EMC® VNX™ Series

Release 7.1

Managing a Multiprotocol Environment on VNX™

P/N 300-013-805 Rev A01

EMC Corporation
Corporate Headquarters:
Hopkinton, MA 01748-9103
1-508-435-1000
www.EMC.com

Published July 2012

Corporate Headquarters: Hopkinton, MA 01748-9103

Contents

Preface

As part of an effort to improve and enhance the performance and capabilities of its product lines, EMC periodically releases revisions of its hardware and software. Therefore, some functions described in this document may not be supported by all versions of the software or hardware currently in use. For the most up-to-date information on product features, refer to your product release notes.

If a product does not function properly or does not function as described in this document, please contact your EMC representative.

Special notice conventions

EMC uses the following conventions for special notices:

Note: Emphasizes content that is of exceptional importance or interest but does not relate to personal injury or business/data loss.

NOTICE Identifies content that warns of potential business or data loss.

CAUTION Indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.

WARNING Indicates a hazardous situation which, if not avoided, could result in death or serious injury.

DANGER Indicates a hazardous situation which, if not avoided, will result in death or serious injury.

Where to get help

EMC support, product, and licensing information can be obtained as follows:

Product information — For documentation, release notes, software updates, or for information about EMC products, licensing, and service, go to the EMC Online Support website (registration required) at http://Support.EMC.com.

Troubleshooting — Go to the EMC Online Support website. After logging in, locate the applicable Support by Product page.

Technical support — For technical support and service requests, go to EMC Customer Service on the EMC Online Support website. After logging in, locate the applicable Support by Product page, and choose either **Live Chat** or **Create a service request**. To open a service request through EMC Online Support, you must have a valid support agreement. Contact your EMC sales representative for details about obtaining a valid support agreement or with questions about your account.

Note: Do not request a specific support representative unless one has already been assigned to your particular system problem.

Your comments

Your suggestions will help us continue to improve the accuracy, organization, and overall quality of the user publications.

Please send your opinion of this document to:

techpubcomments@EMC.com

1 Introduction

In a UNIX environment, the Network File System (NFS) protocol is used to access file systems. In a Windows environment, the Common Internet File System (CIFS) protocol is used to access file systems. VNX supports a mixed NFS and CIFS environment by providing multiprotocol access capabilities such as access-checking policies and locking mechanisms. This multiprotocol access enables UNIX and Windows users to share the same file systems.

This document is part of the VNX documentation set and is intended for system administrators responsible for implementing VNX platform in their mixed Windows and UNIX environment.

Topics included are:

System requirements

For system requirements, see:

- *Configuring and Managing CIFS on VNX* for CIFS access requirements
- *Configuring NFS on VNX* for NFS access requirements

User interface choices

The EMC® VNX™ series offers flexibility in managing networked storage based on your support environment and interface preferences. This document describes how to configure VNX platform in a multiprotocol environment by using the command line interface (CLI). You can also perform some of these tasks by using one of the VNX management applications:

- EMC Unisphere™
- Microsoft Management Console (MMC) snap-ins
- Active Directory Users and Computers (ADUC) extensions

The following documents provide additional information about managing VNX:

- EMC VNX Documentation on EMC Online Support website
- Unisphere online help

The *Installing Management Applications on VNX for File* document includes instructions on launching Unisphere, and on installing the MMC snap-ins and the ADUC extensions.

The *VNX for File Release Notes* contain additional, late-breaking information about VNX management applications.

Related information

Specific information related to the features and functionality described in this document are included in:

- *Configuring and Managing CIFS on VNX*
- *Configuring NFS on VNX*
- *Configuring VNX User Mapping*
- *Using ntxmap for CIFS User Mapping on VNX*
- *Configuring VNX Naming Services*

- *Configuring Time Services on VNX*
- *Configuring Virtual Data Movers on VNX*
- *Installing Management Applications on VNX for File*
- *Using Windows Administrative Tools on VNX*
- *Using EMC Utilities for the CIFS Environment*
- *Managing Volumes and File Systems with VNX Automatic Volume Management*
- *Managing Volumes and File Systems for VNX Manually*
- *Using VNX Replicator*
- *Using International Character Sets on VNX for File*
- *VNX Glossary*
- *EMC VNX Command Line Interface Reference for File*
- *Parameters Guide for VNX for File*

Other related publications include:

- [RFC3530] *Network File System (NFS) Version 4 Protocol , April 2003*
- [RFC5661] *Network File System (NFS) Version 4 Minor Version 1 Protocol, January 2010*

EMC VNX documentation on the EMC Online Support website

The complete set of EMC VNX series customer publications is available on the EMC Online Support website. To search for technical documentation, go to http://Support.EMC.com. After logging in to the website, click the VNX Support by Product page to locate information for the specific feature required.

VNX wizards

Unisphere software provides wizards for performing setup and configuration tasks. The Unisphere online help provides more details on the wizards.

2 Concepts

Planning considerations on page 14 describes the tasks you need to complete prior to configuring features that enable UNIX and Windows users to share the same file systems.

In addition, you might need to understand some, if not all, of the following concepts when operating in a multiprotocol file sharing environment:

Planning considerations

Prior to configuring features that enable UNIX and Windows users to share the same file systems, you need to complete the following tasks:

- Complete the initial configuation of your CIFS environment, including configuring a CIFS server. *Configuring and Managing CIFS on VNX* explains how to configure a basic CIFS configuration on VNX by using the CLI. This initial environment can also be configured by using Unisphere.

  Configuring and Managing CIFS on VNX also contains advanced procedures that can be required after the initial configuration of CIFS on VNX and instructions for modifying and managing VNX in a Windows environment.

- Complete the initial configuration of your NFS environment. *Configuring NFS on VNX* and NFS Exports online help in Unisphere explain how to configure and manage NFS (versions 2, 3, and 4) on VNX.
- Configure a user mapping technique best suited to your mixed environment. *Configuring VNX User Mapping* provides a list of the tools and methods that can be used to map Windows users to UNIX-style user identifier (UID) and group identifiers (GIDs) and the tools that can be used to migrate users from a single-protocol environment to a multiprotocol environment.

CIFS user ID resolution

Every user of VNX, either Windows or UNIX, must be identified by a unique number UID and GID. Windows, however, does not use numeric IDs to identify users. Instead, strings called security identifiers (SIDs) are used.

Before configuring the Windows file-sharing service (referred to as CIFS) on VNX, a method of mapping Windows SIDs to UIDs and GIDs must be selected. The method used depends on whether there is a Windows-only or Windows and UNIX environment. These methods include:

- Active Directory (AD)
- Lightweight Directory Access Protocol (LDAP) Directory
- Local files
- Network Information Service (NIS)
- Usermapper (Internal or External)

EMC recommends use of Internal Usermapper in Windows-only environments. Internal Usermapper can also be used in multiprotocol environments with users having only Windows accounts.

Security on file system objects

In a multiprotocol environment, VNX uses its security policies to manage the access control of its file systems.

UNIX security model

UNIX access rights are referred to as the mode bits of a file system object. They are represented by a bit string in which each bit represents an access mode or privilege granted to the user owning the file, the group associated with the file system object, and all other users.

UNIX mode bits are represented as three sets of concatenated rwx (read, write, and execute) triplets for each category of users (user, or group, or other), as shown in Figure 1 on page 15.

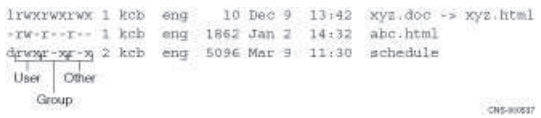


Figure 1. File system directory

The file system directory example illustrates the following:

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

Windows security model

The Windows security model is based primarily on per-object rights, which involve the use of a security descriptor (SD) and an access control list (ACL).

Access to a file system object is based on whether permissions have been set to Allow or Deny through the use of an security descriptor. The SD describes the owner of the object and group SIDs for the object along with its ACLs. An ACL is part of the security descriptor for each object. Each ACL contains access control entries (ACEs). Each ACE,

in turn, contains a single SID that identifies a user, group, or computer and a list of rights that are denied or allowed for that SID.

VNX supports ACLs at the share, directory, and file level.

User access control of file system objects

In a multiprotocol environment, VNX uses access policies to manage user access control of its file systems.

The access policy is specified by using the accesspolicy option on the server_mount command. Set the access-checking policy on page 38 describes this task.

User credentials and access checking

A Windows user credential is built and cached when a user first connects to VNX through the CIFS protocol. The credential contains the user SID and all the SIDs of the groups in which the user is a member. When using regular UNIX authentication (AUTH_SYS), a UNIX user credential is sent along with the Remote Procedure Calling (RPC) protocol request and consists of an UID and up to 16 GIDs to which the user belongs. In a Secure NFS environment, the UNIX user credential is built during the Kerberos authentication, and consists of the user's UID and GIDs of all the groups in which the user is a member. When a user requests access to a file system object, VNX compares the user credentials with the permissions on that file system object.

For an FTP user providing a domain and username, VNX contacts the domain controller for verification, and then builds a Windows credential. For an FTP user providing an unqualified username, VNX builds a UNIX-style credential based on the information in the local passwd and group files, NIS, or LDAP directory service.

VNX access-checking policies

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

Note: By default, when a file system is first created by using the Control Station, there is no ACL on the root of that file system. The UNIX owner is root and is the only one with write access to this file system. VNX automatically sets the ACL permissions as FULL CONTROL for EVERYONE on the root directory of the file system's CIFS share only after the first connection is made to this share.

Note: Ensure that Windows user accounts are not mapped to a UNIX root user. Access as the root user (UID=0) bypasses all privilege checks, which results in Windows users having full access to file system objects regardless of ACLs.

Access-checking policies only apply when the Data Mover's user authentication is set to the recommended default, NT. Table 1 on page 17 provides more information about access-checking policies.

Table 1. VNX access-checking policies

Access-checking policy	Description
Native (default)	◆ Access to a file or directory through NFS or UNIX authenticated FTP is granted only if the UNIX permissions on the file or directory allow it. ◆ Access through CIFS or Windows authenticated FTP is granted only if the Windows permissions on the file or directory allow it. ◆ ACL and UNIX permissions are maintained for every file and directory. ◆ A change in permissions on a file system object in NFS has no impact on permissions in CIFS and a change in permissions on a file system object in CIFS has no impact on permissions in NFS.

Table 1. VNX access-checking policies *(continued)*

Access-checking policy	Description
NT	◆ Access to a file or directory through NFS or UNIX authenticated FTP is granted only if the UNIX and Windows permissions allow it. ◆ Access through CIFS or Windows authenticated FTP is granted only if the Windows permissions allow it (the UNIX permissions do not have any effect). ◆ ACL and UNIX permissions are maintained for every file and directory. ◆ A change in permissions on a file system object in NFS has no impact on permissions in CIFS and a change in permissions on a file system object in CIFS has no impact on permissions in NFS.
UNIX	◆ Access to a file or directory through NFS or UNIX authenticated FTP is granted only if the UNIX permissions allow it (the Windows permissions do not have any effect). ◆ Access through CIFS or Windows authenticated FTP is granted only if the UNIX and Windows permissions on the file or directory allow it. ◆ ACL and UNIX permissions are maintained for every file and directory. ◆ A change in permissions on a file system object in NFS has no impact on permissions in CIFS and a change in permissions on a file system object in CIFS has no impact on permissions in NFS.
SECURE	◆ Provides the greatest security across CIFS and NFS. ◆ Access to a file or directory through either NFS or FTP or CIFS is granted only if the UNIX and Windows permissions on the file or directory allow it. ◆ ACL and UNIX permissions are maintained for every file and directory. ◆ A change in permissions on a file system object in NFS has no impact on permissions in CIFS and a change in permissions on a file system object in CIFS has no impact on permissions in NFS.

