the UNIX side.

NFS may override protections that are set as attributes on the DOS side. For instance, a file with the **DI** attribute set can be deleted from NFS and not from DOS. This could turn out to be a hole in security. File attribute settings on the DOS side don't have any effect on NFS permissions and the ownership of the directory.

Tips for Exporting File Systems

When exporting file systems, there are a few things you should remember or consider:

- Do not expose file systems to the entire world by giving root access to all the NFS clients/workstations. Instead, limit it to a known few.
- Understand the intended usage before setting the following options:
  - Do not create trustee rights from NFS option
  - Modify attributes from NFS option
- For better access control, export individual subtrees rather than the root of the file system.
- File systems containing commonly-used UNIX applications/ utilities like man pages, openwin demos, UNIX demos, games, answer book, and so on, can be exported to all clients with just Read-Only access. This saves disk space on individual NFS clients by having a single copy in the NetWare NFS file server.
- System administrator types can be given root access to the /etc directory (where all the NetWare NFS configuration files are stored) from their workstations, thus centralizing the system issues to NetWare NFS servers and user issues to UNIX boxes.

Examples: NFS File Permissions to NetWare Rights

This section summarizes the rules for mapping between NFS permissions and NetWare rights for a file. The examples that follow include trustee rights for the relevant user/groups only.

Rule 1

Subsequent mappings between NFS permissions and NetWare rights for a file have the following effects (these are effects of doing chmod-type operations on the UNIX side):

- The **S**upervisor, **C**reate, **E**rase, **M**odify, and **F**ile Scan rights for a file are preserved.
- The **E**rase right is granted for a file if the user has w (write) permission in the parent directory.
- The **F**ile Scan right is granted for a file if the user has r,x (read and search) permissions in the parent directory.

Rule 2

Only the file owner has the **A**ccess Control right on the file.

Rule 3

A change in the r or x permissions translates to changes in the **R** and **W** rights for the appropriate user or group. Changes in the x permissions on the UNIX side are not mapped for a file. The NFS permissions correspond to the NetWare rights as shown below:

NFS Permission removed/added	**NetWare rights removed/added**
r	R
w	W
x	no mapping

Example 1

This example shows the effect of a change in Unix file permissions for owner only, on NetWare trustee rights for a NetWare user mapped to the NFS UID for the file.

NFS UID	NetWare User
102	MALA

The NetWare rights for the directory USERDATA:\MYDIR at the start are:

User/Group	**NetWare rights**
MALA	[RWCE FA]
SPARKYGROUP	[R F]
EVERYONE	[ ]

These correspond to permissions rwxr-x--- (750 octal) on the UNIX side.

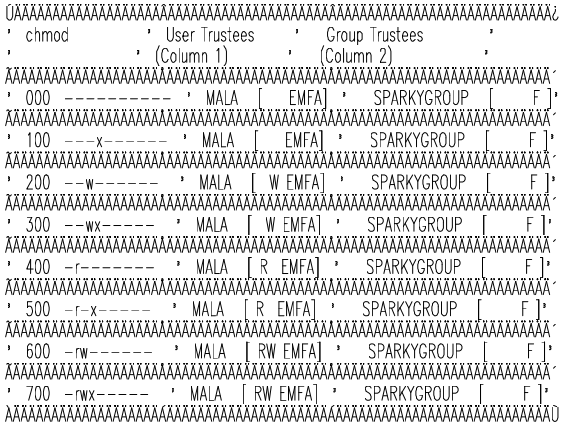
Note: /mnt is the mount point for /userdata/mydir which is exported with the default exports options of Do not modify attributes from NFS and Create rustee rights from NFS.

At a UNIX workstation, the following command is entered to change the NFS user permissions for the file (this can be done only by the file owner or root):

%chmod <xxx> **/mnt/file**

The effect of various values for xxx in the command are shown in Figure 17.

Figure 17: Effects of using the chmod command to change the NFS permissions for a file's user owner.



Column 1 above show how permissions rwx for owner "mala" on /mnt/file translate to NetWare rights, and that the **E**rase, **F**ile Scan, and **M**odify rights are preserved in the subsequent mappings.

NFS Permissions	**NetWare Rights**	
r,w	R W	
x	Not mapped	
Because "mala" has w permission in the parent directory	E	
Because "mala" has w,x permissions in the parent directory	M	
Because "mala" has r,x permissions in the parent directory		F
Because "mala" is the file owner		A

Column 2 shows that the **F**ile Scan right is granted to SPARKYGROUP for /USERDATA/MYDIR/FILE, since that group has r,x permissions in the parent directory. This is preserved in the subsequent mapping between NFS permissions and NetWare rights for the file.

