Technical Report

NFSv4 Enhancements and Best Practices Guide: Data ONTAP Implementation

Justin Parisi, Bikash Roy Choudhury, NetApp
June 2013 | TR-3580

Abstract

Network File System version 4 (NFSv4) is the latest version of NFS with new features such as statefulness, improved security and strong authentication, improved performance, file caching, integrated locking, access control lists (ACLs), and better support for Windows® file sharing semantics. Data ONTAP® 7.3.5 and 8.1 (7-Mode and clustered Data ONTAP) support NFSv4. Switching from NFSv3 to NFSv4 or implementation of NFSv4 in new environments requires proper planning and adherence to best practices. This document describes the best practices that should be followed while implementing NFSv4 components on AIX, Linux®, or Solaris clients attached to NetApp® storage systems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Executive Summary

Network File System (NFS) allows a client computer to access files on shared storage over the network. Clients are generally Linux or UNIX®, but Windows also supports NFS using service for UNIX packages. NFS builds on the Open Network Computing Remote Procedure Call (ONC RPC) system. Its technical and implementation details are defined in RFC 1094. (RFC, or request for comments, is a memorandum published by the Internet Engineering Task Force, or IETF.) NFS version 4 (NFSv4) is an improved version of NFS protocol versions 2 and 3. NFSv4 retains the essential features of versions 2 and 3 such as easy recovery and reliability; simplicity; good performance; and independence from transport protocols, operating systems, and file systems. It also includes some very important enhancements such as:

- Support for integrated locking
- Support for strong security
- Compound operations to improve performance
- Client caching
- Protocol extension support for backward compatibility
- Interoperability with Windows Common Internet File System (CIFS)

This document provides an introduction to the NFS protocol, beginning with an overview of NFS and NFSv4. Data ONTAP 7G and 8.2 (7-Mode) and 8.2 clustered Data ONTAP implementation specifics of NFSv4 such as pseudo file system, ACLs, and delegations are also provided. This also covers various best practices to configure NFSv4 clients running AIX, Linux, and Solaris and connected to NetApp storage running Data ONTAP version 7G or 8.2 (7-Mode) or clustered Data ONTAP 8.2.

2 Intended Audience

This document is intended for system administrators, system and storage architects, database administrators, and server and network administrators. A general working knowledge of NFS and NFSv4 is recommended.

3 NFSv4 Overview

NFS has been the standard distributed file system for UNIX platforms and has been widely used for over two decades. It operates on a client/server basis and allows users to access files and directories on remote servers over the network. Users employ operating system commands on the client to create, delete, read, write, and set attributes of remote files and directories on the server. It is available on all versions of UNIX, Linux, and other well-known operating systems. NFS uses remote procedure calls (RPCs) to remain independent from machine types, operating systems, and network architectures. At a high level, the NFS architecture consists of the following components:

- Network protocols
- Kernel extensions
- Client and server daemons

This report covers the protocols, planning, and implementation of NFS V4, but kernel extensions and daemons are beyond its scope.

3.1 NFSv4 Enhancements

Simplicity, reliability, and ease of manageability led to the wide adoption of NFS in the distributed file system landscape. As business complexity grew, customers demanded stronger authentication, fine-grained access control, multiplatform access, and better I/O performance that the existing NFS versions

failed to address. NFS version 4 inherits all essential features of version 2 and 3 and goes a long way toward addressing the limitations of v2 and v3. NFSv4 provides the following additional features:

- Enhanced built-in security
- Better namespace handling
- Improved and integrated locking support
- Improved performance over the network
- Cross-platform interoperability, including Microsoft® Windows
- Protocol extension to support backward compatibility
- Movement toward an open standard, managed by the IETF, whereas previous versions of NFS were proprietary

For detailed information on enhancements, refer to RFC3530 (www.ietf.org/rfc/rfc3530.txt).

Built-In Security with NFSv4: Kerberos and ACLs

Data security and protection are of utmost priority for businesses. NFS version 4 mandates a strong security model, where client/server interactions over the network are done using the Generic Security Services API (GSS-API) framework called RPCSEC_GSS. The framework supports three levels of protection (authentication, integrity, and privacy) over the network between the server and client. NFS version 4 supports three security mechanisms:

- Kerberos (RFC 4120)
- Low Infrastructure Public Key Mechanism (LIPKEY RFC 2847)
- Simple Public Key Mechanism (SPKM-3 RFC 2847)

The only security mechanism that NetApp and most NFSv4 clients currently provide under RPCSEC_GSS is Kerberos 5. Kerberos uses symmetrical key encryption as an authentication mechanism. It never sends passwords across the network in either cleartext or encrypted form. It uses encrypted tickets and session keys to authenticate users before they can use network resources. The Kerberos system uses key distribution centers (KDCs) that contain a centralized database of user names and passwords. NetApp supports two types of KDCs: UNIX and Windows Active Directory®.

Kerberos provides strong authentication for client-server communications through encryption. The security stack of NFSv4 uses Kerberos to provide granular, improved security. NFSv3 supported Kerberos, but the implementation was weaker. Data ONTAP NFSv4 support uses RPCSEC_GSS with Kerberos for authentication, integrity, and privacy. For further details on Data ONTAP Kerberos implementation, refer to section 7 in this document. Data ONTAP 7.3.5 and later support multirealm, which means you can have MIT configuration for UNIX clients talk to a Windows 2008R2 Active Directory that is configured for Windows clients. Clustered Data ONTAP also supports multiple Kerberos realms, which can be configured in the same storage virtual machine (SVM) on a per data LIF basis.

Kerberos is recommended, but not required, in NFSv4 environments.

ACLs significantly improve NFSv4's security and interoperability with CIFS. ACLs provide much more granular access control and a greater degree of selectivity than traditional UNIX mode permission bits. NFSv4 ACLs are based on the NT model, but they do not contain owner/group information. NFSv4 ACLs are made up of an array of access control entries (ACEs), which contain information regarding access allowed/denied, permission bits, user name/group name, and flags.

Along with support for strong security mechanisms, NFSv4 has implemented improved username/UID management by allowing character strings (for example, user@domain) instead of integers to represent user and group identifiers.

ACLs are a new feature in NFSv4, but are not required for use with NFSv4.

Pseudo File System

NFSv4 introduces the concept of pseudo file system, which provides a single point of entry "/" (root) to access all the file systems below that entry point. This is similar to but not the same as the procfs found in other Linux implementations of pseudo file system. For more information on procfs, see the following: http://linux.die.net/man/5/proc.

In RFC 3530, the following is mentioned regarding the NFSv4 server implantation of pseudo file system:

> *NFS version 4 servers avoid this name space inconsistency by presenting all the exports within the framework of a single server name space. An NFS version 4 client uses LOOKUP and READDIR operations to browse seamlessly from one export to another. Portions of the server name space that are not exported are bridged using a "pseudo filesystem" that provides a view of exported directories only. A pseudo filesystem has a unique fsid and behaves like a normal, read only filesystem.*

The Data ONTAP implementation of a pseudo file system in NFSv4 is leveraged as an entry point into a storage system for determining mount points. A pseudo file system allows the use of one port for security, rather than several. All NFSv4 servers support the use of a pseudo file system. One of the main things to consider about the implementation of a pseudo file system in Data ONTAP is that there might be inconsistencies with mount points between NFSv3 and NFSv4.

In the examples that follow, consider these volumes:

```
/vol/vol0 (root)
/vol/vol1
/vol/home
```

Example 1

In NFSv3 if filer:/ is mounted, the mount point is filer:/vol/vol0. That is, if the path does not begin with /vol in NFSv3, Data ONTAP adds /vol/vol0 to the beginning of the path.

In NFSv4, if filer:/ is mounted, the root of the pseudo file system is mounted and not /vol/vol0. Data ONTAP does not add /vol/vol0 to the beginning of the path in NFSv4.

Example 2

In the Data ONTAP implementation of the NFSv4 pseudo file system, the nodes "/" and "/vol" are always present and form the common prefix of any reference into the pseudo file system. Any reference that does not begin with "/vol" is invalid.

In this example, there is a /vol/vol0/home directory. In NFSv3, if filer:/home/users is mounted, /home is considered as the directory /vol/vol0/home. In NFSv4 filer:/home/users is mounted, /home is not interpreted as the volume /vol/home; it is considered an invalid path in the pseudo file system tree.

When NFSv4 and the pseudo file system are used, the client request will traverse from pseudo file system to an active file system. During this process, the fsid changes from pseudo file system to active file system. The pseudo file system is completely virtual and never takes any physical space. Pseudo file system presents the concept of a global namespace with all volumes/filesystems stitched below the root. In clustered Data ONTAP, the concept of a single namespace is presented in both NFSv3 and NFSv4, so a pseudo file system only needs to be considered when volume permissions flow from less restrictive at the top to more restrictive down the path.

Improved and Integrated Locking Support

NFSv4 does not support ancillary protocols such as Network Lock Manager (NLM), also called lockd, and Network Status Monitor (NSM), also called statd. NFSv4 introduces a new and improved locking mechanism called leased-based locking. In this mechanism, the server assigns a lease to every lock that

is granted to the client. The server checks the client every 30 seconds before the lease expires in case the client responds back to the server to extend it further. This checking is done on a per client basis and per lock on a single client by the server. The value is configurable from the NFS server (NetApp storage) side. The lease time period reduces the chattiness between the server and the client. It is not recommended to change these in most cases.

In 7-Mode (hidden option):

```
Filer> options nfs.v4.lease_seconds [value]
```

In clustered Data ONTAP (diag option):

```
cluster::> set diag
cluster::*> nfs server modify -vserver vs0 -v4-lease-seconds [value]
```

Each lock is stored in server (NetApp storage) memory as a state ID. In the event of a cluster takeover/giveback or if the client crashes, the state information is exchanged between the server and the client after they establish connectivity. Locks are first given back to the processes that owned them prior to the crash, before the server assigns new opens and locks to requesting clients. After a storage failover/giveback, lock reclamation is attempted by the client after the NetApp storage system reclaims the locks from its partner. If locks cannot be reclaimed after a grace period, the locks are discarded, and the client reestablishes a new lock.

With NFSv4 the chances of these events occurring is remote because NFSv4 is a stateful protocol. Therefore, all state information is stored on both the client and the server when they are active and recovered mutually in the event of an outage. In NFSv4, ***nfsd*** is the only daemon required to start the nfs service. Ancillary protocols such as **portmapd**, **mountd, lockd,** and **statd** are no longer present. With the elimination of these adjunct protocols, the locking mechanism is streamlined, and applications face fewer challenges when recovering locks on startup.

NFSv4 locks provide a time-bounded grant of control over file state to an NFS client. During the lease interval, the NFS server may not grant conflicting control to another client. Holding a lease allows a client to assume that its lock will remain valid for a server-specified, renewable time interval. The client is responsible for contacting the NFS server to refresh the lease to maintain a lock every at the end of every lease interval. The lease interval defaults to 30 seconds in NetApp Data ONTAP.

Lease expiration is considered a failure in communication between the client and server, requiring recovery. The server assumes the client has failed and might allow other clients to acquire the same lock. If the NFS server fails, on reboot it waits the full lease interval for clients to reclaim locks before allowing new lock requests. Leases make sure of cache consistency and are kept short to prevent delays in normal operations. Longer lock lease intervals reduce lease refreshes.

Leases protect against loss of locking state by the client. A client normally exists in one of two states: either all locks are correct or all are lost. Refresh of any lock by a client validates all locks held by the client. This reduces the number of lease refreshes by a client from one per lock each lease interval to one per client each lease interval.

Compound Operations

Unlike NFSv3, where multiple NFS remote procedure calls (RPCs) occur for a single NFS operation, for example, READ or WRITE, NFSv4 uses a new mechanism in which multiple RPCs for an NFS request are batched and executed as a single COMPOUND operation. This is designed to reduce the traffic on the network and to improve performance.