Table 1. VNX access-checking policies *(continued)*

Access-checking policy	Description
MIXED	◆ Access to a file or directory through either NFS or FTP or CIFS is always determined by the ACL. ◆ ACL and UNIX permissions are maintained for every file and directory. ◆ ACLs for files and directories are created from the protocol that last set or changed the permissions. For example, if an NFS client sets or changes permissions on a file or directory, the ACL is rebuilt based on the UNIX mode bits. If a CIFS client sets or changes permissions on a file or directory, the ACL is built based on the standard Windows permissions. ◆ In all cases, the ACL determines file and directory access regardless of whether the client is using the NFS, CIFS, or FTP protocol. ◆ ACL permissions are more granular than UNIX mode bits, consequently not all permissions set in an ACL can be translated to UNIX mode bits. In some cases, the mode bits might show more permissions than a user actually has. Using MIXED and MIXED_COMPAT on page 21 provides a detailed description.
MIXED_COMPAT	◆ Access to a file or directory through NFS or FTP or CIFS is determined by which protocol (NFS or CIFS) last set or modified the permissions. ◆ ACL and UNIX permissions are maintained for every file and directory. ◆ If the permissions of a file or directory are set or changed from a CIFS client, then access is determined by the ACL (EXPLICIT ACL). UNIX mode bits are generated based on the ACL but are not used for access checking. ◆ If the permissions of a file or directory are set or changed from a UNIX client, then UNIX mode bits dictate access. An ACL is still created but is not used for access checking. ◆ ACL permissions are more granular than UNIX mode bits, consequently not all permissions set in an ACL can be translated to UNIX mode bits. In some cases, the mode bits might show more permissions than a user actually has.

Selecting an access-checking policy

Figure 2 on page 20 helps you to determine the access-checking policy that is best for your environment.

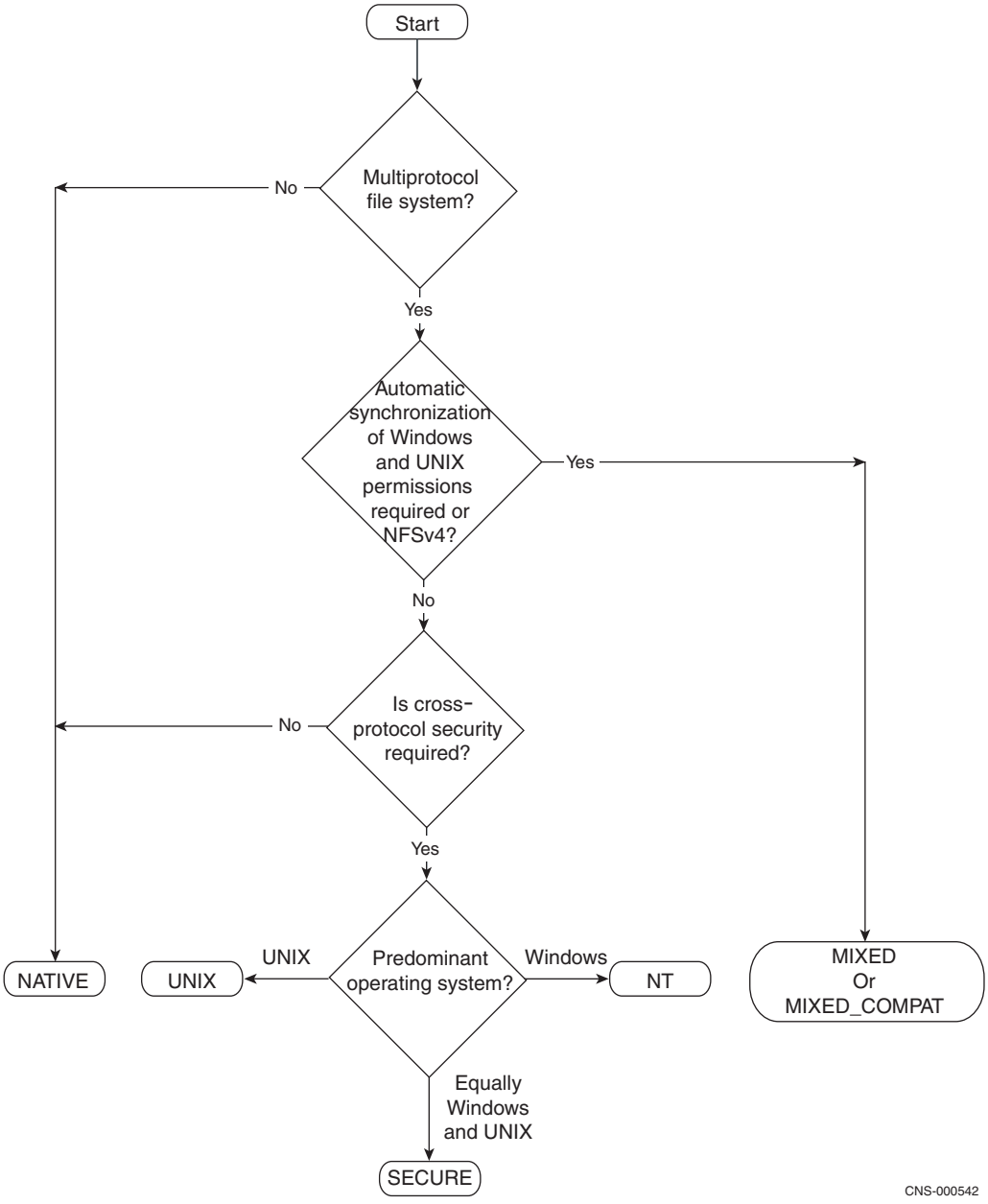


Figure 2. Decision tree for access-checking policies

Note: Automatic synchronization refers to the translation of UNIX mode bits into ACLs when setting permissions from an NFS client, and conversely means the translation of ACLs into UNIX mode bits on file system objects when setting permissions from a CIFS client.

Using MIXED and MIXED_COMPAT

UNIX and Windows handle access control in very different ways, making it difficult to set the same security on a file system object in a multiprotocol environment. VNX MIXED and MIXED_COMPAT policies synchronize UNIX and Windows permissions as closely as possible by using an algorithm that translates UNIX rights into ACL entries and ACL entries into UNIX rights.

The MIXED and MIXED_COMPAT policies differ in the way they translate a UNIX Group into an ACE and how they perform access checking. The MIXED policy always performs access checking against an ACL independent of the protocol accessing a file system object, as explained in the following example. The MIXED_COMPAT policy uses the permissions from the protocol that last set or changed the permissions on a file system object.

MIXED example

FileX is assigned the mode bits of rwxrw-r-. If the ACL of FileX is modified in such a way that user1, who is not the owner of the file, is granted read, write, and execute access to the file, the ACL shows the file owner has read, write, and execute access to the FileX. Members of the owner-group have read and write access, others have read access, and user1 has read, write, and execute access. However, the UNIX mode bits show rwxr-xrwx, meaning that at least one user who is not the file owner has read, write, and execute access. Although it seems that all others have full access to FileX, only user1 has full access because ACL is the one controlling access, not the UNIX mode bits.

Automatic synchronization

When the MIXED and MIXED_COMPAT policies are enabled for a file system object, the ACL and UNIX mode bits are automatically synchronized. Changes to an ACL result in modifications to the mode bits and changes to the mode bits reconstruct the ACL.

Table 2 on page 21 explains how the MIXED access-checking policy translates ACLS and UNIX mode bits during synchronization.

Table 2. MIXED access-checking policy

Translating UNIX mode bits into ACLs	Translating ACLs into UNIX mode bits
Creates ACL entries for File Owner, Group, and Everyone based on the mode bits of Owner, Group, and Other.	Translates the ACL Allow option but not the ACL Deny option.

Table 2. MIXED access-checking policy *(continued)*

Translating UNIX mode bits into ACLs	Translating ACLs into UNIX mode bits
Sets Delete or Change permissions and takes Ownership for the Owner but not for Everyone and other Groups.	Builds Owner mode bits from the Owner entry, the ACE of the file or directory, and the Everyone ACE.

Table 3 on page 22 explains how the MIXED_COMPAT access-checking policy translates Windows ACLs and UNIX mode bits during automatic synchronization.

Table 3. MIXED_COMPAT access-checking policy

Translating UNIX mode bits into ACLs	Translating ACLs into UNIX mode bits
Creates only two entries in the ACL: Owner and Everyone.	Translates the ACL Allow option but not the ACL Deny option.
Creates an Everyone ACE from the Group mode bits because groups are not translated with this policy.	Builds None, Owner, and Granted ACEs into Group and Other mode bits.
Ignores Other mode bits and does not use them to build the ACL.	
Sets Delete or Change permissions and takes Ownership for the File Owner but not for the Everyone Group.	

Mapping ACL permissions to UNIX mode bits

Table 4 on page 22 shows how the MIXED and MIXED_COMPAT access policies map the ACL file and directory permissions into UNIX file and directory rights.

Note: For MIXED and MIXED_COMPAT, ensure that the Windows user default group is set because VNX uses this group to decide which UNIX primary group to assign to a file system object created through CIFS.

Table 4. Translating ACL file and directory permissions into UNIX rights

Permissions	File permissions			Directory permissions		
	R	W	X	R	W	X
Traverse Folder/Execute File						

Table 4. Translating ACL file and directory permissions into UNIX rights *(continued)*

Permissions	File permissions			Directory permissions		
	R	W	X	R	W	X
Read Data	x		x			
Read Attributes	x			x		x
Read Extended Attributes	x			x		
Write Data		x				
Append Data		x				
Write Attributes		x			x	
Write Extended Attributes		x			x	
Delete		x			x	
Read Permissions	x					
List Folders				x		
Create Files					x	
Create Folders					x	
Delete Subfolders and Files					x	

Note: When VNX is used as an NFSv4 server, some NFSv4 clients might place a plus sign in the ls -l output when the file system object has an ACL. For example, rwxrw-r +.

Mapping UNIX mode bits to ACL permissions

Table 5 on page 23 shows how the MIXED and MIXED_COMPAT access-checking policies map UNIX mode bits into ACL permissions.

Table 5. Translating UNIX rights into an ACL

Permissions	R	W	X
Traverse Folder/Execute File			x
List Folder	x		x
Read Data	x		
Read Attributes	x		
Read Extended Attributes	x		

Table 5. Translating UNIX rights into an ACL *(continued)*

Permissions	R	W	X
Create Files/Write Data		x	
Create Folders/Append Data		x	
Write Attributes		x	
Write Extended Attributes		x	
Delete Subfolders and Files		x	
Delete			
Read Permissions	x	x	x
Change Permissions			
Take Ownership			

Inheritance rules

Table 6 on page 25 explains the inheritance rules for the NATIVE, UNIX, NT, and SECURE access-checking policies.

Note: The umask value is specified in octal and is XORed with the permissions of 666 for files and 777 for directories. Common values include 002, which gives complete access to the user or owner and group and read (and directory search) access to others, or 022 (default), which gives full access to the user or owner, and read (and directory search), but not write permissions to the group and others. To change the default umask value, use the parameter share.default.umask.

Table 6. NATIVE, UNIX, NT, and SECURE inheritance rules

Protocol	Rules
CIFS	When a CIFS client creates a file system object: ◆ The ACL is inherited from the parent directory, if one exists. ◆ The UNIX mode bits are determined by the umask set on the share. USe the server_export command to set the umask value.
NFS	When an NFS client creates a file system object: ◆ The ACL is inherited from the parent directory, if one exists. ◆ The UNIX mode bits are determined by the set for the user. NFS v4 clients might set the mode bits or ACL or both at file or directory creation time, overriding inheritance and umask.

Table 7 on page 25 explains the inheritance rules for the MIXED and MIXED_COMPAT access-checking policies.

Table 7. MIXED and MIXED_COMPAT inheritance rules

Protocol	Rules
CIFS	◆ When a CIFS client creates a file system object, if the inheritance flag is set and the object parent has an ACL, the file system object inherits the ACL, and the UNIX mode bit permissions are created based on the ACL translation. ◆ If the parent does not have an ACL, the UNIX permissions are set according to the umask value and an ACL is generated based on these values.