Example 2

This example shows the effect of a change in UNIX file permissions for group owner only, on NetWare trustee rights for the NetWare group mapped to the NFS GID for the file.

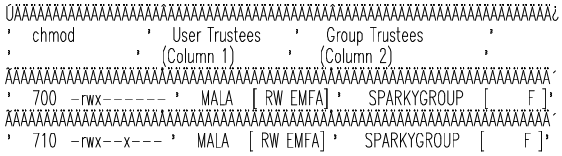
NFS GID	NetWare Group
10	SPARKYGROUP

At a UNIX workstation, the following command is entered to change the NFS group permissions:

%chmod <xxx> **/mnt/file**

The effect of various values for xxx in the command are shown in Figure 18.

Figure 18: Effects of using the chmod command to change NFS permissions for a file's group owner.



720	-rwx-w----	MALA [RW EMFA]	SPARKYGROUP [W F]
730	-rwx-wx--	MALA [RW EMFA]	SPARKYGROUP [W F]
740	-rwxr----	MALA [RW EMFA]	SPARKYGROUP [R F]
750	-rwxr-x--	MALA [RW EMFA]	SPARKYGROUP [R F]
760	-rwxrw----	MALA [RW EMFA]	SPARKYGROUP [RW F]
770	-rwxrwx--	MALA [RW EMFA]	SPARKYGROUP [RW F]

Example 3

This example shows the effect of a change in UNIX file permissions for group others only, on NetWare trustee rights for the NetWare group EVERYONE for the file.

At a UNIX workstation, the following command is entered to change the NFS world permissions:

%chmod <xxx> **/mnt/file**

The effect of various values for xxx in the command are shown in Figure 19.

Figure 19: Effects of using the chmod command to change NFS permissions for others group.

chmod	User Trustees (Column 1)	Group Trustees (Column 2)	
770	-rwxrwx---	MALA [RW EMFA]	EVERYONE [ ]
771	-rwxrwx--x	MALA [RW EMFA]	EVERYONE [ ]
772	-rwxrwx-w-	MALA [RW EMFA]	EVERYONE [W]
773	-rwxrwx-wx	MALA [RW EMFA]	EVERYONE [W]
774	-rwxrwxr--	MALA [RW EMFA]	EVERYONE [R]
775	-rwxrwxr-x	MALA [RW EMFA]	EVERYONE [R]
776	-rwxrwxrw-	MALA [RW EMFA]	EVERYONE [RW]
777	-rwxrwxrwx	MALA [RW EMFA]	EVERYONE [RW]

Examples: NFS Directory Permission to NetWare Rights

This section summarizes the rules for mapping between NFS permissions and NetWare rights for a directory. The examples that follow include trustee rights for the relevant user/groups only.

Rule 1

The effects of subsequent mapping between NFS permissions and NetWare rights for a directory (the effect of doing chmod-type of operations on the UNIX side) are:

- Once the **S**upervisor, **R**ead, **W**rite, and **M**odify trustee rights are assigned to the directory due to the effect of change in permissions on the UNIX side by previous operations, they are preserved in the mapping and are not removed in subsequent mapping between the two systems.

This is the reason why permissions --------- (000 octal) on the UNIX side on /mnt map to some rights on the NetWare side.

- The **E**rase right is granted for a directory if the user has the w (write) permission in the parent directory on the UNIX side.

Rule 2

Only the file owner has **A**ccess Control rights on the directory.

Rule 3

Removal of r or x permissions results in removal of **F**ile Scan rights for the directory. Removal of w permission results in removal of **C**reate and **E**rase rights for the directory for the appropriate user or group trustees.

NFS Permission removed	NetWare Rights removed
r or x	F
w	C E

Example 4

This example shows the effect of a change in UNIX permissions for group owner only, on NetWare trustee rights for the NetWare group mapped to the NFS GID for the directory:

NFS gid	NetWare Groupname
10	SPARKYGROUP

NetWare rights for the directory USERDATA:\MYDIR at the start are:

User/Group	**NetWare Rights**
MALA	[RWCE FA]
SPARKYGROUP	[RWCE F]
EVERYONE	[ ]

These translate to permissions rwxrwx--- (770 octal) on the UNIX side.