Aggressive Client-Side Caching with Delegations

The NFSv4 protocol introduced a delegation mechanism, which allows caching a file or part of the file locally by the client. There are two types of delegations: READ and WRITE. Delegations improve performance in certain scenarios but not for all workloads. A proper evaluation of the workload and the application behavior pattern must be done before using delegations. For example, if there are multiple writers to a single file, then the WRITE delegation might not be a good choice, whereas for read-intensive workloads the READ delegation would provide better performance. A READ delegation can be given out

to multiple clients with an OPEN for a READ as long as they have a callback path. A WRITE delegation, however, is only given out to one client at a time that is requesting OPEN for a WRITE. Generally speaking, delegation is used more for cache correctness than for performance.

Efficient Interoperability with CIFS

NFSv4 provides better interoperability with CIFS compared to NFSv3. CIFS and NFSv3 support ACLs, unlike NFSv3. NFSv4 ACLs are based on the NT model. Flags, access masks, and type generally map one to one to an NT ACL. Although NFSv4 does not offer one-to-one mapping with security ACLs (SACLs), a one-to-one mapping does occur with the discretionary ACLs (DACLs).

When READ requests come from users over NFSv3, NFSv4, or CIFS, the file is read from NetApp storage based on the permission set on that file and directory. WRITE operations are handled differently; When a user opens a file for WRITE operations over CIFS, the DENY mode is set to all other users who want to make changes to that file over NFSv3 and NFSv4. If the user has opened the file over NFSv4 for WRITE operations, a secondary user requesting to OPEN the file over CIFS is refused. This protects files in multiprotocol environments from being “stepped on” during write operations.

3.2 Comparing NFSv3 and NFSv4

As stated in the previous section, NFSv4 is significantly different from NFSv3. The following table provides a summary of the differences between the two versions.

Table 1) NFSv3 vs NFSv4 differences.

Features	NFSv3	NFSV4
Personality	Stateless	Stateful
Semantics	UNIX only	UNIX and Windows
Authentication	Weak (Auth_Sys)	Strong (Kerberos)
Identification	32-bit UID/GID	String based (xyz@netapp.com)
Permissions	UNIX based	Windows based
Transport protocol	UDP and TCP	TCP only
Caching	Ad hoc	File delegations

Refer to the NetApp Interoperability Matrix Tool (IMT) for client support details.

Planning and Implementation Considerations

NFSv4 introduces new advanced capabilities that make implementing NFSv4 more complex than previous NFS versions. Proper planning is recommended to make deployment less painful and ongoing management easier. NetApp recommends that customers plan and choose the elements of their infrastructure carefully before beginning the implementation. The following is a list of items that customers must consider.

- **User and group identification (UID/GID) implementation.** As stated in the previous section, NFSv4 does not support UID or GID; it supports string-based communication for users and groups between the server and the client. Therefore entry of users and groups must be in the `/etc/passwd` file on 7-Mode systems or local UNIX user entries in clustered Data ONTAP. The `/etc/passwd` file can also be leveraged in Network Information Service (NIS) or LDAP. This is mandatory with an NFSv4 environment. When using local files, the names must match exactly on the client/server, including

case sensitivity. If names do not match, the user will not map into the NFSv4 domain and will squash to the “nobody” user.

- **User authentication.** Customers must plan how users will be authenticated. There are two options, Kerberos and standard UNIX password authentication.
- **Directory and file access.** Customers must plan for access control on files and directories. Depending on their business needs, they can choose standard UNIX permissions or Windows style ACLs.
- **Shared file system namespace configuration.** Shared file system namespace allows storage administrators to present NFS exported file systems to NFS clients. Depending on their business needs, storage administrators can choose to make the server location transparent to their users, use distributed directory structures, or structure access controls grouped by organization or by role. For example, if /corporate is the main entry point and a storage administrator wants to keep finance and IT separated, then the entry points for those departments could become /finance and /IT because / is the pseudo file system entry point for the file system.
- **Consideration for high availability.** If high availability is a key business requirement, customers need to consider designing directory/identity service, authentication service, and file services for high availability. For example, if Microsoft Active Directory is used for LDAP, then multiple LDAP servers could exist and update using domain replication, which is native to Microsoft Active Directory.
- **Implementation of pseudo file.** In 7-Mode, if automounter is used with “/” as one of the mounts for NFSv3 that translates to `/vol/vol0`, client mounts will no longer have the same representation in NFSv4. Due to the pseudo file system as described later in this paper, the fsid will change as the request traverses from a pseudo file system to an active file system. This means that if the automount has an entry to mount “/” from the NetApp storage, it will mount “/” and no longer mount “`/vol/vol0`.” Therefore, these entries should be updated with the actual pathnames. In clustered Data ONTAP, all SVMs have a “/” entry point through vsroot. Therefore, the 7-Mode concepts of vol0 do not apply.
- **Single port configuration.** In automount maps, the following entry has to be specified because NFSv4 communicates only over port 2049:

```
-fstype=nfs4, rw, proto=tcp,port=2049
```

- **UDP mount to TCP mount conversion.** Because UDP is no longer supported in NFSv4, any existing mounts over UDP must be changed to TCP.
- **NFSv4 ACL configuration for POSIX ACLs.** NFSv4 only supports NFSv4 ACLs; not POSIX ACLs. Any data that is migrated from third-party storage must manually set NFSv4 ACLs if they are required.
- **Mounting NFSv4 file systems.** Mounting file systems over NFSv4 is not the same for all NFS clients. Clients will mount using different syntax based on what kernel is being used. Some clients will mount NFSv4 by default, so the same considerations must be used if NFSv3 is desired. Customers must plan appropriately to correctly mount file systems to the NFSv4 client.
- **NFSv3 and NFSv4 clients can coexist.** The same file system can be mounted over NFSv3 and NFSv4. However, any ACLs set on a file or directory from NFSv4 are enforced for NFSv3 mounts as well. Setting permissions on a folder with mode bits (rwx) will affect the NFSv4 style ACL. For example. if a directory is set to 777 permissions, then the NFSv4 ACL for “everyone” will be added. If a directory is set to 770, then “everyone” will be removed from the NFSv4 ACL. This can be prevented using ACL preservation options, described in section 5.
- **NFSv4 has a stateful personality, unlike NFSv3, which is stateless in nature.** There is a common misconception that if NFSv4 is stateful in nature like CIFS, some amount of disruption could occur during storage failover and giveback for NetApp storage controllers in an HA pair. However, with NFSv4, the session is connected at the application layer and not at the TCP layer. Lock states are stored in controller memory, which is not persistent. However, lock states are kept at both the controller and network level, so the application can send a request to the NetApp storage controller to make sure of a persistent connection. Therefore, when storage failover and giveback happen, the

break in the TCP connection and lock does not affect the NFSv4 session, and thus no disruption occurs.

4 NFSv4 Implementation in the NetApp Environment

4.1 Enabling NFSv4 on NetApp Storage

First and foremost, a valid NFS license must be applied on the controller.

NFSv4 on the NetApp storage controller is disabled by default. In 7-Mode, it is enabled with the following option:

```
filer> options nfs.v4.enable ON
```

In clustered Data ONTAP, the option is enabled using the `nfs server modify` command, specifying the SVM. NFSv3 and v4 are enabled or disabled individually for each SVM as required:

```
cluster::> nfs server modify -vserver vs0 -v4.0 enabled
```

4.2 Domain Configuration for AIX, Linux, and Solaris

In NFSv3, client-server communication happens using a numeric user ID and group ID (UID/GID). The client is responsible for mapping it back to the string representation: that is, the source of the GID/UID mapping specified in the `/etc/nsswitch.conf` file on the client. As an example, support user `root (uid = 0, gid = 0)` sends a CREATE request for file "foo" to the NetApp controller. The UID and the GID are contained in the (RPC) layer, and parameters for the CREATE procedure such as filename, attributes, and so on are embedded in the NFS layer. When the NetApp controller receives the CREATE request, it uses the UID and GID numbers from the RPC header and stores them with the new file "`foo`." After the file "`foo`" is created, the NetApp storage system returns the numeric UID and GID to every GETATTR request for that file from the client. The client then maps the corresponding UID and GID numbers that the NetApp system returns to its respective owner and group names after consulting its `/etc/nsswitch.conf` file for appropriate sources. Therefore, even if there is no `/etc/passwd` or `/etc/group` file on the NetApp 7-Mode system or local UNIX users and groups in clustered Data ONTAP systems, the mapping of the owner and group IDs to their respective string names still succeeds.

NFSv4 does not use UID/GID as the initial method of interaction between the client and the server by default. Instead there is string-based authentication between the client and the server: for example, john@lab.netapp.com, where "john" is the username and "lab.netapp.com" is the domain from which it is attempting to mount the file system over NFSv4. If the user originates from a subdomain, then all the domains and the subdomains should be listed in the `/etc/resolv.conf` file on the NetApp storage system for 7-Mode or specified in DNS entries per SVM for clustered Data ONTAP. With NFSv4, client-server communication can happen using either numeric UID/GID or in string format, thereby introducing mapping on the NetApp storage system as well as on the client. Controlling the use of the numeric GID/UID in NFSv4 can be done using the following option in 7-Mode:

```
filer> options nfs.v4.id.allow_numerics
```

In clustered Data ONTAP, this behavior is controlled at the NFS server level per SVM with the following:

```
cluster::> nfs server modify -vserver vs0 -v4-numeric-ids [enabled|disabled]
```

To use these options, a Linux client that supports numeric IDs with NFSv4 is required. Older Linux clients (such as RHEL 5.x) will not work with this option. The option only works with AUTH_SYS, as Kerberos security will require name services such as LDAP, which will map the user to an ID anyway. If the right set of Linux clients are used and the option is enabled, string mapping in NFSv4 will no longer be a requirement.

Best Practice

NetApp recommends the use of name mapping for client and storage system communications. In an NFSv4 environment, NetApp highly recommends that a Network Information Service (NIS) or a Lightweight Directory Access Protocol (LDAP) be configured to resolve the strings or user names. A local password file may be used if the number of users is smaller and the environment does not have an existing NIS or LDAP setup. Attempting to manage a large number of local users on multiple clients can result in mistaken user/group entries and access issues.

From the previous example, when a client tries to create the file "`foo`" on the NetApp system, it sends an OPEN operation to the system, instead of a CREATE. The RPC still contains the numeric UID and GID, and the NFS layer has an OPEN operation with the CREATE flag set to ON to create the file, as well as the FATTR parameter, which will contain root@nfsv4domain.netapp.com as the owner and system@nfsv4domain.netapp.com as the owner's group.

Note: @nfsv4domain.netapp.com is the default "nfsv4 id domain" setting used in this example.

Keep in mind that the NFSv4 ID domain name is a pseudo domain name to which both the client and the storage system need to agree before they can do most NFSv4 operations. The NFSv4 domain name could be different from the Network Information Service (NIS) or Domain Name System (DNS) domain name. It could be any string that both client and server NFSv4 implementations understand. If the NIS server is servicing name to ID mappings, then the NFSv4 domain would match the NIS server. If LDAP is in use, then the LDAP domain would become the NFSv4 domain.

For all the NFSv4 requests to be processed correctly, we need to make sure that the user name to user ID mappings are available to both the server and the client and match exactly, including case sensitivity.

Linux NFSv4 Domain Configuration

Note: All Linux clients use the same configuration file (`/etc/idmapd.conf`), and the file would look as follows:

```
# cat /etc/idmapd.conf
[General]
#Verbosity = 0
# The following should be set to the local NFSv4 domain name
# The default is the host's DNS domain name.
#Domain = local.domain.edu
Domain = nfsv4domain.netapp.com
```

After changes are made to the `idmapd.conf` file, the idmap process must be restarted to reload the configuration by executing the following command:

```
/etc/init.d/rpcidmapd restart
```

Solaris NFSv4 Domain Configuration

On the Solaris client edit the line shown in the file `/etc/default/nfs` to specify the NFSv4 environment's domain name:

"`Domain=nfsv4domain.netapp.com`".