Table 7. MIXED and MIXED_COMPAT inheritance rules *(continued)*

Protocol	Rules
NFS	◆ UNIX mode bits are based on the umask value. ◆ An ACL is created from the UNIX mode bits. **Note:** NFS v4 clients might set the mode bits or ACL at file or directory creation time, overriding inheritance and umask.

Windows-style credential for UNIX users

In a multiprotocol environment, users often have both UNIX and Windows user identities. They can use either of these identities to access the data stored by VNX. The Windows credential feature enables VNX to take full account of a user's Windows group memberships when checking an ACL for access through NFS. When a UNIX user initiates a request for a file system object through NFS, VNX maps the UNIX UID to the Windows SID, and then merges the user's UNIX and Windows groups together to generate a Windows credential.

After the Windows credential is generated, it augments the UNIX credential supplied in the NFS RPC request for NFS access checking. The Windows credential provides:

- Consistent permissions on a file system object independent of the protocol accessing it.
- Cache to store Windows credentials, decreasing the access-checking process time.
- Management of access to data by using ACLs regardless of the protocol clients use.
- A new security.maxgroups parameter used to increase the number of UNIX groups for NFS clients from the default limit of 16 to 128.

The *Parameters Guide for VNX for File* provides more detailed information.

Using the Windows credential feature in a multiprotocol environment can be advantageous as it takes full account of the Windows group memberships of a user when checking an ACL through NFS.

Note: The Windows credential functionality closely resembles a Windows credential, but it does not support access to the AD and cannot query the AD database for information about Windows users and groups. UNIX users cannot access the nested, universal, and domain local group information from the local computer when UNIX credentials are translated to Windows credentials.

Generating a Windows credential

Figure 3 on page 27 illustrates how VNX generates a Windows credential after a UNIX client requests access to a file system object.

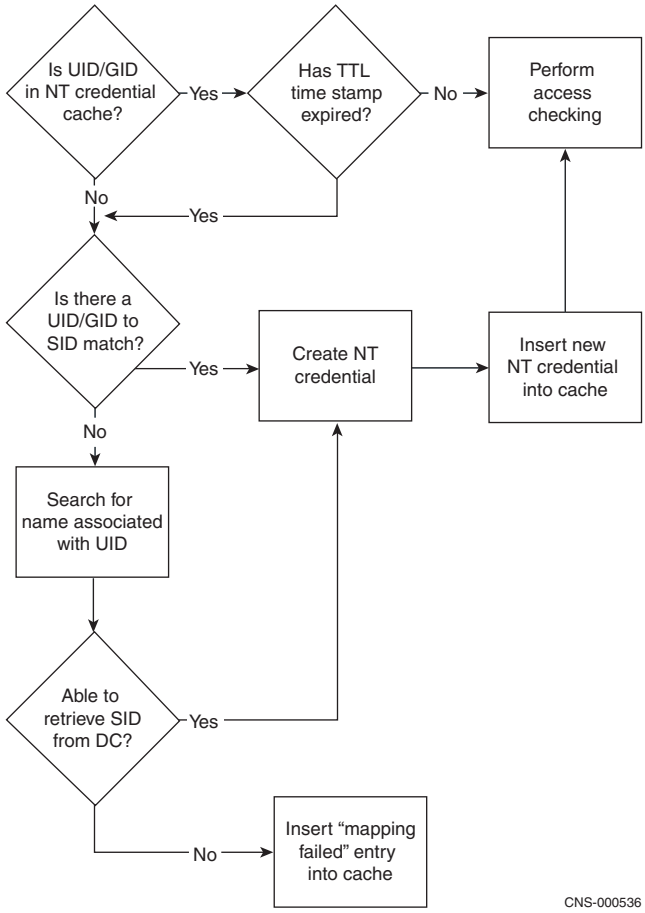


Figure 3. Decision tree for generating a Windows credential

Using Windows-style credential with Virtual Data Mover

VNX provides a multinaming domain solution for the Data Mover in the UNIX environment by implementing NFS server per Virtual Data Mover (VDM). This feature impacts the Windows-style credential for UNIX users. To support the NFS NTcredential domain for the NFS multiple domain feature, the following Windows Registry entry is used to specify the equivalent parameter per VDM value.

For a VDM, the 'nfs NTcred.winDomain' global parameter is replaced with the Windows Registry key.

```
[HKEY_LOCAL_MACHINE\Software\EMC\UnixNTCredential]
"NetbiosDomainName"="<domain_name>”
```

The *Parameters Guide for VNX for File* provides more information about the nfs.NTcred.winDomain parameter.

When the VDM name resolution is confined to the VDM, the nfs.NTcred feature uses the Windows domain specified in the VDM registry. Accordingly, a CIFS server member of this domain must be configured in the VDM and joined to the appropriate CIFS domain user to allow NFS access using a CIFS user credential.

Configuring Virtual Data Movers on VNX provides more information about the VDM for NFS feature.

Determining the GID for file system objects

In a file system, every object (such as a file, directory, link, and shortcut) has an associated owner and owner group that are identified by a UID and GID. NFS uses the UID and GID to control access to the file system object. Because a user can be a member of many groups, VNX needs a way to determine the group that should be associated with a newly created file. The user primary group setting determines which GID gets assigned to the file system object.

NFS and CIFS have the concept of a primary group for a user. In NFS, the primary group is required. However, the primary group is optional on Windows platforms and defaults to the Domain Users group.

All file system objects (FSOs) on a Data Mover have an associated owner (identified by a UID) and group (identified by a GID). Table 8 on page 28 describes how the primary group mapping is determined for file system objects.

Table 8. Determining the file system object GID

Protocol	Description
NFS	When an FSO is created from a UNIX client, the FSO GID is taken from the GID supplied by the UNIX client (based on the primary group of the creator).
CIFS	When an FSO is created from a Windows client, the GID can be determined in these ways: ◆ (Default) The file system object GID is taken from the GID associated with the primary group of the creator. ◆ The file system object GID is taken from the UNIX primary group of the user as defined in the passwd file, NIS, or AD.

Backing up and restoring file system objects

Use Network Data Management Protocol (NDMP) to back up and restore multiprotocol file systems, because network backups through NFS or CIFS capture only one set of metadata on a multiprotocol file system.

Note: For an NFS backup, only the UNIX rights are backed up and restored and determine the ACL permissions. As a result, the most complex ACEs in the ACLs might be lost during an NFS backup.

File naming

NFS and CIFS use different file naming conventions. Table 9 on page 29 explains these differences.

Table 9. NFS and CIFS file naming conventions

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

Note: When creating UID and GID names for Windows clients, Windows names (usernames, domain names, and global group names) must be written in lowercase in the NIS, the password files, and the group UNIX files. You need to be careful when doing explicit user and group mapping. The VNX does not recognize names like Windows.

File locking

File locking ensures file integrity when multiple users attempt to access the same file at the same time. File locks manage attempts to read, write, or lock a file that is in use by another user.

Table 10 on page 30 describes the different ways the NFS and CIFS protocols implement file locking features.

Table 10. File locking in multiprotocol environments

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

In a multiprotocol environment, a file might have locks set by CIFS and NFS users. Because NFSv2 and NFSv3 locks and CIFS or NFSv4 deny modes and oplocks—or delegations—are not directly equivalent, VNX must translate CIFS or NFSv4 deny modes and oplocks to NFSv2 and NFSv3 locks and translate NFSv2 or NFSv3 locks into CIFS or NFSv4 deny modes and oplocks. For example:

- A CIFS deny read/write mode request is translated into an NFSv2 or NFSv3 exclusive read/write lock.
- An NFSv2 or NFSv3 shared read lock is translated into a CIFS deny write mode.

To provide some control over the interaction of CIFS and NFS file locking, VNX provides three different locking policies that can be specified when mounting a file system. These file locking policies are used in a multiprotocol environment and indicate the impact of locking behavior on NFS clients in the case of concurrent NFS and CIFS file locking. Table 11 on page 31 explains the differences between file locking policies.

Table 11. VNX file locking policies

nolock[1]	wlock[2]	rwlock[3]
No locks: Treats all locks as advisory for NFS (v2 or v3) clients (default setting, least secure).	Write lock: Enforces CIFS or NFSv4 write locks for NFSv2 or NFSv3 client access.	Read/write lock: Enforces CIFS or NFSv4 read and write locks for NFSv2 or NFSv3 client access (most secure).
Lock requests: If a CIFS or NFS client locks a file, no other client can lock that file.	Lock requests: If a CIFS or NFS client locks a file, no other client can lock that file.	Lock requests: If a CIFS or NFS client locks a file, no other client can lock that file.
Access requests: ◆ CIFS clients ignore locks set by NFS. ◆ NFSv2 or NFSv3 clients can read and write to files locked by CIFS or NFSv4.	Access requests: ◆ CIFS clients denying concurrent access for write cannot open files locked by NFS for exclusive access. ◆ NFSv2 or NFSv3 clients can read, but cannot write to or delete, files locked by CIFS or NFSv4.	Access requests: ◆ CIFS clients denying concurrent access for read or write cannot open files locked by NFS for exclusive or shared access. ◆ NFSv2 or NFSv3 clients cannot read, write, or delete files locked by CIFS.

Note: VNX enforces file locks only when it is configured to do so and when the client application is using locks. Some simple applications, such as Windows Notepad or Wordpad, UNIX more, and vi do not use file locking. Files created with these applications are not locked. They can be opened and edited by another application even when a file system is mounted with wlock or rwlock.

Set the file locking policy on page 50 describes how to set locks on the file system.

Wide links

The wide links feature uses Microsoft DFS functionality to resolve UNIX absolute symbolic links for Windows clients. This is done by creating a DFS root on a CIFS share and then establishing a link on the DFS root that maps the UNIX mount point to the Windows server:\share\path. This mapping is done through the MMC DFS tool. It is difficult for a Windows client to open a path to a file system object when its path contains an absolute symbolic link. A Windows client requests the server to perform a function on a file system object based on a given path. Unlike Windows, a UNIX client uses a target path relative to its mount point. This can lead to a file system object on a remote server.

For purpose of example, assume that the UNIX client has the following two file systems mounted:

server1:/ufs1 mounted on /first

[3] NFSv2 or NFSv3 applications that do not expect lock conflicts (permission denied) on read/write operations might have problems.

[2] NFSv2 or NFSv3 applications that do not expect lock conflicts (permission denied) on read/write operations might have problems.

[1] Nolock is the only lock policy supported on Multi-Path File System (MPFS).

server2:/ufs2 mounted on /second

On ufs1, there is an absolute symbolic directory link to /second/home. A UNIX client can easily access this link from ufs1. However, as this path exists only on the UNIX client and not on the local server, a Windows client cannot follow this path.

By using wide links, Windows clients are able to access files from the same directory as UNIX clients when following an absolute symbolic link. VNX does this by using the DFS root functionality of the Data Mover. A wide link target must be on a DFS root, which is a CIFS share or a Windows share. This share can be local or on a remote server. As many wide links as required can be created on the root share.

Wide link translation makes it easier to build and maintain a single, multiprotocol file system namespace that spans multiple file servers as it reduces the burden associated with maintaining consistency between the NFS and CIFS namespace structure definitions (for example, NFS auto mount table and DFS). Create wide links on page 53 provides procedural information for creating wide links.

Before creating wide links consider:

- Wide links are based on Microsoft DFS functionality. The DFS requirements must be met, as explained in Distributed File System server on page 34, before establishing wide links.
- A wide link target must be on a DFS root, which is a CIFS share or a Windows share. This share can be local or on a remote server. As many wide links as required can be created on the root share.
- As DFS redirects only directories, a directory pathname must be set in the DFS link used for wide links. During the redirection of wide links, the system:
  - Finds a DFS link matching the beginning of the target path in the symbolic link.
  - Appends the rest of this path to the DFS target for final redirection.
- A wide link can be configured on a per Virtual Data Mover (VDM) basis. This enables a Windows client to be directed to as many different directory locations as needed.
- After the wide links are configured, a symbolic link with an absolute path appears as a directory instead of a file in Windows Explorer.
- The path in the DFS link must be the same in Windows and UNIX (in other words, the UNIX name of each component must be the M256 name in Windows).
- On an NT4 client, the Security tab does not appear in the Properties dialog box for a file that is located in a share that supports wide links. The security can be set by using a security tool such as calcs. Alternatively, manage the ACLs of files by using either files of a Windows 2000 client or shares that do not support wide symbolic links. There can be two different shares on the same directory, one that supports wide links and one that does not, and the share that does not support wide links can be used to manage security setting by using NT4 clients.
- If a Windows client is connected to CIFS share on a DFS root and this share is removed from DFS, the client might not be able to access it. Because the wide links feature is based on Microsoft DFS, this can also happen with wide links. This behavior occurs

because the clients use a DFS cache to track all DFS links. Until the share's cache entry times out, the Windows clients attempt to access the deleted share even though it no longer exists in DFS.