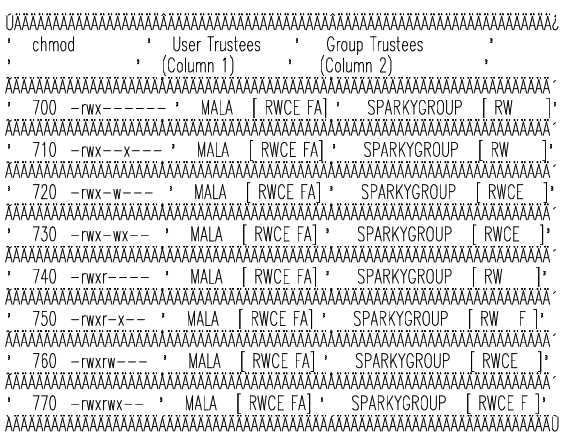
/mnt is the mount point for /userdata/mydir which is exported with the default exports option of Do not modify attributes from NFS and "Create trustee rights from NFS."

At a UNIX workstation, the following command is entered to change the NFS group permissions:

%chmod <xxx> /**mnt**

The effect of various values for xxx in the command are shown in Figure 20.

Figure 20: Effects of using the chmod command to change NFS permissions for a directory's group owner.



chmod	User Trustees (Column 1)	Group Trustees (Column 2)
700 -rwx------	MALA [RWCE FA]	SPARKYGROUP [RW]
710 -rwx--x---	MALA [RWCE FA]	SPARKYGROUP [RW]
720 -rwx-w---	MALA [RWCE FA]	SPARKYGROUP [RWCE]
730 -rwx-wx--	MALA [RWCE FA]	SPARKYGROUP [RWCE]
740 -rwxr----	MALA [RWCE FA]	SPARKYGROUP [RW]
750 -rwxr-x--	MALA [RWCE FA]	SPARKYGROUP [RW F]
760 -rwxrw---	MALA [RWCE FA]	SPARKYGROUP [RWCE]
770 -rwxrwx--	MALA [RWCE FA]	SPARKYGROUP [RWCE F]

Column 1 above shows how permissions "rwx" for owner "mala" on /mnt directory translate to NetWare rights.

NFS Permissions		**NetWare Rights**
r,x	-->	R W
w	-->	E C W
Because "mala" is the owner		A

Thus, MALA has [RWCE FA] rights for the /USERDATA/ MYDIR directory.

Column 2 shows that once SPARKYGROUP has the **R**ead and **W**rite trustee rights assigned for the directory, they are preserved in subsequent mapping between the two systems.

Trustee rights shown in column 2 of Figure 20 hold good for a change in permissions for the owner to values 000, 100, 200, 300 600, 700. The only difference is the addition of the **A**ccess Control right for the owner. The corresponding trustee right mappings are: [RWA], [RWA], [RWCEA], [RWCEA], [RWA], [RWFA], [RWCEA], and [RWCEFA].

Example 5

This example shows the effect of a change in UNIX permissions for group owner only, on NetWare trustee rights for the NetWare group mapped to the NFS GID for the directory.

NFS gid	NetWare Groupname
10	SPARKYGROUP

NetWare rights on directory USERDATA:\MYDIR at the start are:

User/Group	**NetWare Rights**
MALA	[RWCE FA]
SPARKYGROUP	[ ]
EVERYONE	[ ]

These translate to permissions rwx------ (700 octal) on the UNIX side.

At a UNIX workstation, the following command is entered to change the NFS group permissions:

%chmod <xxx> **/mnt**

The effect of various values for xxx in the command are shown in Figure 21.

Figure 21: Effects of using the chmod command to change NFS permissions for a directory's group owner.

chmod	User Trustees (Column 1)	Group Trustees (Column 2)
700 -rwx------	MALA [RWCE FA]	SPARKYGROUP [ ]
710 -rwx--x---	MALA [RWCE FA]	SPARKYGROUP [ ]
720 -rwx-w---	MALA [RWCE FA]	SPARKYGROUP [WCE]
730 -rwx-wx--	MALA [RWCE FA]	SPARKYGROUP [WCE]
740 -rwxr----	MALA [RWCE FA]	SPARKYGROUP [W]
750 -rwxr-x--	MALA [RWCE FA]	SPARKYGROUP [RW F]
760 -rwxrw---	MALA [RWCE FA]	SPARKYGROUP [RWCE]
770 -rwxrwx--	MALA [RWCE FA]	SPARKYGROUP [RWCE F]

Example 6

This example shows the effect of a change in UNIX permissions for group others only, on NetWare trustee rights for the NetWare group EVERYONE for the directory.