After making the previous update, restart the idmapd process.

```
svcadm restart svc:/network/nfs/mapid
```

AIX NFSv4 Domain Configuration

On an AIX client, change the domain with the `chnfsdom` command with the appropriate domain information.

The idmapd process must be restarted to reload the configuration after changing the domain name with the following command:

```
# startsrc -s nfsrgyd
0513-059 The nfsrgyd Subsystem has been started. Subsystem PID is 13435084.
```

NetApp NFSv4 Domain Configuration

On the NetApp storage system, the same domain should be specified in the `nfs.v4.id.domain` option in 7-Mode. In clustered Data ONTAP, the NFSv4 ID domain is managed through the NFS server on a per-SVM basis:

```
cluster::> nfs server modify -vserver vs0 -v4-id-domain nfsv4domain.netapp.com
```

When NFSv4 domains are matched, the NetApp storage system maps incoming strings into numeric UIDs/GIDs; otherwise, `UID=65535/GID=65535` (nobody/nobody) is used. If the domain check is successful, the specified name server is consulted for mapping of the received string to numeric UID and GID. The name server does the mapping according to the `/etc/nsswitch.conf` file in 7-Mode and SVM nm-switch and ns-switch options in clustered Data ONTAP. Because ID domains match, it is assumed that the client and NetApp system will use the same source for this mapping, the same NIS or LDAP server, or the same set of passwd and group files.

Example:

Configuration is local files (/etc/passwd in 7-Mode, UNIX users in clustered Data ONTAP), and nfsv4domain.netapp.com is the NFSv4 domain string.

If a client makes a request for the user “root,” the request will arrive as root@nfsv4domain.netapp.com. The NetApp storage system will then check the NFSv4 domain in the name server specified for the user root@ nfsv4domain.netapp.com. Because local files are configured, that user must exist locally on the storage system, as the local storage system is considered the NFSv4 domain. After the user is found, the UID and GID are translated and returned to the client. If there is no match (for instance, if root does not exist, or if root is listed in the local files instead), then the user will get mapped to the “nobody” user specified in the `/etc/idmapd.conf` file on the client. If name strings are not desired, then there are storage side options to allow for numerical IDs in NFSv4.

7-Mode:

```
filer> options nfs.v4.id.allow_numerics [on|off]
```

Clustered Data ONTAP:

```
cluster::> set diag
cluster::*> nfs server modify -vserver vs0 -v4-numeric-ids [enabled|disabled]
```

4.3 NFSv4 Client Mount

This section includes some of the different mount options that are available when you mount the exported file system from NetApp storage for clients such as AIX, Solaris 10, SUSE/SLES, and CentOS/RHEL. These client types will be referred to in the rest of this document for configuring and supporting various different NFS features. The following assumes that the client has been properly configured to use NFSv4 and can find an appropriate NFSv4 ID domain. Generally this is set in the `/etc/idmapd.conf` file for most Linux clients. If the `/etc/idmapd.conf` file is not set, the client will default to its domain name setting, found using the `domainname` command.

NFSv4 with Solaris, RHEL/CentOS, and SUSE/SLES

Solaris 10 clients attempt to mount NFSv4 by default. The `rsize` and `wsize` parameters default to 64kB.

```
% mount 172.17.44.43:/vol/test_vol /mnt/b1
% mount | grep /mnt/b1
‘/mnt/b1 from 172.17.44.43:/vol/test_vol’
Flags:vers=4,proto=tcp,sec=sys,hard,intr,link,symlink,acl,rsize=65536,wsize=65536,
retrans=5,timeo=600
Attr cache:    acregmin=3,acregmax=60,acdirmin=30,acdirmax=60
```

RHEL/CentOS versions 5.3 through 5.8 clients mount NFSv4 with a `-t` option, and the `rsize` and `wsize` parameters default to 64kB. Starting with RHEL6.x, clients mount NFSv4 by default.

```
# mount -t nfs4 172.17.44.43:/vol/test_vol /mnt/b1
# mount | grep /mnt/b1
172.17.44.43:/vol/test_vol on /mnt/b1 type nfs4
(rw,vers=4,rsize=65536,wsize=65536,hard,intr,proto=tcp,timeo=600,retrans=3,sec=sys,addr=172.17.44
.43)
```

The newest SUSE defaults to NFSv4. If using an older SUSE/SLES client, mount NFSv4 with a `-t` option, and the `rsize` and `wsize` parameters default to 64kB.

```
# mount -t nfs4 172.17.44.43:/vol/test_vol /mnt/b1
# mount | grep /mnt/b1172.17.44.43:/vol/test_vol on /mnt/b1 type nfs4
(rw,vers=4,rsize=65536,wsize=65536,hard,intr,proto=tcp,timeo=600,retrans=3,sec=sys,addr=172.17.44
.43)
```

NFSv4 with AIX

AIX 6.1 and later versions are recommended for mounting file systems over NFSv4. Follow these steps on AIX to mount with NFSv4.

1. Set the appropriate domain for the NFSv4 environment. In this example the domain “nfsv4domain.netapp.com” is used.

```
# chnfsdom nfsv4domain.netapp.com
# chnfsdom
Current local domain: nfsv4domain.netapp.com
```

2. Check if all the NFS daemons are started and running in the background. In the following example none of the processes have started.

```
The AIX client is running on version 6.1
# oslevel -g
6.1.0.0

# lssrc -g nfs
Subsystem         Group            PID          Status
 biod             nfs                           inoperative
 nfsd             nfs                           inoperative
 rpc.mountd       nfs                           inoperative
 nfsrgyd          nfs                           inoperative
 gssd             nfs                           inoperative
 rpc.lockd        nfs                           inoperative
 rpc.statd        nfs                           inoperative
```

3. Execute the following command to start NFS processes. All the NFS processes are started in the background, including “nfsrgyd,” which is similar to idmapd in Solaris and Linux.

```
# startsrc -g nfs
0513-059 The biod Subsystem has been started. Subsystem PID is 12910622.
0513-059 The nfsd Subsystem has been started. Subsystem PID is 10420248.
0513-059 The rpc.mountd Subsystem has been started. Subsystem PID is 11337902.
0513-059 The nfsrgyd Subsystem has been started. Subsystem PID is 5767244.
0513-059 The gssd Subsystem has been started. Subsystem PID is 11599908.
0513-059 The rpc.lockd Subsystem has been started. Subsystem PID is 11599910.
0513-059 The rpc.statd Subsystem has been started. Subsystem PID is 10354746.
```

4. Verify all the NFS daemons are up and running.

```
# lssrc -g nfs
```

```
Subsystem         Group            PID          Status
 biod             nfs              12910622     active
 nfsd             nfs              10420248     active
 rpc.mountd       nfs              11337902     active
 nfsrgyd          nfs              5767244      active
 rpc.lockd        nfs              11599910     active
 rpc.statd        nfs              10354746     active
 gssd             nfs                           inoperative
```

5. Whenever the domain information is changed, stop and start the “nfsrgyd” daemon.

```
# stopsrc -s nfsrgyd
0513-044 The nfsrgyd Subsystem was requested to stop.

# lssrc -g nfs
Subsystem         Group            PID          Status
 biod             nfs              12910622     active
 nfsd             nfs              10420248     active
 rpc.mountd       nfs              11337902     active
 rpc.lockd        nfs              11599910     active
 rpc.statd        nfs              10354746     active
 nfsrgyd          nfs                           inoperative
 gssd             nfs                           inoperative

# nfsrgyd -f

# startsrc -s nfsrgyd
0513-059 The nfsrgyd Subsystem has been started. Subsystem PID is 13435084.

# lssrc -g nfs
Subsystem         Group            PID          Status
 biod             nfs              12910622     active
 nfsd             nfs              10420248     active
 rpc.mountd       nfs              11337902     active
 rpc.lockd        nfs              11599910     active
 rpc.statd        nfs              10354746     active
 nfsrgyd          nfs              13435084     active
 gssd             nfs                           inoperative
```

6. Verify the UID and GID of the user that will be mounting the file system over NFSv4. In this case the user is “root.” Notice that the UID is “root” and the GID is “system.”

```
# id
uid=0(root) gid=0(system)
groups=2(bin),3(sys),7(security),8(cron),10(audit),11(lp),204(sapinst)
```

7. On the NetApp storage, check NFSv4 and “id.domain” are enabled and set, respectively.
   In 7-Mode:

```
options nfs.v4.enable                on
options nfs.v4.id.domain             nfsv4domain.netapp.com
```

   In clustered Data ONTAP:

```
::*> nfs server show -vserver vs0 -fields v4.0,v4-id-domain
vserver v4.0    v4-id-domain
------- ------- ----------------------------
Vs0     enabled nfsv4domain.netapp.com
```

8. The `/etc/passwd` and `/etc/group` files (or corresponding name service such as NIS or LDAP) must be updated with the entry for “root” and “system,” for 7-Mode in AIX:

```
filer*> rdfile /etc/passwd
root:_J9..j4mwouOY5UinHuY:0:1::/:
pcuser::65534:65534::/:
nobody::65535:65535::/:
ftp::65533:65533:FTP Anonymous:/home/ftp:
root:x:0:0::/:
oracle:!:1005:600::/home/oracle:/usr/bin/ksh

filer*> rdfile /etc/group
```

```
system:!:0:root,pconsole,esaadmin
root:*:0:
daemon:*:1:
dba:!:600:oracle
```

Note: There is no need to set the /etc/passwd file in this manner when using clients other than AIX. The default /etc/passwd file is sufficient for other clients.

In clustered Data ONTAP, there are no local passwd or group files. Everything is stored in a database table. Users and groups can be created using the CLI.

```
cluster::> unix-user show -vserver vs0 -user root
  (vserver services unix-user show)

        Vserver: vs0
      User Name: root
        User ID: 0
Primary Group ID: 1
User's Full Name: -
```

```
cluster::> unix-group show -vserver vs0 -name system
  (vserver services unix-group show)

   Vserver: vs0
Group Name: system
  Group ID: 0
     Users: -
```

The following verification resolves the UID and GID with the correct names.

```
filer*> getXXbyYY getpwbyuid_r 0
pw_name = root
pw_passwd = _J9..j4mwouOY5UinHuY
pw_uid = 0, pw_gid = 1
pw_gecos =
pw_dir = /
pw_shell =

filer*> getXXbyYY getgrbygid 0
name = system
gid = 0

filer*> getXXbyYY getgrbygid 1
name = daemon
gid = 1
```

In clustered Data ONTAP, diag-level commands provide verification of UID and GID. The node is specified in these commands because the authentication daemon is node-specific. Traffic will pass through a physical port on a node, so the node that owns the desired data LIF will need to be specified.

```
cluster::> net int show -vserver vs0 -lif data -fields curr-node,curr-port
  (network interface show)
vserver lif  curr-node     curr-port
------- ---- ------------- ---------
vs0     data node1         e0a

cluster::> diag secd authentication translate -node node1 -vserver vs0 -uid 0
root

cluster::> diag secd authentication translate -node node1 -vserver vs0 -gid 0
system
```

9. Mount the file system over NFSv4. In this example volume `/vol/svl` is mounted over NFSv4. An entry in the `/etc/filesystems` makes this mount persistent.

```
# mount -o vers=4 172.17.44.104:/vol/svl /test1
# cd /test1
```

Note that the correct UID and GID are displayed after the mount is completed.

```
# ls -al
total 53625176
drwxr-xr-x   12 root     system         4096 Apr 13 16:02 .
drwxr-xr-x   56 5742     30             4096 Apr 19 11:56 ..
-rw-r--r--    1 root     system            0 Apr 13 11:20 aa
-rw-r--r--    1 root     system            0 Apr 13 11:23 bb
-rw-r--r--    1 root     system            0 Apr 13 16:02 cc
```

If the NFSv4 ID mapping is broken or the username does not map into the NFSv4 domain, the output would look similar to this. If this occurs, check the client system's syslog file for details.

```
# ls -al
total 53625176
drwxr-xr-x   12 nobody   system         4096 Apr 13 16:02 .
drwxr-xr-x   56 5742     30             4096 Apr 19 11:56 ..
-rw-r--r--    1 nobody   system            0 Apr 13 11:20 aa
-rw-r--r--    1 nobody   system            0 Apr 13 11:23 bb
-rw-r--r--    1 nobody   system            0 Apr 13 16:02 cc
```

Files created inherit the correct UID and GID from the parent directory.

```
# touch kk
# ls -l
total 53625160
-rw-r--r--    1 root     system            0 Apr 13 11:20 aa
-rw-r--r--    1 root     system            0 Apr 13 11:23 bb
-rw-r--r--    1 root     system            0 Apr 13 16:02 cc
-rw-r--r--    1 root     system            0 Apr 19 11:54 kk
```

Best Practice

NetApp recommends RHEL5 update 6 (2.6.18 kernel) or later for NFSv4 implementation for these reasons:

- Performance:
  - Clients do more work with fewer operations and invalidate the cache less often.
  - Clients move more data in fewer bytes and fewer CPU cycles.
  - There is a more efficient use of CPU and memory resources.
- ACL: nfsv4_acls are more compatible with 2.6.18 kernel (RHEL 5) and later.