To resolve this:

- Wait for the DFS cache entry to time out.
- Or, disconnect the client from the share, clear the client's DFS cache by using the Microsoft command line tool dfsutil/pktflush, and reconnect to the share.

Note: Wide links cannot be used in conjunction with symbolic links containing absolute paths. The value of the parameter shadow.followabsolutpath must be 0 for wide links support to be enabled. If symbolic links using paths from the root of the Data Mover are needed, then the linkscan must be emulated by using wide links. To do this, create DFS links that emulate the directory structure from the root of the Data Mover. Distributed File System server on page 34 provides more information.

Process steps

The following steps describe how a Windows client processes the wide links feature by using DFS:

1. Client opens a path that has an absolute symbolic link.
2. Server detects an absolute link path and sends an error stating this path is not covered. This is typical DFS behavior.
3. Client requests DFS referrals of this path to determine where to connect next.
4. From the DFS root on the Data Mover defined as the widelink database in the Windows Registry of the Data Mover, the CIFS server finds a link that matches the beginning of the target path in the symbolic link and determines the CIFS share to use for wide links resolution.
5. CIFS server sends DFS referrals pointing the client to the new path.

Establishing wide links

In the following example, the w1_root-1 file system contains the user1 directory that has the symbolic links:

- link to fs_wslink-1\user1 on the local Data Mover
- link to fs_wlink-29\user1 on a remote Data Mover

Example

w1_root-1 file system exists on server_2:

```
$ server_export server_2 | grep -w "wl_root-1"
export "/wl_root-1"
share "wl_root-1" "/wl_root-1" maxusr=4294967295 umask=22
```

user1 directory is located in w1_root-1:

```
[user1@LINUX1PAG01 user1]$ pwd
/wl_root-1/user1
```

user1 has two UNIX symbolic links to other directories on separate Data Movers:

```
[user1@LINUX1PAG01 user1]$ ls -lhat
total 8.0K
drwxr-xr-x 3 root root 0 Feb 2 13:19 ..
drwxr-xr-x 3 user1 group-1001 1.0K Feb 2 12:43 .
-rw-r--r-- 1 user1 group-1001 0 Feb 2 12:43 NFS_user_file
lrwxrwxrwx 1 user1 group-1001 15 Feb 2 12:25
user1_on_fs_wlink-29 -> /wlink-29/user1
lrwxrwxrwx 1 user1 group-1001 14 Feb 2 12:25
user1_on_fs_wlink-1 -> /wlink-1/user1
drwxr-xr-x 2 user1 group-1001 80 Feb 1 17:49 user1
```

These symbolic links point to:

- user1 on fs_wlink-1 on the local Data Mover (server_2):

```
[user1@LINUX1PAG01 user1]$ mount | grep wlink-1
automount(pid26562) on /wlink-1 type autofs
(rw,fd=5,pgrp=26562,minproto=2,maxproto=3)
dm2-ana0-1-sa:/wlink-1/user1 on /wlink-1/user1 type nfs
(rw,addr=172.24.100.50)
```

- user1 on fs_wlink-29 on a remote Data Mover (server_3):

```
[user1@LINUX1PAG01 user1_on_fs_wlink-29]$ mount | grep wlink-29
automount(pid26592) on /wlink-29 type autofs
(rw,fd=5,pgrp=26592,minproto=2,maxproto=3)
vdm3-ana0-6-sa:/root_vdm_3/wlink-29/user1 on /wlink-29/user1 type nfs
(rw,addr=172.24.100.58)
```

From Windows, the symbolic links in user1 display as files:

```
C:\>dir \\dm2-ana0-1-sa\wl_root-1\user1
Volume in drive \\dm2-ana0-1-sa\wl_root-1 is 102
Volume Serial Number is 0000-0014
Directory of \\dm2-ana0-1-sa\wl_root-1\user1
02/02/2005 12:43 PM <DIR> .
02/02/2005 12:23 PM <DIR> ..
02/01/2005 05:49 PM <DIR> user1
02/02/2005 12:25 PM 14 user1_on_fs_wlink-1
02/02/2005 12:25 PM 15 user1_on_fs_wlink-29
02/02/2005 12:43 PM 0 NFS_user_file
3 File(s) 29 bytes
3 Dir(s) 52,867,235,840 bytes free
```

Distributed File System server

Microsoft Distributed File System (DFS) allows you to group shared folders located on different servers into a logical DFS namespace. A DFS namespace is a virtual view of these shared folders shown in a directory tree structure. By using DFS, you can group shared

folders into a logical DFS namespace and make folders that are distributed across multiple servers appear to users as if they reside in one place on the network. Users can navigate through the namespace without needing to know server names or the actual shared folders hosting the data.

Each DFS tree structure has a root target, which is the host server running the DFS service and hosting the namespace. A DFS root contains DFS links that point to the shared folders—a share and any directory below it—on the network. The shared folders are referred to as DFS targets.

Microsoft offers stand-alone and domain-based DFS root servers: the domain DFS root server and the stand-alone DFS root server. The domain-based DFS server stores the DFS hierarchy in the AD. The stand-alone DFS root server stores the DFS hierarchy locally. VNX provides the same functionality as a Windows 2000 or Windows Server 2003 stand-alone DFS root server.

The Microsoft website at http://www.microsoft.com/windowsserversystem/dfs/default.mspx provides detailed information about DFS. Configure and administer DFS support on page 51 provides procedural information for creating a DFS root.

3 Managing

The tasks to manage multiprotocol environments are:

Set the access-checking policy

User access control of file system objects on page 16 provides conceptual information about security models and access-checking policies.

Action
To set the access-checking policy for a file system, use this command syntax: `$ server_mount <movername> -option accesspolicy={NT|UNIX|SECURE|NATIVE|MIXED|MIXED_COMPAT}<fs_name> <mountpoint>` where: `<movername>` = name of the Data Mover or VDM `<fs_name>` = name of the file system being mounted `<mountpoint>` = name of the mount point **Note:** Always verify the current access-checking policy on the file system before executing this command. The default policy is NATIVE. Example: To set the access-checking policy to NT for file system ufs1 on server_2, type: `$ server_mount server_2 -option accesspolicy=NT ufs1 /ufs1`
Output
`server_2 : done`

Migrate access_checking policy to MIXED and MIXED_COMPAT

To migrate the access-checking policy to MIXED and MIXED_COMPAT, you must perform the following tasks:

- Synchronize Windows and UNIX permissions on page 39
- Reset the access policy on page 39
- Check the translation status on page 40

Synchronize Windows and UNIX permissions

Note: Because the synchronization task cannot be undone, first perform a backup of the file system. Always check the access-checking policy set on the file system before and after executing the translate command. The file system must be mounted as MIXED or MIXED_COMPAT before executing this command. If not, the command is refused. The file system to be translated must be a UXFS file system object mounted as read/write.

After remounting a file system object to MIXED or MIXED_COMPAT, perform the following steps to synchronize Windows and UNIX permissions.

Action
To synchronize Windows and UNIX permissions on the file system, use this command syntax: **$ nas_fs -translate** *<fs_name>* **-access_policy start -to {MIXED} -from {NT|NATIVE|UNIX|SECURE}** where: *fs_name* = name of the file system Example: To synchronize Windows and UNIX permissions for ufs1 on server_2 and regenerate ACLs based on UNIX modes, type: **$ nas_fs -translate ufs1 access_policy start -to MIXED -from UNIX**
Output
server_2 : done

Note: Using MIXED and MIXED_COMPAT on page 21 explains how Windows and UNIX permissions are translated to MIXED or MIXED_COMPAT from an NT, NATIVE, UNIX, or SECURE originating policy.

Reset the access policy

You can remount a file system to reset the access-checking of the file system object to its originating policy. This action applies the new access right policy and causes the ACLs and mode bits to become independent when first modified. ACL permissions and the UNIX mode bits remain unchanged.

Note: File systems might have permissions that are not synchronized. Synchronize Windows and UNIX permissions on page 39 provides more information.

Action
To reset the MIXED or MIXED_COMPAT access-checking policy for a file system, use this command syntax: `$ server_mount <movername> -option accesspolicy={NT|UNIX|SECURE|NATIVE|MIXED|MIXED_COMPAT} <fs_name><mount_point>` where: *<movername>* = name of the Data Mover *<fs_name>* = name of the file system being mounted *<mount_point>* = name of the mount point, which begins with a forward slash (/) Example: To reset the access-checking policy to UNIX for file system ufs1 on server_2, type: `$ server_mount server_2 -option accesspolicy=UNIX ufs1 /ufs1`
Output
`server_2: done`

Check the translation status

Action	
To check the translation status of a file system, use this command syntax: `$ nas_fs -translate <fs_name> -access_policy status` where: *<fs_name>* = name of the file system being translated Example: To check the translation status for ufs1, type: `$ nas_fs -translate ufs1 -a status`	
Output	**Notes**
`status=In progress` `percent_inode_scanned=68` `1097154093: ADMIN: 4: Command succeeded: acl database=/ufs1 convertAccessPolicy status`	◆ If the translation failed, check if the file system is mounted as MIXED or MIXED_COMPAT. ◆ If the translation does not complete due to system failure, run the command again.

Manage a Windows credential

The tasks to manage a Windows credential are:

- Generate Windows credentials on page 41
- Include UNIX groups in a Windows credential on page 42
- Modify Windows credential settings on page 42
- Set the Windows default domain on page 43
- Define the Windows credential cache on page 43
- Set the time-to-live expiration stamp on page 44

Windows-style credential for UNIX users on page 26 provides conceptual information.

Generate Windows credentials

Action
To generate Windows credentials for a file system object, use this command syntax: `$ server_mount <movername> -option accesspolicy={NT|UNIX|SECURE|NATIVE|MIXED|MIXED_COMPAT},ntcredential <fs_name><mount_point>` where: *<movername>* = name of the Data Mover or VDM *<fs_name>* = name of the file system being mounted *<mount_point>* = name of the mount point **Note:** The Windows credential function is for multiprotocol file systems. Use this feature only with NT, SECURE, MIXED, and MIXED_COMPAT access-checking policies. Example: To set the access-checking policy and generate the Windows credential for file system ufs1 on server_2, type: `$ server_mount server_2 -option accesspolicy=NT,ntcredential ufs1 /ufs1`
Output
`server_2: done`

Include UNIX groups in a Windows credential

EMC recommends setting the acl.extendExtraGid parameter if you use credentials. When the user accesses VNX through CIFS, VNX can be configured to include the users' UNIX groups in their Windows credential. This is in addition to their Windows groups. VNX will include user's UNIX groups in their Windows credential if the server parameter cifs acl.extendExtraGid is set to 1. There is no limit to the number of groups a Windows credential can contain.

Note: The acl.extendExtraGid parameter applies only in multiprotocol environments with a Network Information Service (NIS) or .etc/group file on the Data Mover. The UNIX groups are retrieved from the UNIX name services configured on the Data Mover—for example, local group file, NIS, LDAP and so on—by using the username without the .domain extension.