NetWare rights on the directory USERDATA:\MYDIR at the start are:

User/Group	**NetWare Rights**
MALA	[RWCE FA]
SPARCYGROUP	[RWCE F]
EVERYONE	[ ]

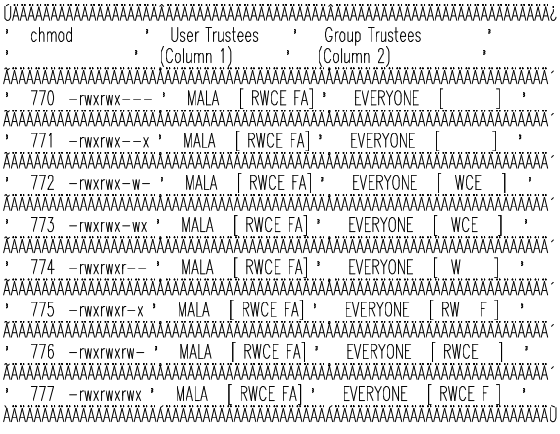
These translate to permissions rwxrwx--- (770 octal) on the UNIX side.

At a UNIX workstation, the following command is entered to change the NFS group permissions:

%chmod <xxx> **/mnt**

The effect of various values for xxx in the command are shown in Figure 22.

Figure 22: Effects of using the chmod command to change NFS permissions for a directory's group owner.



chmod	User Trustees (Column 1)	Group Trustees (Column 2)
770 -rwxrwx---	MALA [RWCE FA]	EVERYONE [ ]
771 -rwxrwx--x	MALA [RWCE FA]	EVERYONE [ ]
772 -rwxrwx-w-	MALA [RWCE FA]	EVERYONE [WCE]
773 -rwxrwx-wx	MALA [RWCE FA]	EVERYONE [WCE]
774 -rwxrwxr--	MALA [RWCE FA]	EVERYONE [W]
775 -rwxrwxr-x	MALA [RWCE FA]	EVERYONE [RW F]
776 -rwxrwxrw-	MALA [RWCE FA]	EVERYONE [RWCE]
777 -rwxrwxrwx	MALA [RWCE FA]	EVERYONE [RWCE F]

Example 7

This example shows the effect of a change in UNIX permissions for group others only, on NetWare trustee rights for the NetWare group EVERYONE.

NetWare rights on the directory USERDATA:\MYDIR at the start are:

User/Group	**NetWare rights**
MALA	[RWCE FA]
SPARCYGROUP	[RWCE F]
EVERYONE	[RWCE F]

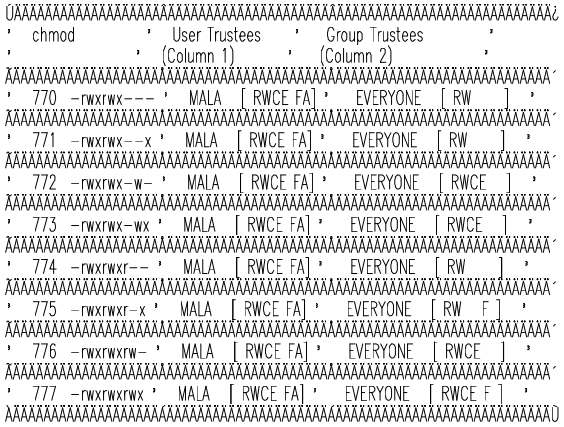
These translate to permissions rwxrwxrwx (777 octal) on the UNIX side.

At a UNIX workstation, the following command is entered to change the NFS group permissions:

%chmod <xxx> **/mnt**

The effect of various values for xxx in the command are shown in Figure 23.

Figure 23: Effects of using the chmod command to change NFS permissions for a directory's others group.



Questions and Answers

Q. How do I obtain UID/GID mappings to fill in the /etc/nfsusers and /etc/nfsgroup tables in NFSADMIN?

A. Most UNIX systems have UID/GID information in the following files:

/etc/passwd UIDs, GIDs (user account info)

/etc/group Group names, GIDs, members

If NIS (Network Information Service) is implemented in the network to ensure consistency of system administration information in a heterogeneous network environment, the NIS master for the NIS domain of interest will have the information that you would normally find in files like /etc/passwd, /etc/group, and so on. Read the UNIX system reference manual for the master server for information on how to retrieve ASCII information from YP maps (databases). The ypcat command is a widely supported command which dumps the contents of an NIS map.