Pseudo File System in NFSv4

Most client systems mount local disks or partitions on directories of the root file system. NFS exports are exported relative to root or "`/`." Early versions of Data ONTAP had only one volume, so directories were exported relative to root just like any other NFS server. As data requirements grew to the point that a single volume was no longer practical, the ability to create multiple volumes was added. Because users don't log directly into the NetApp storage system, there was no reason to mount volumes internally to the NetApp system. To distinguish between volumes in 7-Mode, the `/vol/volname` syntax was created. To maintain compatibility, support was kept for directories within the root volume to be exported without any such prefix, so `/home` is equivalent to `/vol/vol0/home`, assuming that `vol0` is the root volume, `/` is the physical root of the system, and `/etc` is for the configuration information.

NetApp storage systems running 7-Mode are among the few implementations, possibly the only one, that requires a prefix such as "/vol" before every volume that is exported. In some implementations, this

means that deployers can't simply drop the NetApp 7-Mode system into the place of an existing NFS server without changing the client mounts, depending how things are implemented in `/etc/vfstab` or automounter. In NFSv3, if the complete path from `/vol/vol0` is not used, and `<NetApp storage: />` is mounted, the mount point is `NetApp storage: /vol/vol0`. That is, if the path does not begin with `/vol` in NFSv3, then Data ONTAP assumes `/vol/vol0` is the beginning of the path. This does not get users into the desired areas of the NFS file system. In clustered Data ONTAP, there is no concept of /vol/vol0. Volumes are junctioned below the root of the SVM, and nested junctions are supported. Therefore, in NFSv3, there is no need to modify anything when cutting over from an existing NFS server. It simply works.

In NFSv4, if the complete path from `/vol/vol0` is not used, and `<NetApp storage: />` is mounted. That is considered the root of the pseudo file system and not `/vol/vol0`. Data ONTAP does not add `/vol/vol0` to the beginning of the path, unlike NFSv3. Therefore, if `<NetApp storage: /n/NetApp storage>` is mounted using NFSv3 and the same mount is mounted using NFSv4, a different file system would be mounted.

This is why Data ONTAP 7-Mode has the "`/vol`" prefix in the exported global namespace, and that feature represents an instance of the NFSv4 pseudo file system namespace. The traversal from the pseudo file system namespace to those of actual exported volumes is marked by a change in file system ID (fsid). In the Data ONTAP implementation of the NFSv4 pseudo file system, the paths "`/`" and "`/vol`" are always present and form the common prefix of any reference into the pseudo file system. Any reference that does not begin with "`/vol`" is invalid in 7-Mode.

In clustered Data ONTAP, the notion of a pseudo file-system integrates seamlessly with junction paths and the unified namespace, so no additional pathing considerations are needed when leveraging NFSv4.

The NFSv4 server has a known root file handle for the server's available exported file systems that are visible from this global server root, by means of ordinary NFSv4 operations (for example, LOOKUP, GETATTTR) used within the pseudo file system. As mentioned in section 3, the mountd protocol is not supported in NFSv4; it is replaced by PUTROOTFH, which represents `ROOT` all the time. PUTFH represents the location of the pointer in the directory tree under `ROOT`. When a request to mount a file system comes from the client, it traverses the pseudo file system (`/` and `/vol`) before it gets to the active file system. While traversing from the pseudo file system to the active file system, the FSID changes.

In clustered Data ONTAP there is a diag-level option on the NFS server to enable preservation of the FSID in NFSv4. This is on by default and should not be changed in most cases.

```
cluster::> set diag
cluster::*> nfs server modify -vserver vs0 -v4-fsid-change
```

The following occurs when mounting a file system over NFSv4 (see Figure 1 and Figure 2).

1. A request from the client (SETCLIENTID) is sent from the client to the server to establish its identity.
2. After the server ACKnowledges and the client's identity is verified, the server checks whether there is a CALLBACK from the client using a CB_NULL command. This is done to check whether the client is eligible to be granted a DELEGATION.
3. Then the client sends a COMPOUND operation that includes PUTROOTFH, LOOKUP of the path that is requested to be mounted, and GETFH (get a file handle) as a batch process to the server.
4. The server sends a filehandle (FH), and the mount process is complete. The COMPOUND operation reduces RPC calls during this mount operation.

Figure 1) Client request to mount a file system in NFSv4.

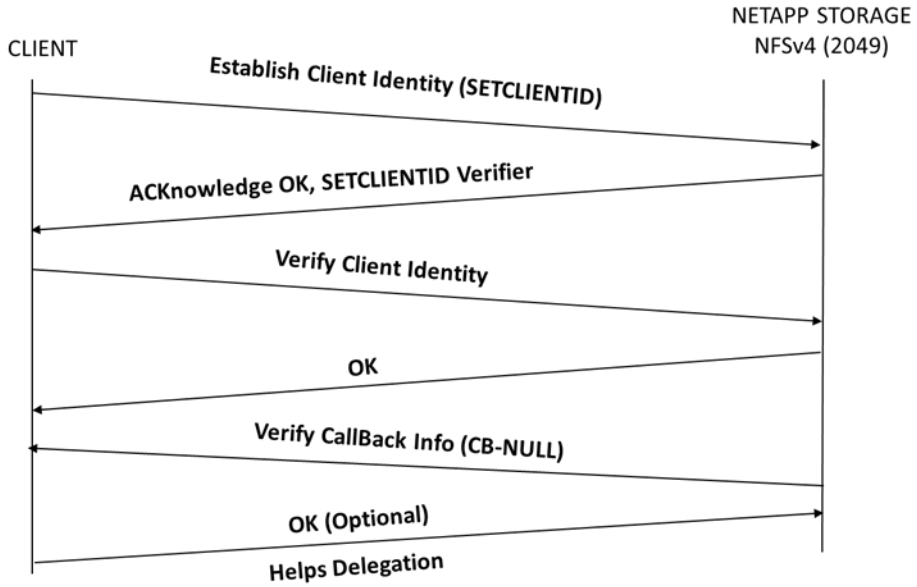
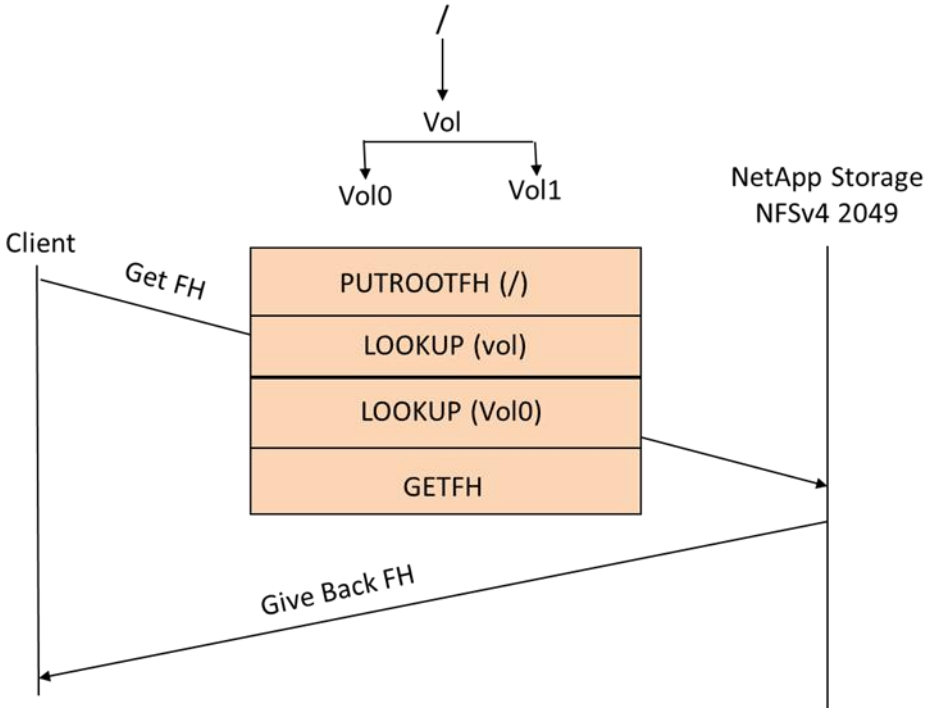


Figure 2) Server sends file handle to complete request.



4.4 Example of Pseudo File System in 7-Mode

The following example indicates how the fsid changes from a pseudo file system to an active file system in 7-Mode.

```
filer*> exportfs
/vol/test_vol/demo/test -sec=sys,rw,anon=0,nosuid
/vol/vol0/home  -sec=sys,rw,nosuid
/vol/vol0       -sec=sys,rw,anon=0,nosuid

/vol/test_vol/demo/test1 -sec=sys,rw,anon=0, nosuid
```

Compound LOOKUP for "/", "vol", generates a "special" (pseudo file system) fsid. "vol0" has a different fsid.

```
filer*> showfh4 -v /
/ (really /): exp.fileid=0x00042a exp.snapgen=0x000001 flags=0x05 snapid=0x0 unused=0x0
fileid=0x00042a gen=0x000001 fsid=0x000002 handle_type=0x03

filer*> showfh4 -v /vol
/vol (really /vol): exp.fileid=0x00042b exp.snapgen=0x000001 flags=0x05 snapid=0x0 unused=0x0
fileid=0x00042b gen=0x000001 fsid=0x000002 handle_type=0x02

filer*> showfh4 -v /vol/vol0
/vol/vol0 (really /vol/vol0): exp.fileid=0x000040 exp.snapgen=0x1da07d flags=0x00 snapid=0x0
unused=0x0 fileid=0x000040 gen=0x1da07d fsid=0xe802e1 handle_type=0x00
```

There will be a COMPOUND LOOKUP for "/", "vol", "test_vol", and "demo" that generates "special" (pseudo file system) fsids for each of them.

Automounter is the closest form of namespace implementation for NFSv3 and NFSv4. However, automounter cannot migrate files and locks, which is an important feature of a namespace. The NFSv4 RFC provides a namespace framework that represents all of the exported file systems. An NFSv4 client would perform a LOOKUP and a READDIR operation to parse though the exported file system. The exported file system could be a regular Data ONTAP volume or a qtree as listed in the following. The FSID changes when the client request traverses from the pseudo file system to the active or real file system.

```
/vol                          -pseudo filesystem
/vol/test_vol          -pseudo filesystem
/vol/test_vol/demo     -pseudo Filesystem
/vol/test_vol/demo/test       -real filesystem
```

Data ONTAP presents the namespace to the client and then performs a LOOKUP and a READDIR on the ROOT to traverse through the directory structure below it. The NetApp controller responds to the client with the file handle of the file system /vol/test_vol/demo/test. NFSv4 clients can detect export point crossing by changing the fsid attribute. For each of these operations, either a LOOKUP or a GETATTR operation is associated in order to check that every component of the path is either a file or a directory. There will be access checks for these components.