Action
To include users' UNIX group in their Windows credential, use this command syntax: `$ server_param <movername> -facility cifs -modify acl.extendExtraGID -value <new_value>` where: *<movername>* = name of the Data Mover or VDM *<new_value>* = 1 (to enable mapping) or 0 (to disable mapping) Example: To merge the users' UNIX and Windows groups together to build a Windows credential, type: `$ server_param server_2 -facility cifs -modify acl.extendExtraGid -value 1`
Output
`server_2: done`

Modify Windows credential settings

For Windows 2000, access to a trusted domain requires setting additional rights for the CIFS server retrieving a list of groups to which a user belongs. This server must be granted the List contents and Read all properties rights.

Perform the following steps to set rights for the CIFS server:

1. Use the Microsoft AD User and Computer MMC in expert mode.
2. From the menu, select **View ➤ Advanced** features.
3. Right-click the domain name, and select **Security ➤ Advanced**.

4. Grant rights:
   - For a Data Mover NetBIOS name: Everyone or Anonymous
   - For a Data Mover computer name: serverDomain\Domain Computers

Set the Windows default domain

The default Windows domain name is used if several different SIDs match the user UNIX UID, or the UID-to-name reverse mapping returns an ambiguous username (no domain).

The nfs NTcred.winDomain parameter specifies the Data Mover default Windows NetBIOS domain name to be used for NFS users accessing a file system where the ntcredential option has been used.

Note: Parameter and facility names are case-sensitive. NTcred.winDomain can be used with NFSv2, NFSv3, and NFSv4.

Action
To set the Windows default domain, use this command syntax: `$ server_param <movername> -facility nfs -modify NTcred.winDomain -value <new_value>` where: `<movername>` = name of the Data Mover `<new_value>` = a valid NetBIOS domain name Example: To set the Windows default domain to nasdocs.emc.com, type: `$ server_param server_2 -facility nfs -modify NTcred.winDomain -value nasdocs.emc.com`
Output
`server_2: done`

Define the Windows credential cache

The credential cache is a size-limited cache containing Windows credentials and any UID entries that could not be mapped to SIDs.

Note: If the CIFS service is stopped, connected users can continue to use the cache for 20 minutes. When CIFS is restarted, all the failed mapped entries are removed from the cache.

Action
To set the Windows credential cache size, use this command syntax: `$ server_param <movername> -facility nfs -modify NTcred.size -value <new_value>` where: *<movername>* = name of the Data Mover or VDM *<new_value>* = the maximum number of entries in the cache. The default is 1009. Example: To set the Windows credential cache size to 1000, type: `$ server_param server_2 -facility nfs -modify NTcred.size -value 1000`
Output
`server_2: done`

Set the time-to-live expiration stamp

Each time entry has a time-to-live expiration stamp.

Note: Parameter and facility names are case-sensitive. NTcred.TTL parameter can be used with NFSv2, NFSv3, and NFSv4.

Action
To set the time-to-live expiration stamp of the Windows entry in the credential cache size, use this command syntax: `$ server_param <movername> -facility nfs -modify NTcred.TTL -value <new_value>` where: *<movername>* = name of the Data Mover or VDM *<new_value>* = the number of minutes. The default is 20 minutes. Example: To set the time-to-live expiration stamp of the Windows credential to 30 minutes, type: `$ server_param server_2 -facility nfs -modify NTcred.TTL -value 30`
Output
`server_2: done`

Use only UNIX permissions for access checking

User access control of file system objects on page 16 provides conceptual information.

Note: Setting the cifs acl.unixCheckAcl value to 0 (zero) alters the behavior of the UNIX file system access policy so that only the UNIX mode bits on directories and files are used to determine user access to files—regardless of the protocol they use for accessing the file system. The file system still stores any ACL set, but it does not affect user access rights.

Note: When you use the acl.unixCheckAcl parameter, you might also consider setting bit 1 of the server parameter cifs acl.extacl value to 2. Doing so exposes the UNIX mode bits on directories and files as additional ACEs in the ACLs of directories and files. Manage UNIX permissions from a Windows client on page 46 provides procedural information for this task.

Action
To specify that only UNIX permissions are checked when the file system access policy is set to UNIX, use this command syntax: `$ server_param <movername> -facility cifs -modify acl.unixCheckAcl -value <new_value>` where: *<movername>* = name of the Data Mover *<new_value>* = 0 to dismiss ACL checking; 1 to enforce ACL checking **Note:** Parameter and facility names are case-sensitive. The cifs acl.unixCheckAcl parameter affects only those file systems on a Data Mover that are configured to use the UNIX access policy. Example: To ensure that only UNIX permissions are checked when the file system access policy is set to UNIX, type: `$ server_param server_2 -facility cifs -modify acl.unixCheckAcl -value 0`
Output
`server_2 : done`

Manage UNIX permissions from a Windows client

Action
To enable a specific capability for ACL management, use this command syntax: `$ server_param <movername> -facility cifs -modify acl.extacl -value <new_value>` where: `<movername>` = name of the Data Mover `<new_value>` = The bit list that enables special capabilities for access control list management. The bit list consists of seven binary bits (0 to 6). Any combination of bits is allowed. Each bit is 1 when set, otherwise 0. Bit 0 set (0000001 or +1) = UNIX metadata associated with files and directories is presented to CIFS backup client. Bit 1 set (0000010 or +2) = Windows clients can view and modify UNIX permissions. Bit 2 set (0000100 or +4) = CIFS network back up applications can backup and restore UNIX file and directory security attributes. Bit 3 set (0001000 or +8) = CIFS network back up applications can backup and restore UNIX symbolic links. Bit 4 set (0010000 or +16) = CIFS network back up applications can backup and restore all three names of files and directories. Bit 5 set (0100000 or +32) = Allows NFS v2 and v3 clients to view and modify ACLs by using the emcsetsd client tool. Bit 6 set (1000000 or +64) = Modifies the behavior of bit 1. UNIX rights applied are the granted rights less the denied rights by the discretionary ACL. Example: To enable Windows users to view and modify UNIX permissions, type: `$ server_param server_2 -facility cifs -modify acl.extacl -value 2` **Note:** To use emcsetsd to view and modify ACLs, you must first enable the emcsetsd tool. Examples: To use emcsetsd or emcgetsd tools from a UNIX client, you must set bit 5, type: `$ server_param server_2 -facility cifs -modify acl.extacl -value 32` To use emcsetsd and manage UNIX permissions from Windows, type: `$ server_param server_2 -facility cifs -modify acl.extacl -value 34`
Output
`server_2 : done`

Manage Windows ACL from a UNIX client

Use the emcgetsd and emcsetsd tools to perform the following tasks:

- Display the security descriptor on page 48
- View access rights on page 49

Note: Before you can use the emcsetsd tool, you must first enable it by using the cifs acl.extacl parameter. Manage UNIX permissions from a Windows client on page 46 provides procedural information for this task.

Appendix A provides detailed information about the emcgetsd and emcsetsd tools.

Display security descriptor

If a file or directory has SIDs in an ACL belonging to more than one domain, the output lists the users in the format of domain/user or domain/group without the -D option being specified in the emcgetsd command.

Action
To display the security descriptor of a file system by using the verbose option, use this command syntax: `# ./emcgetsd -v <local_node_path>` where: *<local_node_path>* = path of the file or directory on the UNIX client Example: To display the security descriptor of file system /fs2000A/apache/logs by using the verbose option, type: `# ./emcgetsd -v /fs2000A/apache/logs`
Output
`Dump of /fs2000A/apache/logs Security Descriptor` `----------` `Owner uid=677 fro` `Group gid=2765 media` `DACL` `...` `READ_CONTROL` `Flags 0x3` `OBJECT_INHERIT` `CONTAINER_INHERIT` `SACL` `None`

View access rights

The emcgetsd tool allows you to view ACLs on a file or directory from a UNIX client or Control Station.

Action
To display the access rights of a user currently logged in from a UNIX client, use this command syntax: `# ./emcgetsd -a <local_node_path>` where: `<local_node_path>` = path of the file or directory on the UNIX client Example: `# ./emcgetsd -a /fred/test1/TestDir`
Output
`Server=dffr1, Path in the server=//test1/TestDir` `Access of user uid=602 with groups gids={107-2765} on //test1/TestDir` `NT Rights: R-XPDO 0x1f00e9` `ReadExecute` `Read ListFolderContents`

Use UNIX GIDs for file system objects

The cifs acl.useUnixGid parameter controls whether VNX obtains an FSO's GID from a user's primary group or from the user's GID stored in the passwd file, NIS, or AD.

Action
To set the GID mapping for FSOs created on a Windows client to the Windows user's GID stored in the passwd file, NIS, or AD, use this command syntax: `$ server_param <movername> -facility -modify acl.useUnixGid -value <new_value>` where: `<movername>` = name of the Data Mover `<new_value>` = value of the specified parameter Examples: To set the GID mapping for file system objects created on a Windows client to the Windows user's GID, type: `$ server_param server_2 -facility -modify acl.UnixGid -value 1`
Output
`server_2 : done`

Determine the GIDs on copied file system objects

Typically, when a Windows user copies a file system object (FSO) by using a tool such as Windows Explorer, the ownership of the FSO is assigned to the user who performed the copy. VNX also maintains GIDs on FSOs. A GID must be applied to the copied FSO.

Action
To change the source of the GID for the copied FSO, that is to determine that the primary group is derived from the source specified by the acl.useUnixGID, use this command syntax: `$ server_param <movername> -facility cifs -modify acl.takegroupship-value 1` where: *<movername>* = name of the Data Mover Example: To determine that the primary group is derived from the source specified by the acl.useUnixGID, type: `$ server_param server_2 -facility cifs -modify acl.takegroupship -value 1`
Output
`server_2 : done`

Set the file locking policy

When mounting a file system, the policy to control the interaction of CIFS and NFS locking can be specified.

The file locking option that you choose depends on your business requirements and whether the network environment is CIFS only or a multiprotocol environment. File locking on page 30 provides more information about file locking policies in a multiprotocol environment.

Action
To specify file system locking, use this command syntax: `$ server_mount <movername> -option [nolock|wlock|rwlock] <fs_name> <mount_point>` where: *<movername>* = name of the Data Mover *<fs_name>* = name of the file system being mounted *<mount_point>* = name of the mount point Example: To mount the file system ufs1 with a read/write lock, type: `$ server_mount server_2 -option rwlock ufs1 /ufs1`

Output
server_2 : done

Configure and administer DFS support

Before you begin

Complete the following tasks before configuring a DFS root on a CIFS share. *Configuring CIFS on VNX* provides detailed instructions for these tasks:

1. Configure the CIFS server on the Data Mover.
2. Start the CIFS service on the Data Mover, which automatically enables DFS support.
3. On the CIFS server, configure a file system share on which to create the DFS root.

Note: Do not establish a DFS root on a file system object with an access-checking policy of UNIX or SECURE because none of the DFS link components are created with UNIX rights.

DFS provided with	Share type	Number of DFS roots
Windows 2000	Local	Single
Windows Server 2003 Windows XP	Global	Multiple This is the recommended option as it can manage multiple DFS roots on the same CIFS server.

To create a DFS root on a CIFS share, perform one of the following:

- Create a DFS root using dfsutil.exe on page 51
- Create a stand-alone DFS root using DFS MMC on page 52

Create a DFS root using dfsutil.exe

Note: If you intend to use a CIFS server as a stand-alone DFS server only (no domain structure), you must use the dfsutil.exe to create the DFS root.

Note: You can use the optional flag to work with the API instead of the Windows Registry.

CAUTION When the DFS is queried on the network by executing the dfsutil /siteinfo:<cifs_server_name> command that comes with Microsoft Windows 2000 support tools, the client connects to the srvsvc pipe and issues a NetrDfsManager ReportSiteInfo command. VNX then returns a DCE RPC Fault of 0x1c010002 or Range Error. To avoid this error, use the Microsoft Windows Server 2003 dfsutil /sitename:<cifs_server_name> command.

Action
To create a DFS root on a global share in a Windows Server 2003 environment, use this command syntax: **C: dfsutil /AddstdRoot /Server:DM2-ANA0-1-SA /Share:wl_root-1**
Output
Microsoft(R) Windows(TM) Dfs Utility Version 4.0 Copyright (C) Microsoft Corporation 1991-2001. All Rights Reserved. Indicates that DfsUtil command completed successfully.