Q. Why am I getting a UNIX UID ownership of -2 or -3 (6553x) at the mount point for files/directories?

A. The 16 bit representation of (-3) is 65533 and the 16 bit representation of (-2) is 65534.

UIDs of -2 and -3 are considered to be UID mapping exceptions during specific situations given by:

UID = -2

Superuser (root) on an NFS client machine will be mapped to UID -2 if the exported option for the file system does not allow remote root access for this client machine.

Unmapped NFS UID or NetWare username will be mapped to UID -2 (NOBODY).

UID = -3

If a file system is exported with the option Do not modify DOS attributes from NFS and if a file below it has the NetWare file attribute **DI** (Delete Inhibit) or **RI** (Rename Inhibit) or **T** (Transactional) associated with it, or if a directory below it has the NetWare file attributes **DI** (Delete Inhibit) associated with it, the owner of the parent directory will be mapped to UID -3.

If a file system is exported with the option Do not modify DOS attributes from NFS and if a file below it has the NetWare file attribute **RO** (Read only) associated with it, the owner of both the file and its parent directory will be mapped to UID -3.

If a volume is exported with the option Modify DOS attributes from NFS and if a file below it has the NetWare file attribute **T** (Transactional) associated with it, the owner of its parent directory will be mapped to UID -3.

If the owner of a file/directory does not have the "Access Control" right, the owner will be mapped to UID -3.

If a mapping results in NOBODY, and the default NetWare user NOBODY does not exist and cannot be created, the user will be mapped to UID -3.

Q. What do the default user name NOBODY and the group name NOGROUP mean?

A. NetWare user NOBODY (UID = -2) and NetWare group NOGROUP (GID = -2) are created automatically during NetWare NFS install.

The first step in the translation of NFS access permissions to NetWare rights (and vice versa) is the one-to-one mapping of NFS UID to NetWare username (using information in the /ETC/NFSUSERS file) and NFS GID to NetWare group name (using information in the /ETC/NFSGROUP file).

An unmapped NFS UID or NetWare username becomes UID -2 (NOBODY). Likewise unmapped NFS GID or NetWare group name becomes GID -2 (NOGROUP).

Superuser (root) on an NFS client machine will be mapped to UID -2 (NOBODY), if the exported option for the file system does not allow remote root access for this client machine.

Q. How does a file/directory created in DOS under an exported directory get its group assignment the very first time?

A. When a file/directory is created in DOS, it will have a file owner but there is no such thing as "group owner" in NetWare and many groups can be granted access to a file/directory. A unique group owner for each file/directory created is more UNIX like.

When a NetWare file/directory is imported to NFS from any of the NetWare Desktop clients, the tree is searched for a NFS group in the NFS name space for each directory, starting from the parent all the way up to the top. If an NFS group is found the file/directory inherits that. If an NFS group is not found, a GID of -2 (nogroup) is assigned.

If you want all the files and directories created to have a known group name (thus eliminating guesswork), perform the following at the root of the directory (the highest point in the NetWare directory structure) and avoid "NOGROUP" being assigned at any level:

chgrp -r mygroup mydir (recursive mode)

Q. Do the UID and GID have to be unique for users on various NFS clients in a network in order to access exported NetWare file systems?

A. Yes. Though a user may belong to several groups, he/she is supposed to have an unique UID and a primary group membership in a network. Name mapping between UID and NetWare username, and between GID and NetWare group name, is done only once for each unique UID or GID in the network, at the NetWare NFS server.

Q. Can I use tape backup to backup files from and NFS-mounted directory residing in a NetWare NFS file server and restore it using tar or cpio?

A. When UNIX utilities are used to backup files, only NFS file information is saved. When files are restored from tape, the information on Inherited Rights Mask, Trustee Rights, and DOS file attributes may get disrupted or lost and the entire file system may be opened up to everyone. Never use UNIX restore if permissions on the file system have to be maintained to control access.

This can be used as a safety measure besides backing up the file system using the normal NBACKUP utility. If all else fails, files can be restored using tar or cpio. However, the IRMs, Trustee Right Assignments, and NetWare attributes must be set manually to revert to the original security. This can turn out to be a very time consuming proposition.

Q. Can I export the /vol and /vol/subdir separately? Will this cause any problems?

A. The current version of NetWare NFS (1.2x) lets you do this. However, this might cause discrepancies in the permissions for the mounted directories. For instance, you may think that you have certain permissions in /vol/subdir but it may not be so, since it is also governed by the permissions you have on the (exported) parent directory, whether mounted or not.

Export the specific directory and have control over its permissions, rather than the root directory and try to figure out the permissions of its descendants.