When an NFSv4 client tries to mount one of the exports from the NetApp storage system and the export is a subdirectory or a qtree of another export that does not grant the client permission to mount it, the request fails even though the higher level of the pseudo file system does grant the access. In the following case, the two exports are exported with the following permissions, where /vol/vol0 has more restrictive permissions than /vol/vol0/home:

```
        /vol/vol0     -rw=foo:bar
        /vol/vol0/home    -rw=gonzo
```

OR

```
        /vol/vol0    -sec=krb5,rw
        /vol/vol0/home -sec=sys,rw
```

Best Practice

As a best practice, NetApp does not recommend having clients with stricter permissions or security addressed in the pseudo file system or root path, compared with the descendant paths that are exported separately with weaker security. There is some call processing overhead if the client is denied permission at the top level for additional security checks while using nested exports with different or stricter exports with security policies at different subexport levels. This can lead to suboptimal performance. NetApp does not recommend the use of nested exports.

The Snapshot™ directory in Data ONTAP is also a pseudo file system. Unlike other NFSv4 clients, Solaris 10 cannot cross fsid change boundaries, and the content of the Snapshot directory is not visible unless explicitly mounted. However, to work around this problem, there was an option introduced in Data ONTAP 7.3 to address this issue. Data ONTAP 7.3 allows the NFSv4 clients to access nested exports even if the top level has restrictive permissions or security.filer> options nfs.v4.snapshot.active.fsid.enable.

The `-actual` option in `/etc/exports` on a 7-Mode NetApp storage system has been supported in NFSv4 since Data ONTAP 7.3.5. However, although `-actual="/xxx"` is supported, `-actual="/yyy/zzz"` is not. To get the same effect as `-actual="/yyy/zzz"`, create a symlink. For example, `exportfs -o actual=/yyy/zzz /vol/vol5/aaa/bbb/ccc` becomes `exportfs -o actual=/yyy /vol/vol5/aaa/bbb` and then in `/vol/vol5/aaa/bbb` create a symlink named zzz that links to ccc.

The –actual option is no longer necessary or available in clustered Data ONTAP.

4.5 Pseudo File Systems in Clustered Data ONTAP

Clustered Data ONTAP has removed the /vol requirement for exported volumes and instead uses a more standardized approach to the pseudo file system. Because of this, it is now possible to seamlessly integrate an existing NFS infrastructure with NetApp storage because “/” is now truly “/” and not simply a redirector to /vol/vol0 as it was in 7-Mode. Pseudo file system would only apply in clustered Data ONTAP if the permissions flow from more restrictive to less restrictive. For example, if the vsroot (mounted to /) has more restrictive permissions than a data volume does (such as /volname), then pseudo file systems would apply.

5 ACL Implementation with NFSv4

When a client sets an NFSv4 ACL on a file during a SETATTR operation, the NetApp storage system sets that ACL on the object, replacing any existing ACL. If there is no ACL on a file, then the mode permissions on the file are calculated from OWNER@, GROUP@, and EVERYONE@. If there are any existing SUID/SGID/STICKY bits on the file, they are not affected.

When a client gets an NFSv4 ACL on a file during the course of a GETATTR operation, the NetApp system reads the NFSV4 ACL associated with the object and constructs a list of ACEs and returns it to the client. If the file has an NT ACL or mode bits, then an ACL is constructed from mode bits and is returned to the client.

While doing a permission check for a CIFS user trying to access a file that has an NFSv4 ACL, the NT SID is converted into a UNIX ID, which is checked against the NFSv4 ACL. Similarly, when a UNIX user tries to access a file with an NT ACL, the UNIX ID is converted to an NT SID, which is checked against the NT ACL.

Access is denied if a DENY ACE is present in the ACL, and access is granted if an ALLOW ACE exists. However, access is also denied if neither of the ACEs is present in the ACL.

A security descriptor consists of a security ACL (SACL) and a discretionary ACL (DACL). When NFSv4 interoperates with CIFS, the DACL is one-to-one mapped with NFSv4 and CIFS. The DACL consists of the ALLOW and the DENY ACEs.

Beginning with Data ONTAP 7.3, the maximum number of ACEs in an ACL is increased from 192 to 400. The default number of ACEs is 192 in 7.3.5.x. However, the number of ACEs can be increased to 400 using `-options nfs.v4.acl.max.aces`. Data ONTAP 8.1.x 7-Mode supports more than 400 ACEs, as needed in certain customer scenarios. Clustered Data ONTAP 8.2 supports 1,024 ACEs. In that version, the number of max ACEs can be changed at the NFS server level.

```
cluster::> nfs server modify –vserver vs0 -v4-acl-max-aces [number up to 1024]
```

When reverting to a release older than 7.3, any NFSv4 ACE set on a file or directory with more than 192 ACEs will be dropped. After the reversion, files and directories that had more than 192 ACEs do not use ACLs for permission checking; rather, the mode bits are enforced.

In Data ONTAP 7.3.5, a new option was introduced to preserve the ACL when the mode bits are set. The ability to do this has existed in clustered Data ONTAP since NFSv4 support started in 8.1.x. This option is set to `off` by default. With the option set to `off`, the behavior of Data ONTAP NFSv4 server is to drop ACLs on a `chmod <mode bits>`.When the option is set to `on`, the NFSv4 server does not drop the ACL on a `chmod <mode bits>` and the following behavior occurs:

- For a SETATTR with mode bits, the NFSv4 ACL on the file/directory does not get dropped and is modified so as to conform to the new mode bits. Only the OWNER/GROUP/EVERYONE ACEs are modified according to this, and ACEs with individual user names are not modified.
- For a guarded mode OPEN of a file with mode bits, if the file has an inherited ACL, the inherited ACL gets modified to conform with the mode bits.
- For a `mkdir` (CREATE) with mode bits, the behavior is the same as described earlier.

The NFSv4 ACL implementation in Data ONTAP can be summarized as follows:

- Mode bits are a subset of the ACLs; ACLs are more expressive than the mode bits.
- If a file or directory has an ACL, the displayed mode bits are evaluated from the actual ACL on the file or directory.
- If an ACL is set on the file, all the permission checking is done against the ACL.
- When an ACL is set on a file or directory, the mode bits do not show the exact permissions on the file; rather, they indicate a close approximation.

For example, if the owner of the file or directory has “read” and “write” permission and another user has just the “read” ACE set on it, then listing the file or directory with `ls –l` might not display the complete set of permissions assigned to the other user.

NFSv4 ACLs can provide access to multiple groups. For additional information on the syntax to set multiple groups, see http://www.linuxcertif.com/man/1/nfs4_setfacl/145707/ and http://linux.die.net/man/5/nfs4_acl.

- Third-party applications such as Quest and Centrify can be used to map and centrally manage NFS and CIFS users and groups.
- An option exists in both 7-Mode and clustered Data ONTAP to allow the setting of mode bits while ACLs still exist on the file. As previously mentioned, when both mode bits and ACLs exist on a file or

directory, the permission checking is done against the ACLs.

In 7-Mode (hidden option):

```
filer> options nfs.v4.acl_preserve
```

In clustered Data ONTAP (diag-level option):

```
cluster::> set diag
cluster::*> nfs server modify -vserver vs0 -v4-acl-preserve [enabled|disabled]
```

5.1 ACL Preservation in Action

This is a newly created UNIX style volume:

```
filer> fsecurity show /vol/unix
[/vol/unix - Directory (inum 64)]
  Security style: Unix
  Effective style: Unix

  DOS attributes: 0x0010 (----D---)

  Unix security:
    uid: 0 (root)
    gid: 0 (root)
    mode: 0755 (rwxr-xr-x)

  No security descriptor available.
```

In the preceding example, the volume (/vol/unix) has 755 permissions. That means the owner has ALL access, the owning group has READ/EXECUTE access, and everyone else has READ/EXECUTE access.

Even though there are no NFSv4 ACLs in the fsecurity output, there are default values set that can be viewed from the client:

```
[root@centos6 nfsv4]# nfs4_getfacl /nfsv4
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A:g:GROUP@:rxtncy
D:g:GROUP@:waDTC
A::EVERYONE@:rxtncy
D::EVERYONE@:waDTC
```

The preceding NFSv4 ACLs show the same: the owner has ALL access, the owning group has READ/EXECUTE access, and everyone else has READ/EXECUTE access. The default mode bits are tied to the NFSv4 ACLs.

When mode bits are changed, the NFSv4 ACLs are also changed:

```
[root@centos6 /]# chmod 775 /nfsv4
[root@centos6 /]# ls -la | grep nfsv4
drwxrwxr-x.   3 root     root           4096 Apr 29 14:00 nfsv4
[root@centos6 /]# nfs4_getfacl /nfsv4
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A:g:GROUP@:rwaDxtTnNcy
D:g:GROUP@:C
A::EVERYONE@:rxtncy
D::EVERYONE@:waDTC
```

When a user ACE is added to the ACL, the entry is reflected in the ACL on the storage system. In addition, the entire ACL is now populated:

```
[root@centos6 /]# nfs4_setfacl -a A::ldapuser@parisi2k8.ngslabs.netapp.com:ratTnNcCy /nfsv4
```

```
[root@centos6 /]# nfs4_getfacl /nfsv4
A::ldapuser@parisi2k8.ngslabs.netapp.com:ratTnNcCy
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A:g:GROUP@:rwaDxtTnNcy
D:g:GROUP@:C
A::EVERYONE@:rxtncy
D::EVERYONE@:waDTC

filer> fsecurity show /vol/unix
[/vol/unix - Directory (inum 64)]
  Security style: Unix
  Effective style: Unix

  DOS attributes: 0x0010 (----D---)

  Unix security:
    uid: 0 (root)
    gid: 0 (root)
    mode: 0775 (rwxrwxr-x)

  NFSv4 security descriptor:                  ------------
    DACL:                                                 |
      Allow - uid: 55(ldapuser) - 0x0016019d              |
      Allow - OWNER@ - 0x001601ff                         |
      Deny  - OWNER@ - 0x00000000                         |
      Allow - GROUP@ - 0x001201ff                         |- Note the ACL is now populated
      Deny  - GROUP@ - 0x00040000                         |
      Allow - EVERYONE@ - 0x001200a9 (Read and Execute)   |
      Deny  - EVERYONE@ - 0x00040146                      |
                                                ------------
```

When a change is made to the mode bit when NFSv4 ACLs are present, the volume ACL will get wiped by default:

```
[root@centos6 /]# nfs4_getfacl /nfsv4
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A:g:GROUP@:rxtncy
D:g:GROUP@:waDTC
A::EVERYONE@:rxtncy
D::EVERYONE@:waDTC

filer> fsecurity show /vol/unix
[/vol/unix - Directory (inum 64)]
  Security style: Unix
  Effective style: Unix

  DOS attributes: 0x0010 (----D---)

  Unix security:
    uid: 0 (root)
    gid: 0 (root)
    mode: 0755 (rwxr-xr-x)

  No security descriptor available.
```

This can be controlled in 7-Mode and clustered Data ONTAP.