Create a stand-alone DFS root using DFS MMC

Use the New Root Wizard to create a stand-alone DFS root on the CIFS share:

1. Start the New Root Wizard tool from DFS MMC. Click **Next** on the Welcome screen to begin.
2. On the **Host Server** dialog box, type the name CIFS server on the Data Mover that will host the DFS root and then click **Next**.
3. On the **Root Type** dialog box, select **Stand-alone root**, and then click **Next**.

   The **Root Name** dialog box displays the UNC path to the root and the share.
4. Type a unique name for the DFS root. Optionally, add a comment if you want to further describe this DFS root. Click **Next**.
5. Browse to a folder that you want to share as part of the DFS environment. Select the folder and click **Next**.

   You can add additional shares to the DFS root at any time after the initial configuration.
6. Click **Finish** to complete the DFS New Root installation wizard.

Disable DFS support

1. Open the Registry Editor of your choice and locate the following Registry key:

   `HKEY_LOCAL_MACHINE\SOFTWARE\EMC\DFS\Enable`
2. Set the Registry key to zero on the Data Mover.

3. Stop and restart the CIFS service.

   After starting the CIFS service, DFS support is enabled by default.

Create wide links

The following task illustrates how to create two wide links to direct a Windows client to the fs_wslink-1\user1 directory on the local Data Mover and the fs_wlink-29\user1 directory on the remote Data Mover. After the two wide links are created, the user1 directory displays these symbolic links as directories in Windows instead of files, as explained in Wide links on page 31.

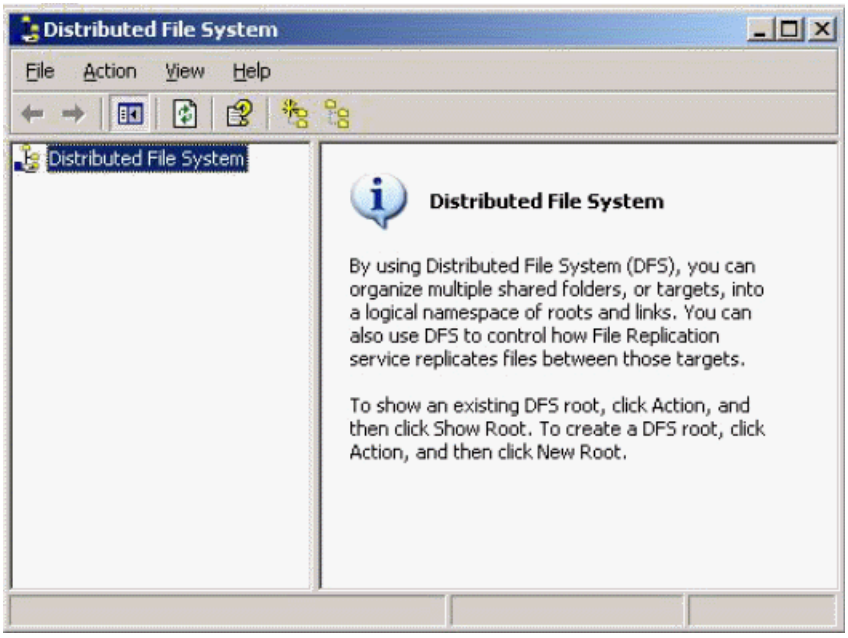
For purpose of illustration, the DFS root name used throughout the procedure is DM2-ANA0-1-SA.

You can configure a wide link on a per VDM basis. This enables a Windows client to be directed to as many different directory locations as needed.

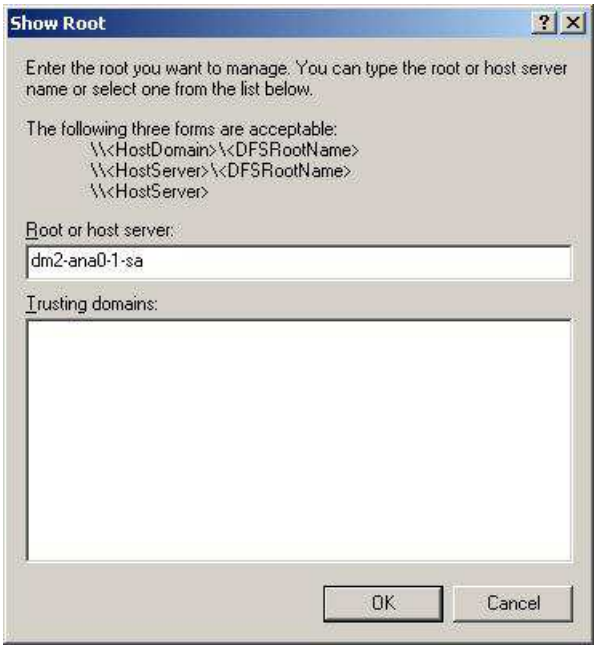
Note: This task assumes that DFS support is enabled (by default) and that the DFS roots are created. Distributed File System server on page 34 provides conceptual information.

---

1. Start the MMC Distributed File System tool. From the **Action** dialog box, select **Show Root**.

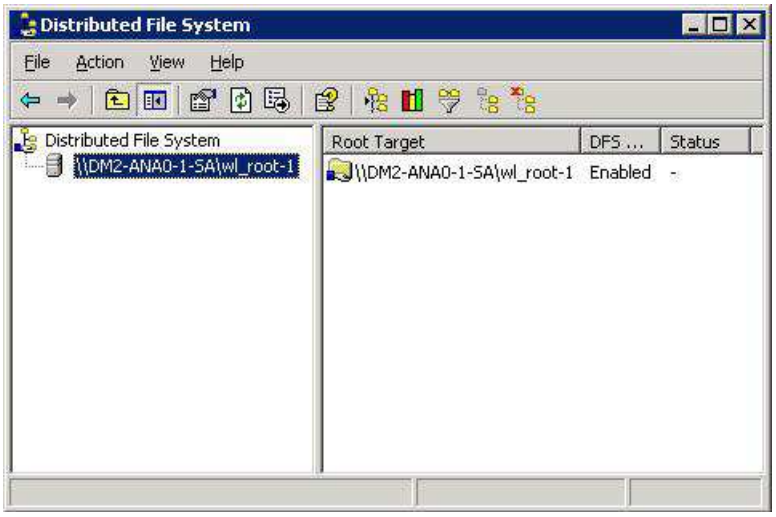


2. In the **Show Root** dialog box, type the NetBIOS name of the CIFS server used as the DFS root, on which a wide link is to be created. Then click **OK**.



3. The system displays the DFS roots for DM2-ANA0-1-SA.

   Select the DFS root on which a wide link is to be created.

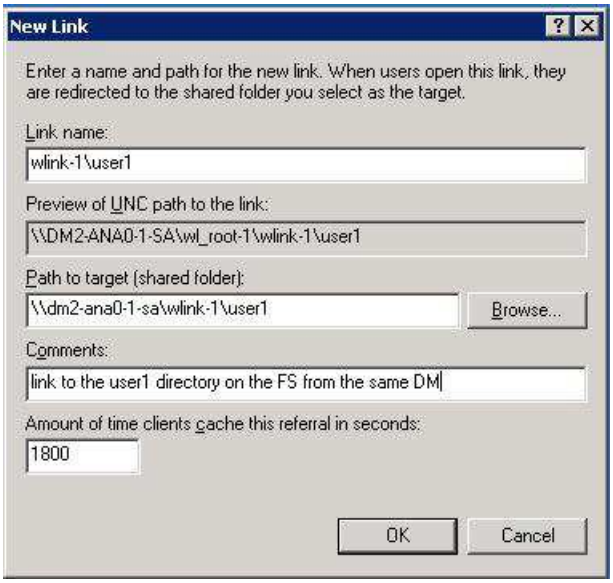


4. Right-click the \\DM2-ANA0-1-SA\w1_root-1 DFS root and select **New Link**.

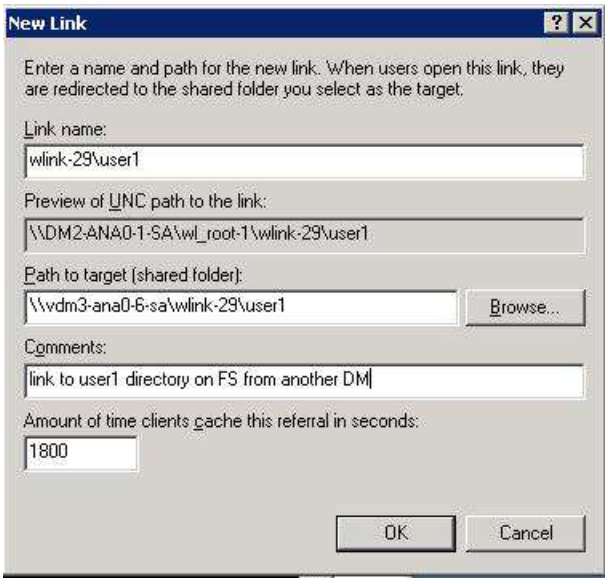
5. In the **New Link** dialog box, type the link name and path to target, which must be the same in Windows and UNIX, and then click **OK.**

Note: The UNIX name of each component must be the M256 name in Windows, as is shown in the following illustration.

This example shows how to create the first wide link, wlink-1\user1 to user1 on wlink-1 on the local Data Mover.



6. Create the second link to user1 on the remote Data Mover by repeating step 5. Type the link name and path to target.



7. Set the CIFS server and the DFS root in the Windows Registry:

a. Set the CIFS server and CIFS share by using the following Registry:
`HKEY_LOCAL_MACHINE\SOFTWARE\EMC\WideLink\Share`

The Registry must contain: `\\server_name\share_name`

where:

- server_name is the NetBIOS name of the CIFS server. If a global share is used, only CIFS share name is typed.
- share_name is the name of the CIFS share or the Windows share.

The following shows the Registry key for wl_root-1, which is a global share:

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\Software\EMC\WideLink]
"Share"="wl_root-1"
```

b. Stop the CIFS service.

c. Restart the CIFS service.

If the Registry is updated with a share name that is not in DFS, errors similar to the following appear in server_log:

```
SMB: 3: Widelink:\\Global\wl_root-1 is not in DFS
SMB: 3: Widelink: error while updating from Registry 7
```

If the Registry is set, but the wide links feature is not configured, messages similar to the following appear on the screen:

```
C:\>dir \\dm2-ana0-1-sa\wl_root-1\user1\user1_on_fs_wlink-1\
The network name cannot be found.
C:\>dir \\dm2-ana0-1-sa\wl_root-1\user1\user1_on_fs_wlink-29\
The network name cannot be found.
```

After setting the Registry key, symbolic links appear as directories in Windows, enabling users to read the contents of the following two directories:

Local Data Mover:

```
C:\>dir \\dm2-ana0-1-sa\wl_root-1\user1\user1_on_fs_wlink-1\
Volume in drive \\dm2-ana0-1-sa\wl_root-1 is 102
Volume Serial Number is 0000-0014
Directory of \\dm2-ana0-1-sa\wl_root-1\user1\user1_on_fs_wlink-1

02/02/2005 11:07 AM <DIR> .
02/02/2005 11:56 AM <DIR> ..
02/01/2005 07:08 PM 0 user1_NFS_file
02/02/2005 10:27 AM <DIR> CIFS-user
               1 File(s) 0 bytes
               3 Dir(s) 52,867,260,416 bytes free
```

Remote Data Mover:

```
C:\>dir \\dm2-ana0-1-sa\wl_root-1\user1\user1_on_fs_wlink-29\
Volume in drive \\dm2-ana0-1-sa\wl_root-1 is 102
Volume Serial Number is 0000-0014
Directory of \\dm2-ana0-1-sa\wl_root-1\user1\user1_on_fs_wlink-29