In 7-Mode (hidden option):

```
filer> options nfs.v4.acl_preserve
```

In clustered Data ONTAP (diag-level option):

```
cluster::> set diag
cluster::*> nfs server modify -vserver vs0 -v4-acl-preserve [enabled|disabled]
```

After the option is enabled, the ACL will stay intact when mode bits are set. It is a best practice to make sure this is set when NFSv3 coexists with NFSv4 ACLs.

```
[root@centos6 /]# nfs4_setfacl -a A::ldapuser@parisi2k8.ngslabs.netapp.com:ratTnNcCy /nfsv4
[root@centos6 /]# nfs4_getfacl /nfsv4
A::ldapuser@parisi2k8.ngslabs.netapp.com:ratTnNcCy
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A:g:GROUP@:rxtncy
D:g:GROUP@:waDTC
A::EVERYONE@:rxtncy
D::EVERYONE@:waDTC

filer> fsecurity show /vol/unix
[/vol/unix - Directory (inum 64)]
  Security style: Unix
  Effective style: Unix

  DOS attributes: 0x0010 (----D---)

  Unix security:
    uid: 0 (root)
    gid: 0 (root)
    mode: 0755 (rwxr-xr-x)

  NFSv4 security descriptor:
    DACL:
      Allow - uid: 55(ldapuser) - 0x0016019d
      Allow - OWNER@ - 0x001601ff
      Deny  - OWNER@ - 0x00000000
      Allow - GROUP@ - 0x001200a9 (Read and Execute)
      Deny  - GROUP@ - 0x00040146
      Allow - EVERYONE@ - 0x001200a9 (Read and Execute)
      Deny  - EVERYONE@ - 0x00040146

[root@centos6 /]# chmod 775 /nfsv4
[root@centos6 /]# ls -la | grep nfsv4
drwxrwxr-x.   3 root     root           4096 Apr 29 14:00 nfsv4
```

Note the ACL is still intact:

```
[root@centos6 /]# nfs4_getfacl /nfsv4
A::ldapuser@parisi2k8.ngslabs.netapp.com:ratTnNcCy
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A:g:GROUP@:rwaxtTnNcy
D:g:GROUP@:DC
A::EVERYONE@:rxtncy
D::EVERYONE@:waDTC

filer> fsecurity show /vol/unix
[/vol/unix - Directory (inum 64)]
  Security style: Unix
  Effective style: Unix

  DOS attributes: 0x0010 (----D---)

  Unix security:
    uid: 0 (root)
    gid: 0 (root)
    mode: 0775 (rwxrwxr-x)

  NFSv4 security descriptor:
    DACL:
      Allow - uid: 55(ldapuser) - 0x0016019d
      Allow - OWNER@ - 0x001601ff
      Deny  - OWNER@ - 0x00000000
      Allow - GROUP@ - 0x001201bf
      Deny  - GROUP@ - 0x00040040
      Allow - EVERYONE@ - 0x001200a9 (Read and Execute)
      Deny  - EVERYONE@ - 0x00040146
```

ACLs can be set on the root of the directory or qtree (parent directory) `/vol/vol2/users/*`, and those ACLs will be propagated to all the subdirectories such as “Tom” and “Joe.” Separate or different ACEs cannot be set on “Tom” and “Joe” compared to the rest of the users in the directory. If you want to set different ACEs on different users, then you might not be able to set any ACE on the parent directory, in this instance `/vol/vol2/users`. Refer to the following link for information on “inheritance flags” on files and directories, particularly “directory inherit” and “file inherit”: http://linux.die.net/man/5/nfs4_acl.

5.2 Enabling and Disabling NFSv4 ACLs

In 7-Mode, use the `nfs.v4.acl.enable` option (disabled by default) to control the setting and viewing of NFSv4 ACLs.

```
filer> options nfs.v4.acl.enable on | off [off]
```

Note: The `nfs.v4.acl.enable` option does not affect whether an ACL is enforced and does not affect existing ACLs. The ACL will be enforced independent of the option value. The root user always has precedence over any ACL set on a file or directory.

In clustered Data ONTAP:

```
cluster::> nfs server modify -vserver vs0 -v4.0-acl [enabled|disabled]
```

5.3 Setting or Modifying an NFSv4 ACL

Setting NFSv4 style ACLs will depend on the kernel being used.

For Solaris:

Use the `setfacl` command to set or modify an NFSv4 ACL.

Use the `getfacl` command to view an NFSv4 ACL.

Note: Some Solaris versions may use the nfs4_setfacl and nfs4_getfacl commands.

For Linux:

Use `nfs4_setfacl` command to set or modify an NFSv4 ACL.

Use the `nfs4_getfacl` command to view an NFSv4 ACL.

Note: When setting an NFSv4 ACL, it might be easiest to use the –e option with the ACl command, as that will open the ACL in a text editor (such as vi) and allow easier editing of NFSv4 ACLs.

5.4 ACL Set/Modify Examples

Solaris 10 Update 1 (Generic_118844-26 i86pc i386 i86pc)

To set an NFSv4 ACL:

```
bash-3.00$ touch drum

bash-3.00$ ls -l
total 231140464
-rw-r--r--   1 blue     users          0 Apr 11  2007 drum

bash-3.00$ setfacl -m u:blue:rwx /mnt/b1/drum
```

To view an NFSv4 ACL:

```
bash-3.00$ getfacl drum

# file: drum
# owner: blue
# group: users
```

```
user::rw-
user:blue:rwx           #effective:rwx
group::r--              #effective:r--
mask:rwx
other:r—

bash-3.00$ ls -l
total 231140464
-rw-r--r--+  1 blue     users          0 Apr 11 14:03 drum
```

Solaris 10 Update 3 (Sun OS Generic_118855-33 i86pc i386 i86pc)

Starting with Solaris 10 update 3, `chmod` and "`ls -v`" (or "`ls -V`") are the preferred ways to set and get ACLs. These commands will handle both POSIX-draft and real NFSv4 ACLs. GETFACL and SETFACL are still there, and they are still limited to POSIX-draft, but in the future they will be removed.

Here are the file permissions on file `foo2` before setting an ACL:

```
bash-3.00$ ls -v foo2
-rw-r--r--   1 green    users         0 Apr 19 13:33 foo2
     0:owner@:read_data/write_data/append_data/read_xattr/write_xattr
         /read_attributes/write_attributes/read_acl/write_acl/synchronize
         :allow
     1:owner@:execute/write_owner:deny
     2:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
     3:group@:write_data/append_data/write_xattr/execute/write_attributes
         /write_acl/write_owner:deny
     4:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
     5:everyone@:write_data/append_data/write_xattr/execute/write_attributes
         /write_acl/write_owner:deny
bash-3.00$ ls -V foo2
-rw-r--r--   1 green    users         0 Apr 19 13:33 foo2
            owner@:rw-p--aARWcC-s:------:allow
            owner@:--x---------o-:------:deny
            group@:r-----a-R-c--s:------:allow
            group@:-wxp---A-W-Co-:------:deny
          everyone@:r-----a-R-c--s:------:allow
          everyone@:-wxp---A-W-Co-:------:deny
```

Replacing ACLs entirely for user "`green`" on file `foo2`.

```
bash-3.00$ chmod A=owner@:read_data/write_data:allow,group@:read_data
/write_data:allow,user:green:read_data:allow foo2

bash-3.00$ ls -V foo2
-rw-rw----+  1 green    users         0 Apr 19 13:33 foo2
            owner@:rw------------:------:allow
            group@:rw------------:------:allow
         user:green:r-------------:------:allow

bash-3.00$ ls -v foo2
-rw-rw----+  1 green    users         0 Apr 19 13:33 foo2
     0:owner@:read_data/write_data:allow
     1:group@:read_data/write_data:allow
     2:user:green:read_data:allow
```

Linux CentOS/RHEL5.x or CentOS RHEL6.x

Linux 2.6.9-42 (RHEL 4.4) is not completely ready to support NFSv4 ACLs and delegations. Therefore testing was done on RHEL5.7 and RHEL6.1. These releases work perfectly with NFSv4 ACLs.

Check the man pages for `nfs4_setfacl` (`nroff -man /usr/local/man/man1/nfs4_getfacl.1 |less`) for more examples of using this command:

```
[root@ibmx335-svl47 ~]# nfs4_setfacl -e /mnt/b3/testfile1
A::OWNER@:rwatTnNcCy
D::OWNER@:x
```

```
A:g:GROUP@:rtncy
D:g:GROUP@:waxTC
A::EVERYONE@:rtncy
D::EVERYONE@:waxTC

[root@ibmx335-svl47 b3]# nfs4_getfacl typescript.old
A::OWNER@:rwatTnNcCy
D::OWNER@:x
A:g:GROUP@:rtncy
D:g:GROUP@:waxTC
A::EVERYONE@:rtncy
D::EVERYONE@:waxTC
```

If clients are running on the latest kernel, an ACL set on a file from a Solaris client can be read from Linux and vice versa. The NFS clients depend on RPC for authentication using AUTH_SYS to identify users. There is a limitation on the NFS clients that any user can be part of 16 groups. However, the limit can be eliminated with the `rpcsecgssd` service that Kerberos uses. The number of groups in which the user can be part in 7-Mode is now 256. In clustered Data ONTAP, that current limit is 32.

AIX 6.1

AIX 6.1 supports NFSv4 ACLs. The file system that needs to be exported over NFSv4 must be included in the `/etc/filesystems` with “acl” and “ea=v2” in the mount options.

```
/test1:
        dev             = /vol/svl
        vfs             = nfs
        nodename        = 172.17.44.104
        mount           = true
        options         =
rw,bg,hard,intr,acl,grpid,rsize=65536,wsize=65536,timeo=600,vers=4,proto=tcp,sec=sys
        account         = false
        ea              = v2
```

If the file system is already mounted, it has to be unmounted and remounted when ACL is enabled on the NetApp storage. The `aclgettypes` command helps to verify the supported ACL version. In this example it is NFSv4.

```
# umount -f /test1
forced unmount of /test1
# mount /test1
# mount
172.17.44.104 /vol/svl         /test1           nfs4   Apr 24 11:51
rw,bg,hard,intr,acl,grpid,rsize=65536,wsize=65536,timeo=600,vers=4,proto=tcp,sec=sys

# aclgettypes /test1
Supported ACL type is:
        NFS4
```

After ACL support is enabled on the storage and the AIX client, the `aclget` command can be used to list the existing ACL on a file “`kk`” that was created earlier.

```
# aclget kk
*
* ACL_type   NFS4
*
*
* Owner: root
* Group: system
*
s:(OWNER@):     a       rwpRWaAcCs
s:(OWNER@):     d       xo
s:(GROUP@):     a       rRacs
s:(GROUP@):     d       wpWxACo
s:(EVERYONE@):  a       rRacs
s:(EVERYONE@):  d       wpWxACo
```

Setting an ACL can be done in two ways: `acledit` and `aclput`. In the following example, `acledit` is used to change the ACL on a file "`kk`".

```
# export EDITOR=/usr/bin/vi
# acledit kk
*
* ACL_type   NFS4
*
*
* Owner: root
* Group: system
*
s:(OWNER@):     a       rwpRWaAcCs
s:(OWNER@):     d       xo
s:(GROUP@):     a       rRacs
s:(GROUP@):     a       wpWxACo
s:(EVERYONE@):  a       rRacs
s:(EVERYONE@):  d       wpWxACo

Should the modified ACL be applied? (yes) or (no)
#
# aclget kk
*
* ACL_type   NFS4
*
*
* Owner: root
* Group: system
*
s:(OWNER@):     a       rwpRWaAcCs
s:(OWNER@):     d       xo
s:(GROUP@):     a       rRacs
s:(GROUP@):     a       wpWxACo
s:(EVERYONE@):  a       rRacs
s:(EVERYONE@):  d       wpWxACo
```

5.5 Interaction of Mode Bits, NFSv4 ACLs, and NT ACLs in Different Qtree Security Styles

In 7-Mode and clustered Data ONTAP, qtrees can have three different types of security styles: UNIX, NTFS, or mixed. The security style used will depend on the environment.

Best Practice

As a best practice for environments heavy in UNIX using NFSv4, NetApp recommends using UNIX style qtrees and volumes. However, NTFS security-style qtree may be used in some scenarios when the application depends on being able to read an ACL accurately from a file when the ACL could have been set from the other protocol.