02/02/2005 05:11 PM <DIR> .
02/01/2005 05:17 PM <DIR> ..
02/02/2005 05:11 PM 0 NFS_user1_wlink-29
1 File(s) 0 bytes
2 Dir(s) 52,867,268,608 bytes free
```

4 Troubleshooting

As part of an effort to continuously improve and enhance the performance and capabilities of its product lines, EMC periodically releases new versions of its hardware and software. Therefore, some functions described in this document may not be supported by all versions of the software or hardware currently in use. For the most up-to-date information on product features, refer to your product release notes.

If a product does not function properly or does not function as described in this document, contact your EMC Customer Support Representative.

Problem Resolution Roadmap for VNX contains additional information about using the EMC Online Support website and resolving problems.

Topics included are:

EMC E-Lab Interoperability Navigator

The EMC E-Lab™ Interoperability Navigator is a searchable, web-based application that provides access to EMC interoperability support matrices. It is available on the EMC Online Support website at http://Support.EMC.com. After logging in, locate the applicable Support by Product page, find **Tools**, and click **E-Lab Interoperability Navigator**.

VNX user customized documentation

EMC provides the ability to create step-by-step planning, installation, and maintenance instructions tailored to your environment. To create VNX user customized documentation, go to: https://mydocs.emc.com/VNX.

server_log error message construct

The format of the event code can help narrow the scope of where to look for a message. There are several components in the beginning of each line that are fairly consistent across the entire scope of event logging. For example, the typical event message looks like:

```
2005-09-16 18:27:21: NFS: 3:  commit failed, status = NoPermission
2005-09-16 18:27:23: CFS: 3:  Failed to open file, status NoPermission
2005-09-16 18:27:23: LIB: 6:  last message repeated 1 times
```

The *EMC VNX Command Line Interface Reference for File* provides detailed information on server_log. This logging mechanism uses the logging facilities typical with many systems.

The various components of the event message are:

- The first part is the date and time of the logged event.
- The second part is the subsystem of VNX code that reported the event (for example, NFS, CFS, and LIB).
- The third part is a classification code, which is typical of event logging facilities. Information on classification codes can be found on most UNIX systems under the header file syslog.h in the directory /usr/include/sys.

  The definition of the possible classification codes that VNX supports are:

```
#define LOG_EMERG   0    /* system is unusable */
#define LOG_ALERT   1    /* action must be taken immediately */
#define LOG_CRIT    2    /* critical conditions */
#define LOG_ERR     3    /* error conditions */
#define LOG_WARNING 4    /* warning conditions */
#define LOG_NOTICE  5    /* normal but signification condition */
#define LOG_INFO    6    /* informational */
#define LOG_DEBUG   7    /* debug-level messages */
```

- The fourth part describes the error condition. The error condition on the first two lines of the example is self-explanatory. The operations being performed are ‘commit’ and ‘open’ with the error condition, NoPermission. Other events are not as descriptive.

Kerberos error codes

Kerberos error codes are status codes generally displayed by the Server Message Box (SMB) subsystem. These can be recognized in the logged events by the appearance of a large negative number.

Example Kerberos error code

```
2003-07-24 16:29:35: SMB: 3: SSXAK=c0020030 origin=401 stat=e0000,
-1765328160
```

Because Kerberos is standardized, there are public resources available on the Internet for looking up the meanings of a majority of these status codes.

NT status codes

The NT status codes are reported for CIFS or Microsoft Windows emulation functions on VNX product. The NT status codes are 32-bit unsigned integers that are broken up into subgroups of binary data that identify the particulars of an event status. The 32-bit values are laid out as follows:

```
3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
+---+-+-+-----------------------+-------------------------------+
|Sev|C|R|     Facility          |               Code            |
+---+-+-+-----------------------+-------------------------------+
```

where:

- Sev — Is the severity code:
  - 00 — Success
  - 01 — Informational
  - 10 — Warning
  - 11 — Error
- C — Is the customer code flag
- R — Is a reserved bit
- Facility — Is the facility code
- Code — Is the status code of the facility

Typically, the NT status codes appear in the server_log with a subsystem specification of SMB. The NT status code is presented in several ways in logged system events. Some popular ones are:

- A hexadecimal number prefixed by a Em=0x:

  SMB: 4: authLogon=SamLogonInvalidReply Es=0x0 Em=0xc0000064

- A simple hexadecimal number with no prefix nor any indication of its format:

  SMB: 4: SSXAuth_SERVER_EXT13 aT=3 mT=1 c0000016

- A simple hexadecimal number with a prefix of reply= with no indication of the format:

  SMB: 4: lookupNames:bad reply=c0000073

- A simple hexadecimal number with a prefix of failed= with no indication of the format:

  SMB: 4: SessSetupX failed=c0000016

- A hexadecimal number clearly marked as NTStatus= but no indication of the format:

  SMB: 4: MsError sendLookupNames=21 NTStatus=c0000073

Known problems and limitations

Table 12 on page 62 describes problems that might occur when using VNX for a multiprotocol environment and presents workarounds.

Table 12. Known problems

Known problem	Symptom	Workaround
With NT user authentication, certain Windows 95 clients might not be able to map drives from the Data Mover.	The domain name sent to the Data Mover by the client was incorrectly specified, or the username.domain is not mapped in the passwd file on the Data Mover.	Verify that the client is sending the correct domain name to the passwd file. To verify that the client is sending the correct domain: ◆ In the Network option in the Control Panel, double-click the network client (Client for Microsoft Networks). ◆ Under General properties, verify that the correct domain name is shown.

Table 12. Known problems *(continued)*

Known problem	Symptom	Workaround
With NT user authentication, `Incorrect password` or `unknown username` error message appears after attempts to connect to the server, and the username and password window appears.	The Windows NT user account might be missing from the primary domain controller (PDC), or the Data Mover was unable to determine a UID to use for this user.	Add the Windows NT user to the PDC and map the user to a UNIX username and UID.
With NT user authentication, clients are unable to connect to the server, and the window to prompt for username and password does not appear on the client side.	No domain controller found for the domain.	Check if PDC or backup domain controller (BDC) is up. Check if the Data Mover can access a WINS server that knows about the PDC domain, or have the PDC and BDC in the same local subnet as the Data Mover.
	The server NetBIOS name is not registered as a computer account on the PDC domain or a trust relationship has not been established between the client and server domains. The following message appears in the server_log: `The SAM database on the Windows NT server does not have a complete account for this workstation trust relationship.`	Verify that the computer account exists and add the computer account, if needed. If the computer account does exist, remove it and add it again before retrying the command. The Microsoft NT server 4.0 documentation provides more information on setting up a trust relationship between domains.
After joining a CIFS server to a domain, the following error appears in the server_cifs output, indicating that the system cannot update the DNS record: `FQDN=dm4-a140-ana0.c1t1.pt1.c3lab.nsgprod.emc.com (Update of "A" record failed during update: Operation refused for policy or security reasons)`	The DNS servers zone might include the same fully-qualified domain name (FQDN) for another computer account.	Verify that the DNS server's zone does not have the same FQDN with a different IP address for another computer account.

Table 12. Known problems *(continued)*

Known problem	Symptom	Workaround
`0xC0000022 2004-04-26 10:49:40: SMB: 3: Srv=<VNX_netbios_name> buildSecureChanel=Authenticate2 InvalidReply E=0xc0000022`	Access is denied because the computer was created on the domain controller without enabling the Allow pre-Windows 2000 computers to use this account option on the Windows New Object - Computer dialog box.	Delete the computer and then re-create it with the Allow pre-Windows 2000 computers to use this account option enabled.
When attempting to start MMC, the following error message appears: `OLE Object: PBrush`	MMC requires Internet Explorer 6.0 to use its Document Object Model (DOM) XML parser.	Upgrade the version of the Internet Explorer to 6.0.
Solaris client receives the following warning message during the creation of a user account: `UX:useradd: WARNING: more than NGROUPS_MAX(16) groups specified` The user account is still created but data availability might occur. When attempting access, Solaris client receives an error message similar to: `nfs: [ID XXXXXX kern.notice] NFS access failed for server dm3-121-ana0-2: error 1 (RPC: Can not encode arguments)`	This is likely caused by the Solaris NGROUPS_MAX kernel parameter being set to more than 16 groups, which is the default limit on Solaris systems. NFS only has support for a maximum of 16 groups. Depending on the UNIX implementation, the limit on the number of groups per user is different: ◆ Solaris has a limit of 16 groups ◆ Linux has a limit of 32 groups	In a multiprotocol environment, to extend this group number to more than 16, use VNX File Server NT credential feature.
When upgrading from a Windows NT domain to Windows 2000, unable to change the original domain suffix during Windows 2000 setup.	Unable to change domain suffix because it was hardcoded in dynamic DNS (DDNS).	Before upgrading, change the domain suffix.
Access is denied to Internet Information Services (IIS) 6.0 when attempting to connect to the web directory on VNX share. In the IIS web log, the error bad username or password appears even though the username and password are in the local user database.	For a stand-alone CIFS server with local user support enabled, the username and password must be the same on IIS 6.0, the Data Mover, and the client.	Specify the same username and password on IIS 6.0, the Data Mover, and the client.

Error messages

All event, alert, and status messages provide detailed information and recommended actions to help you troubleshoot the situation.

To view message details, use any of these methods:

- Unisphere software:
  - Right-click an event, alert, or status message and select to view Event Details, Alert Details, or Status Details.
- CLI:
  - Type nas_message -info <MessageID>, where <MessageID> is the message identification number.
- *Celerra Error Messages Guide*:
  - Use this guide to locate information about messages that are in the earlier-release message format.
- EMC Online Support website:
  - Use the text from the error message's brief description or the message's ID to search the Knowledgebase on the EMC Online Support website. After logging in to EMC Online Support, locate the applicable **Support by Product** page, and search for the error message.

EMC Training and Professional Services

EMC Customer Education courses help you learn how EMC storage products work together within your environment to maximize your entire infrastructure investment. EMC Customer Education features online and hands-on training in state-of-the-art labs conveniently located throughout the world. EMC customer training courses are developed and delivered by EMC experts. Go to the EMC Online Support website at http://Support.EMC.com for course and registration information.

EMC Professional Services can help you implement your system efficiently. Consultants evaluate your business, IT processes, and technology, and recommend ways that you can leverage your information for the most benefit. From business plan to implementation, you get the experience and expertise that you need without straining your IT staff or hiring and training new personnel. Contact your EMC Customer Support Representative for more information.

Appendix A

emcgetsd and emcsetsd

The emcgetsd and emcsetsd tools can be used on Linux, Solaris, and HP-UX operating systems. Use the executable appropriate for your operating system.

You can copy these tools on to a UNIX client without performing an installation procedure on the client. EMC recommends that prior to using these tools, you use the chmod command to be sure that the files are executable, for example:

```
chmod 755 <filename>
```

Topic included is:

- Using emcgetsd and emcsetsd on page 68

Using emcgetsd and emcsetsd

View ACLs

Use the emcgetsd tool to view ACLs on a file or directory from a UNIX client or Control Station. Table 13 on page 68 lists the emcgetsd tool command options and descriptions.

Table 13. emcgetsd

Command	Description
emcgetsd -D *<domain>* **-v -x** *<local_node_path>* **emcgetsd -a** *<local_node_path>*	Displays the security descriptor of a file system. A security descriptor lists the owner, ACL, and auditing information of the file system.
emcgetsd -D*<domain>*	Directs the command to a specified domain. This domain can be different from the user domain if a trust relationship exists between the user domain and another domain. In this case, the command displays the SIDs for both domains. If the domain is not specified, CNS uses the default domain of the CIFS server.
-v	Displays full ACL details.
-x	If CIFS is not started or if the Windows name of a user or group cannot be found, the SID of this user is returned in decimal format unless the –x option is specified.
-s	Displays the access rights of a user currently logged in from a UNIX client or a Control Station.
<local_node_path>	Path of the file or directory on the UNIX client.

Modify ACLs

Use the emcsetsd tool to modify and view the ACL on a file or directory from a UNIX client or Control Station.

When Windows permissions are changed by using the emcsetsd tool, the Windows owner is replaced by the UNIX SID and the UNIX UID/GID, as shown in the following examples:

```
Owner uid=898 Unix='luc' Sid=S-1-5-18-1-898

Group gid=109 Unix='emc2' Sid=S-1-5-18-2-109
```

Note: You must have the appropriate rights to use this tool.

Table 14 on page 70 lists the emcsetsd tool command options and descriptions.

Table 14. emcsetsd

Command	Description
emcsetsd -D<*domain*> **-r -g** <*user_or_group*>,<*rights*>[,<*flags*>] **-d** <*user_or_group*>,<*rights*>[,<*flags*>] **-s** <*user_or_group*>,<*rights*>[,<*flags*>] **-f** <*user_or_group*>,<*rights*>[,<*flags*>] **-a**<*user_or_group*>,<*rights*>[,<*flags*>] <*local_node_path*>	

Table 14. emcsetsd *(continued)*

Command	Description
	Sets, resets, and audits user or group access control rights on a file or directory. **Note:** If CIFS is not started or if the Windows name of a user or group is not found, the command is rejected. User A user can be one of the following: ◆ UI=number ◆ User=NIS name ◆ domain\user Group A group can be one of the following: ◆ GID=number ◆ Group=NIS name ◆ Everyone ◆ CreatorOwner ◆ CreatorGroup ◆ domain\user **Note:** The user and group owner can be changed by using the chown or chgrp UNIX command. Rights The rights can be one of the following separated by a pipe (|): ◆ READ_DATA ◆ WRITE_DATA ◆ APPEND_DATA ◆ READ_EA

Table 14. emcsetsd *(continued)*

Command	Description
	◆ WRITE_EA ◆ EXECUTE ◆ DELETE_CHILD ◆ READ_ATTRIBUTES ◆ WRITE_ATTRIBUTES ◆ DELETE ◆ READ_CONTROL ◆ WRITE_DAC ◆ WRITE_OWNER A combination of RWXPDO: ◆ R: Read ◆ W: Write ◆ X: Execute ◆ P: ChangePermission ◆ D: Delete ◆ O: TakeOwnership One or more of the following separated by a pipe (|): ◆ FullControl ◆ Modify ◆ ReadExecute ◆ ListFolderContents ◆ Read ◆ Write Flags

Table 14. emcsetsd *(continued)*

Command	Description
	One or more of the following values separated by a pipe (|): ◆ OBJECT_INHERIT: subfiles inherit this ACE. ◆ CONTAINER_INHERIT: subfolders inherit this ACE. ◆ NO_PROPAGATE_INHERIT: block inheritance from its parent. ◆ INHERIT_ONLY: ACE is not part of access rights on the current directory, only for inheritance. ◆ INHERITED_ACE: ACE was inherited.
-D *<domain>*	Directs the command to a specified domain. This domain can be different from the user domain if there is a trust relationship between the user domain and another domain. In this case, the command displays the SIDs for both domains. If the domain is not specified, CNS uses the default domain of the CIFS server.
-r	Removes current ACLs. **Note:** When the -r option is used, the SID of the owner and group are replaced by UNIX SIDs and therefore, after using the -r option, the identity of the owner and group reflects the new SIDs.
-g	Grants access to a user or group.
-d	Denies access to a user or group.
-s	Audits success access of a user or group.

Table 14. emcsetsd *(continued)*

Command	Description
-f	Audits fail access of a user or group.
-a	Audits all access of a user or group.
<local_node_path>	Specifies the path of the file or directory on the UNIX client.

Appendix B

nfs4_getfacl, nfs4_setfacl, and nfs4_editfacl

The nfs4_getfacl, nfs4_editfacl, and nfs4_setfacl are Linux tools. They are part of the nfs4_acl_tools package. These tools can be used to get the ACLs data for any file or directory.

Topics included are:

NFS4 ACL

An NFS4 ACL contains an ordered sequence of ACEs. NFS4 ACL can be used on an NFS4 file system by using the nfs4_acl_tools package. Each ACE has type, flags, and an access mask.

The bitmask constants used for the ACE flag fields are as follows:

```
const ACE4_FILE_INHERIT_ACE             = 0x00000001;
const ACE4_DIRECTORY_INHERIT_ACE        = 0x00000002;
const ACE4_NO_PROPAGATE_INHERIT_ACE     = 0x00000004;
const ACE4_INHERIT_ONLY_ACE             = 0x00000008;
const ACE4_SUCCESSFUL_ACCESS_ACE_FLAG   = 0x00000010;
const ACE4_FAILED_ACCESS_ACE_FLAG       = 0x00000020;
const ACE4_IDENTIFIER_GROUP             = 0x00000040;
const ACE4_INHERITED_ACE                = 0x00000080;
```

The VNX translates the flags which are stored in the ACL to these flags.

Table 15 on page 76 describes the flags stored in the SD for each ACE.

Table 15. VNX ACL Flags

Flag Values	Bits	Description
#define EFFECTIVE_ACE	(0x0)	effective only
#define OBJECT_INHERIT_ACE	(0x1)	ACE used on file creation
#define CONTAINER_INHERIT_ACE	(0x2)	ACE used on directory creation
#define NO_PROPAGATE_INHERIT_ACE	(0x4)	ACE not propagated
#define INHERIT_ONLY_ACE	(0x8)	ACE used only for inheritance
#define INHERITED_ACE	(0x10)	ACE inherited cannot be overwritten without special action
#define ACE4_INHERITED_ACE	(0x80)	The O flag in nfs4_getfacl correspond to this flag

The bit INHERITED_ACE (0x10) is translated by the protocol layer to the bit ACE4_INHERITED_ACE (0x80). The new O flag is displayed and managed by the tools nfs4_getfacl, nfs4_setfacl, and nfs4_editfacl.

Using nfs4_getfacl, nfs4_setfacl, and nfs4_editfacl

nfs4_getfacl

The nfs4_getfacl is used to view the NFSv4 ACL for file or directory, provided that the file is on a mounted NFSv4 file system which supports ACLs. A file system which supports NFSv4 ACL is mounted with the option accesspolicy=MIXED.

To read an acl, use the nfs4_getfacl:

```
nfs4_getfacl /mnt/nfsv4/fs_name
```

nfs4_setfacl

Use the nfs4_setfacl tool to modify the NFSv4 ACL of one or more files or directories. The file is required to be on a mounted NFSv4 file system which supports ACLs. A file system that supports NFSv4 ACL is mounted with the option accesspolicy=MIXED.

To modify an acl, use the nfs4_setfacl:

```
nfs4_setfacl -e /mnt/nfsv4/fs_name
```

nfs4_editfacl

The nfs4_editfacl is equivalent to nfs4_setfacl. The nfs4_editfacl edits file's ACL in the editor defined in the EDITOR environment variable.

Note: These are not EMC tools. You can refer to the nfs4_getfacl and nfs4_setfacl man pages available online for more information.

Glossary

A

access control list (ACL)
List of access control entries (ACEs) that provide information about the users and groups allowed access to an object.

Active Directory (AD)
Advanced directory service included with Windows operating systems. It stores information about objects on a network and makes this information available to users and network administrators through a protocol such as Lightweight Directory Access Protocol (LDAP).

Active Directory Users and Computers (ADUC)
Administrative tool designed to perform day-to-day Active Directory administration tasks. These tasks include creating, deleting, modifying, moving, and setting permissions on objects stored in the directory. These objects include organizational units, users, contacts, groups, computers, printers, and shared file objects.

Alternate data stream (ADS)
Alternate data stream allows files to be associated with more than one data stream. For example, a file such as text.txt can have an ADS with the name of text.txt:secret (of form filename:streamname) that can only be accessed by knowing the ADS name or by specialized directory browsing programs.

authentication
Process for verifying the identity of a user trying to access a resource, object, or service, such as a file or a directory.

C

CIFS server
Logical server that uses the CIFS protocol to transfer files. A Data Mover can host many instances of a CIFS server. Each instance is referred to as a CIFS server.

CIFS service
CIFS server process that is running on the Data Mover and presents shares on a network as well as on Microsoft Windows-based computers.

Common Internet File System (CIFS)
File-sharing protocol based on the Microsoft Server Message Block (SMB). It allows users to share file systems over the Internet and intranets.

D

Data Mover
In VNX for file, a cabinet component that is running its own operating system that retrieves data from a storage device and makes it available to a network client. This is also referred to as a blade.

default CIFS server
CIFS server created when you add a CIFS server and do not specify any interfaces (with the interfaces= option of the server_cifs -add command). The default CIFS server uses all interfaces not assigned to other CIFS servers on the Data Mover.

domain
Logical grouping of Microsoft Windows Servers and other computers that share common security and user account information. All resources such as computers and users are domain members and have an account in the domain that uniquely identifies them. The domain administrator creates one user account for each user in the domain, and the users log in to the domain once. Users do not log in to each individual server.

Domain Name System (DNS)
Name resolution software that allows users to locate computers on a UNIX network or TCP/IP network by domain name. The DNS server maintains a database of domain names, hostnames, and their corresponding IP addresses, and services provided by the application servers.

See also *ntxmap*.

F

File Allocation Table (FAT)
File system used by MS-DOS and other Windows-based operating systems to organize and manage files. The file allocation table (FAT) is a data structure that Windows creates when you format a volume by using the FAT or FAT32 file systems. Windows stores information about each file in the FAT so that it can retrieve the file later.

file system
Method of cataloging and managing the files and directories on a system.

G

Group Policy Objects (GPO)
In Windows operating systems, administrators can use Group Policy to define configuration options for groups of users and computers. Windows Group Policy Objects can control elements such as local, domain, and network security settings.

L

Lightweight Directory Access Protocol (LDAP)
Industry-standard information access protocol that runs directly over TCP/IP. It is the primary access protocol for Active Directory and LDAP-based directory servers. LDAP version 3 is defined by a set of Proposed Standard documents in Internet Engineering Task Force (IETF) RFC 2251.

N

NetBIOS name
Name recognized by WINS, which maps the name to an IP address.

network basic input/output system (NetBIOS)
Network programming interface and protocol developed for IBM personal computers.

network file system (NFS)
Network file system (NFS) is a network file system protocol that allows a user on a client computer to access files over a network as easily as if the network devices were attached to its local disks.

NTFS
NTFS is the standard file system of Windows NT, including its later versions. NTFS supersedes the FAT file system as the preferred file system for Microsoft Windows. NTFS has several improvements over FAT such as improved support for metadata and the use of advanced data structures to improve performance, reliability, and disk space utilization, plus additional extensions such as security access control lists (ACLs) and file system journaling.

S

Security Access Manager or Security Accounts Manager (SAM)
Microsoft Windows service that authenticates users to use resources on the network. The SAM database is the location for all security and user account information for a Windows NT domain.

Server Message Block (SMB)
Underlying protocol used by the CIFS protocol enhanced for use on the Internet to request file, print, and communication services from a server over the network. The CIFS protocol uses SMB to provide secure file access and transfer to many types of hosts such as LANs, intranets, and the Internet.

share name
Name given to a file system, or resource on a file system available from a particular CIFS server to CIFS users. There may be multiple shares with the same name, shared from different CIFS servers.

V

Virtual Data Mover (VDM)
VNX for file software feature that enables users to administratively separate CIFS servers, replicate CIFS environments, and move CIFS servers from one Data Mover to another.

W

Windows domain
Microsoft Windows domain controlled and managed by a Microsoft Windows Server by using the Active Directory to manage all system resources and by using the DNS for name resolution.

Windows Internet Naming Service (WINS)
Software service that dynamically maps IP addresses to computer names (NetBIOS names). This allows users to access resources by name instead of requiring them to use IP addresses that are difficult to recognize and remember. WINS servers support clients by running Windows NT 4.0 and earlier versions of Microsoft operating systems.

Windows NT domain
Microsoft Windows domain controlled and managed by a Microsoft Windows NT server by using a SAM database to manage user and group accounts and a NetBIOS namespace. In a Windows NT domain, there is one primary domain controller (PDC) with a read/write copy of the SAM, and possibly several backup domain controllers (BDCs) with read-only copies of the SAM.

See also *domain* and *domain controller*.

Index

N

P

S

T

U

W