UNIX Qtree

- Only mode bits and NFSv4 ACLs are considered for permission checking; NT ACLs are ignored. CIFS is not allowed to change permissions on a file.
- The ACL is evaluated to see if the user can do a `chmod`. Assuming that the user has the correct permission, the mode bits are set. Because an ACL is more expressive than mode bits, those new mode bits are represented as ACEs, and the existing ACL is accordingly modified (@OWNER @GROUP @EVERYONE ACEs in it are either added, dropped, or changed). The new mode bits that are seen are just an expression of the modified ACL. If the new mode bits are set correctly, the ACL should work exactly the way the user has specified with `chmod`. Thereafter, any subsequent file access is still checked against the ACL as it normally would be. This behavior can be controlled using ACL preservation as detailed in section 5.1.

- If a file has an NFSv4 ACL, and a UNIX client does a `chmod` to do a SUID/SGID/STICKY operation, the ACL is not dropped, and these special permission bits are added to the mode. Similarly, if a `chmod` is done to remove only these permissions, the ACL is not dropped, and the mode bits are modified accordingly.
- If a file has an NFSv4 ACL and a UNIX client does a `chown` or `chgrp`, the ACL is not affected.
- If a file has an NT ACL (this can happen if the qtree security has been changed from MIXED/NTFS to UNIX), mode bits will be returned upon a GETATTR request. Similarly, a fake NFSv4 ACL will be created from the mode bits upon a get ACL request.
- If a file has mode bits and a get ACL request is received, a fake NFSv4 ACL is created from the mode bits and returned to the client.

NTFS Qtree

- Only mode bits and NT ACLs are considered for permission checking; NFSv4 ACLs are ignored.
- NFS is not allowed to change permissions on group, owner, or ACL for a file, as access will be controlled by NTFS ACLs and UNIX to Windows user mapping.
- If a file has an NT ACL and NFSv4 reads the ACL, the mode bits will be converted to an NFSv4 ACL and returned to the client. Data ONTAP does not currently support translation of an NT ACL into an NFSv4 ACL.
- If a file has an NFSv4 ACL, and a get ACL request comes in from CIFS, the NFSv4 ACL will be mapped to an NT ACL and will be returned. The ACL displayed on the NT side might be incorrect because there is no mapping from NT groups to UNIX groups. Any such ACEs will not be returned to the NT client as part of the ACL. Also, any user ACEs where the UID could not be mapped to an NT SID are not returned.
- If a file has an NT ACL and a UNIX client sends a GETATTR request to get the mode permissions, the mode permissions calculated at the time of setting the ACL are returned.
- If a file doesn't have an ACL and a CIFS or NFSv4 client makes a get ACL request, the mode bits are converted into a fake ACL and returned to the client.
- If a CIFS user tries to access a file that does not have an ACL, the checking is done by using mode bits. This is done by mapping the SID to a UNIX UID and then authenticating the UID against the mode bits.

Mixed Qtree

- A file will have either mode bits, or mode bits and an NFSv4 ACL, or mode bits and an NT ACL.
- If a file has mode bits and either NFSv4 or CIFS has set an ACL, a set of "display" mode bits is created from the ACL. However, the permission checking is done from the ACL.
- If a file has an NFSv4 ACL and a UNIX client tries to do a chmod, the ACL is not dropped. The special bits (SUID/SGID/STICKY) on the file remain intact. Note that changing the SUID/SGID/STICKY bits does not cause the ACL to be dropped.
- If a file has an NT ACL and an NFSv4 client makes a GETATTR request, the NT ACL is not converted to an NFSv4 ACL; an ACL is calculated from the mode bits on the file and returned to the client.
- If a file has an NFSv4 ACL and a CIFS client tries to read the ACL, the NFSv4 ACL is converted to an NT ACL and returned. The ACL displayed on the NT side might be incorrect because there is no mapping from NT groups to a UNIX group. Any such ACEs will not be returned to the NT client as part of the ACL. Also, any user ACEs where the UID could not be mapped to an NT SID are not returned.

Mixed Security Style Considerations

Mixed qtree styles can cause issues with permissions if not set up properly. It can also be confusing to know what permissions are set on a file or folder when using mixed security style, as the NFS or CIFS

clients might not display the ACLs properly. Mixed security style can get messy when clients are modifying permissions, even with identity management in place. Therefore, it's best practice to choose either NTFS or UNIX security style unless there is a specific recommendation from an application vendor to use mixed mode. If an application attempts to set the ACL on another file based on this displayed ACL information, it would actually be setting an incomplete ACL and not the same one as on the original file.

- For any NT user, the user's SID is mapped to a UNIX ID, and the NFSv4 ACL is then checked for access for that UNIX ID. Regardless of which permissions are displayed, the actual permissions set on the file take effect and are returned to the client.
- If a file has an NT ACL and a UNIX client does a `chmod`, `chgrp`, or `chown`, the NT ACL is dropped.

To list the actual permissions on the file, the `fsecurity show` command on the NetApp storage system running 7-Mode provides accurate information.

```
Filer> fsecurity show /vol/volname
```

In clustered Data ONTAP 8.1.x and prior, run the following command on the node that owns the volume:

```
cluster::> node run –node nodename “fsecurity show /vol/volname”
```

In clustered Data ONTAP 8.2 and later, use the following command:

```
cluster::> vserver security file-directory show -vserver vs0 -path /junction-path
```

The following example shows how NTFS and NFSv4 ACLs interact on a mixed style volume in both 7-Mode and clustered Data ONTAP.

Mixed Qtree Example: 7-Mode

The following is a newly created volume using mixed security. Note there are a "security style" and an "effective style":

```
filer> fsecurity show /vol/mixed
[/vol/mixed - Directory (inum 64)]
  Security style: Mixed
  Effective style: Unix

  DOS attributes: 0x0010 (----D---)

  Unix security:
    uid: 0 (root)
    gid: 0 (root)
    mode: 0755 (rwxr-xr-x)

  No security descriptor available.
```

Mixed security style volumes will leverage a "last ACL modified" model to determine the effective security style of the volume.

In the following example, an ACL is added to the mixed security style volume using “iCACLS.exe” in Windows. After an ACL is set on a folder, volume, or file, the effective security style changes for that object to the style of the client making the change.

```
C:\>net use Z: \\filer\mixed
The command completed successfully.

C:\>net use
New connections will be remembered.

Status       Local     Remote                    Network
-------------------------------------------------------------------------------
```

```
OK           Z:        \\filer\mixed       Microsoft Windows Network
The command completed successfully.

C:\>z:

Z:\>

C:\>icacls Z:
Z: DOMAIN\Administrator:(OI)(IO)(F)
   DOMAIN\Administrator:(CI)(F)
   Everyone:(OI)(IO)(GR,GE)
   Everyone:(RX)

Successfully processed 1 files; Failed processing 0 files


C:\>icacls Z: /grant ldapuser:f
processed file: Z:
Successfully processed 1 files; Failed processing 0 files

C:\>icacls Z:
Z: DOMAIN\ldapuser:(F)
   DOMAIN\Administrator:(OI)(IO)(F)
   DOMAIN\Administrator:(CI)(F)
   Everyone:(OI)(IO)(GR,GE)
   Everyone:(RX)

Successfully processed 1 files; Failed processing 0 files
```

“ldapuser” is the user being added to this share. This user exists in the LDAP server and maps to UID 55:

```
filer> wcc -s ldapuser
(NT - UNIX) account name(s):  (DOMAIN\ldapuser - ldapuser)
        ***************
        UNIX uid = 55
        user is a member of group Domain Users (513)
        user is a member of group unixadmins (503)

        NT membership
                DOMAIN\ldapuser
                DOMAIN\Domain Users
                DOMAIN\unixadmins
                BUILTIN\Users
        User is also a member of Everyone, Network Users,
        Authenticated Users
        ***************
```

The ACL is updated on the storage system and can be confirmed by using `fsecurity`. Note the following changes:

- The "effective style" is now NTFS.
- The GID owner is now 1 (daemon) instead of 0 (root).
- The bit mask has changed from 755 to 777.
- There is now a list of NTFS security descriptors:

```
filer> fsecurity show /vol/mixed
[/vol/mixed - Directory (inum 64)]
  Security style: Mixed
  Effective style: NTFS  <---- effective style changes

  DOS attributes: 0x0030 (---AD---)

  Unix security:
    uid: 0 (root)
```

```
    gid: 1 (daemon) <-- GID changes
    mode: 0777 (rwxrwxrwx) <-- bit mask changes

  NTFS security descriptor: <--- List of NTFS style security descriptors
    Owner: BUILTIN\Administrators
    Group: DOMAIN\Domain Users
    DACL:
      Allow - DOMAIN\ldapuser - 0x001f01ff (Full Control)
      Allow - DOMAIN\Administrator - 0x10000000 - OI|IO
      Allow - DOMAIN\Administrator - 0x001f01ff (Full Control) - CI
      Allow - Everyone - 0xa0000000 - OI|IO
      Allow - Everyone - 0x001200a9 (Read and Execute)
```

From an NFS client, the permissions look like this:

```
[root@centos6 /]# mount -t nfs4 10.61.84.240:/vol/mixed /mixed
[root@centos6 /]# ls -la | grep mixed
drwxr-xr-x.   3 root     daemon         4096 Apr 29 15:26 mixed
```

Note that the mode bits are still 755, even though the storage system says the permissions are 777. This is due to the option nfs.ntacl_display_permissive_perms. This is set to "off" by default. After the option is set to "on" then the permissions match:

```
filer> options nfs.ntacl_display_permissive_perms on

[root@centos6 /]# umount /mixed
[root@centos6 /]# mount -t nfs4 10.61.84.240:/vol/mixed /mixed
[root@centos6 /]# ls -la | grep mixed
drwxrwxrwx.   3 root     daemon         4096 Apr 29 15:26 mixed
```

Because root doesn't map to a user or belong to a group that has NTFS permissions to the mount, and because the effective permissions on the volume are currently NTFS, root cannot access the mount or see permissions:

```
[root@centos6 /]# nfs4_getfacl /mixed
Failed getxattr operation: Permission denied

[root@centos6 /]# cd /mixed
-bash: cd: /mixed: Permission denied
```

However, “ldapuser,” which *does* map to a valid Windows user with NTFS access to the volume, can see the NFSv4 ACLs and access the mount:

```
[root@centos6 /]# su ldapuser
sh-4.1$ nfs4_getfacl /mixed
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A:g:GROUP@:rxtncy
D:g:GROUP@:waDTC
A::EVERYONE@:rxtncy
D::EVERYONE@:waDTC

sh-4.1$ cd /mixed
sh-4.1$ ls -la
total 12
drwxrwxrwx.  3 root     daemon       4096 Apr 29 15:42 .
dr-xr-xr-x. 28 root     root         4096 Apr 29 15:24 ..
drwxrwxrwx.  2 root     root         4096 Apr 29 15:26 .snapshot
-rwxrwxrwx.  1 ldapuser Domain Users    0 Apr 29 15:42 newfile
-rwxrwxrwx.  1 ldapuser Domain Users    0 Apr 29 15:42 newfile2
```

After a user with permissions to change ACLs adds an NFSv4 ACL, the following occurs:

- The effective style changes back to UNIX.
- The NTFS ACLs get wiped, but an NFSv4 ACL is added.
- The GID owner stays 1 (daemon).

- The NTFS owner/group gets removed from the ACL:

```
sh-4.1$ nfs4_setfacl -e /mixed
sh-4.1$ nfs4_getfacl /mixed
A::ldapuser@domain.netapp.com:ratTnNcCy
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A:g:GROUP@:rxtncy
D:g:GROUP@:waDTC
A::EVERYONE@:rxtncy
D::EVERYONE@:waDTC

filer> fsecurity show /vol/mixed
[/vol/mixed - Directory (inum 64)]
  Security style: Mixed
  Effective style: Unix  <-- effective style changes again

  DOS attributes: 0x0030 (---AD---)

  Unix security:
    uid: 0 (root)
    gid: 1 (daemon)  <-- GID stays as 1 (daemon)
    mode: 0755 (rwxr-xr-x)

  NFSv4 security descriptor:
    DACL:
      Allow - uid: 55(ldapuser) - 0x0016019d  <-- ldapuser is added to the DACL
      Allow - OWNER@ - 0x001601ff
      Deny  - OWNER@ - 0x00000000
      Allow - GROUP@ - 0x001200a9 (Read and Execute)
      Deny  - GROUP@ - 0x00040146
      Allow - EVERYONE@ - 0x001200a9 (Read and Execute)
      Deny  - EVERYONE@ - 0x00040146
```

Explicit Deny

NFSv4 permissions may include explicit DENY attributes for OWNER, GROUP, and EVERYONE. That is because NFSv4 ACLs are “default-deny,” which means that if an ACL is not explicitly granted by an ACE, then it is denied.

```
sh-4.1$ nfs4_getfacl /mixed
A::ldapuser@domain.netapp.com:ratTnNcCy
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A:g:GROUP@:rxtncy
D:g:GROUP@:waDTC
A::EVERYONE@:rxtncy
D::EVERYONE@:waDTC
```

DENY ACEs should be avoided whenever possible, as they can be confusing and complicated. When DENY ACEs are set, users might get denied access when they are expected to be granted access. This is because the ordering of NFSv4 ACLs affects how they are evaluated.

The preceding set of ACEs are the equivalent to 755 in mode bits. That means:

- Owner has full rights.
- Groups have read only.
- Others have read only.

However, even if permissions are adjusted to 775 equivalent, access can be denied due to the explicit DENY set on EVERYONE.

For example, the user “ldapuser” belongs to the group “Domain Users.”

```
sh-4.1$ id
uid=55(ldapuser) gid=513(Domain Users) groups=513(Domain Users),503(unixadmins)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Permissions on the volume “mixed” are 775. The owner is root, and the group is “Domain Users”:

```
[root@centos6 /]# nfs4_getfacl /mixed
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A:g:GROUP@:rwaDxtTnNcy
D:g:GROUP@:C
A::EVERYONE@:rxtncy
D::EVERYONE@:waDTC

[root@centos6 /]# ls -la | grep mixed
drwxrwxr-x.   3 root     Domain Users   4096 Apr 30 09:52 mixed
```

Because “ldapuser” is a member of Domain Users, it should have write access to the volume, and it does:

```
[root@centos6 /]# su ldapuser
sh-4.1$ cd /mixed
sh-4.1$ ls -la
total 12
drwxrwxr-x.  3 root Domain Users 4096 Apr 30 09:52 .
dr-xr-xr-x. 28 root root         4096 Apr 29 15:24 ..
drwxrwxrwx.  6 root root         4096 Apr 30 08:00 .snapshot
sh-4.1$ touch newfile
sh-4.1$ nfs4_getfacl /mixed

sh-4.1$ ls -la
total 12
drwxrwxr-x.  3 root     Domain Users 4096 Apr 30 09:56 .
dr-xr-xr-x. 28 root     root         4096 Apr 29 15:24 ..
drwxrwxrwx.  6 root     root         4096 Apr 30 08:00 .snapshot
-rw-r--r--.  1 ldapuser Domain Users    0 Apr 30 09:56 newfile
```

However, if the ACLs are reordered and the explicit DENY for EVERYONE is placed ahead of group, then “ldapuser” is denied access to write to the same volume to which it just had access to write:

```
[root@centos6 /]# nfs4_getfacl /mixed
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A::EVERYONE@:rxtncy
D::EVERYONE@:waDTC
A:g:GROUP@:rwaDxtTnNcy

[root@centos6 /]# su ldapuser
sh-4.1$ cd /mixed
sh-4.1$ ls -la
total 12
drwxrwxr-x.  3 root     Domain Users 4096 Apr 30 09:56 .
dr-xr-xr-x. 28 root     root         4096 Apr 29 15:24 ..
drwxrwxrwx.  6 root     root         4096 Apr 30 08:00 .snapshot
-rw-r--r--.  1 ldapuser Domain Users    0 Apr 30 09:56 newfile

sh-4.1$ touch newfile2
touch: cannot touch `newfile2': Permission denied
```

If the explicit DENY rule is removed, the desired access is restored:

```
[root@centos6 /]# nfs4_getfacl /mixed
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A::EVERYONE@:rxtncy
A:g:GROUP@:rwaDxtTnNcy

[root@centos6 /]# su ldapuser

sh-4.1$ cd /mixed

sh-4.1$ ls -la
total 12
drwxrwxr-x.  3 root     Domain Users 4096 Apr 30 09:56 .
```

```
dr-xr-xr-x. 28 root     root         4096 Apr 29 15:24 ..
drwxrwxrwx.  6 root     root         4096 Apr 30 08:00 .snapshot
-rw-r--r--.  1 ldapuser Domain Users    0 Apr 30 09:56 newfile

sh-4.1$ touch newfile2

sh-4.1$ ls -la
total 12
drwxrwxr-x.  3 root     Domain Users 4096 Apr 30 10:06 .
dr-xr-xr-x. 28 root     root         4096 Apr 29 15:24 ..
drwxrwxrwx.  6 root     root         4096 Apr 30 08:00 .snapshot
-rw-r--r--.  1 ldapuser Domain Users    0 Apr 30 09:56 newfile
-rw-r--r--.  1 ldapuser Domain Users    0 Apr 30 10:06 newfile2
```

It is best practice to only set DENY ACEs when absolutely necessary.

6 NFSv4 Delegations

Delegations in NFSv4 provide cache correctness for the byte range that is cached in the client's local memory. CB_NULL from the server determines whether there is a callback path available for issuing a delegation to the requesting client. In a read-intensive environment, read delegations would reduce the number of additional RPC calls that are generated from GETATTRs. Because a NetApp controller is shared storage, it is important to update "mtime" for any file that is accessed from the clients. NFSv4 supported clients such as Solaris 10 have an additional lightweight parameter called "nverify," which silently checks the NetApp storage for any "mtime" change, thus reducing any additional GETATTR RPC calls over the network. This helps to minimize the number of GETATTRs generated by the application that is mounting the file system over NFSv4. A WRITE delegation can be helpful in software build environments where the user checks out the code in their user scratch space and nobody is allowed to modify that section of the code until the user checks it back into the main codeline. Directory delegations are not part of NFSv4. They are currently proposed for NFSv4.1.

The NetApp controller can recall a delegation as long as a callback path is available from the client. If there are conflicts in requests from clients or a resource constraint on the server, Data ONTAP can recall the delegation. In Data ONTAP 7.3 and later, the callbacks originate from ports higher than 1024; those are the nonprivileged ports. This release also started supporting Kerberos callbacks as the same security type as incoming requests. The security type used in callbacks will depend on client support for callbacks. If secure callbacks are supported by the client, then the storage system running Data ONTAP operating in 7-Mode with the following hidden option:

```
filer> options nfs.v4.secure.callback [on|off]
```

The option is set to "off" by default and currently is not available in systems running clustered Data ONTAP.

Delegation of file operations to a client can be recalled when the lease expires or when the storage system receives any of the following requests from another client:

- Write to file, open file for writing, or open file for "deny read"
- Change file attributes
- Rename file
- Delete file

When a lease expires, the delegation state is revoked, and all of the associated states are marked "soft." This means that if the storage system receives a conflicting lock request for this same file from another client before the lease has been renewed by the original client previously holding the delegation, the conflicting lock is granted. If there is no conflicting lock and the original client holding the delegation renews the lease, the soft locks are changed to hard locks and are not removed in the case of a

conflicting access. However, the delegation is not granted to the client if the client does not have a callback path to the server (CB_NULL).

The default time for a client to get new delegations from a 7-Mode NetApp storage system running Data ONTAP 7.3 or later is 45 seconds. This value can be controlled using the following option:

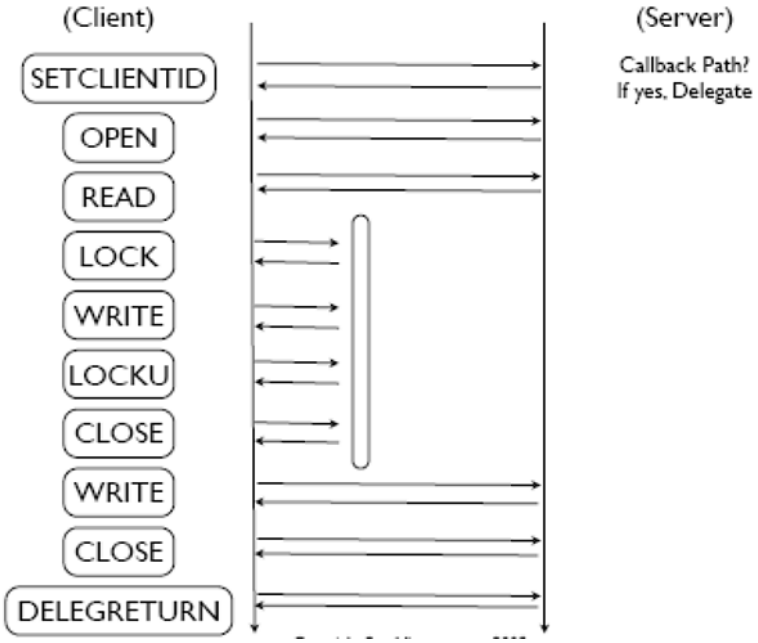
```
filer> options nfs.v4.state_recycle_timeout [time in seconds]
```

It is not recommended to adjust this value without guidance from NetApp Support. The option is set due to considerations for data resiliency during storage failover and giveback or in the case of storage system head reboots.

When the server (NetApp storage system) reboots, the delegation state is lost. Clients can reclaim the delegation state upon reconnection instead of going through the entire delegation request process again. When a client holding a read delegation reboots, all delegation state information is flushed from the storage system cache upon reconnection. The client must issue a delegation request to establish a new delegation.

Note: In RHEL5, the delegations seem to be broken. NetApp recommends that you run on kernel 2.6.19-rc3 and later for the delegations to work properly. RHEL6.x based on 2.6.32 kernel would be a more appropriate choice for using delegations.

Figure 3) Delegation flow.



6.1 Getting a Read Delegation

The NFSv4 server in Data ONTAP grants a read delegation in response to an OPEN for read, if no other client has the file open for read or is denying read. This guarantees that no other client will be able to write to the file. If other clients open the same file for read-only access, they will be allowed to read the file. If another NFSv4 client supporting read delegations opens the file for read-only access, it is granted a read delegation as well. For files being created, no delegation is returned on exclusive creates. This is because the client issues a SETATTR request after the CREATE, which causes the delegation to be recalled anyway, so as an optimization the delegation is not given out in the first place.

When using a read delegation, a client can do read-only opens and corresponding closes locally. It also doesn't need to poll the server to check modified times on the file, because no one will be allowed to

modify the file. A lease is associated with a delegation. Upon lease expiration, the delegation's state goes away, and any locks associated with the delegation are marked expired. If the client does not renew its lease within a certain time period (controlled by an option), these locks are revoked.

6.2 Recalling a Read Delegation

A client can voluntarily return the delegation. Additionally, the NetApp storage system can recall the delegation in case of conflicting access to the file. This is done through a callback path established from the server to the client.

A read delegation is recalled when any of the following events occur:

- OPEN for WRITE
- OPEN denying READ
- REMOVE, RENAME, SETATTR
- WRITE

In Data ONTAP 7.3 or later, a READ delegation can be recalled by the NetApp system when it is running low on resources, when a per-client limit on the locks is reached, or when the client has been idle for some time. When a delegation is recalled, there might be a number of opens that the client has done locally and now needs to propagate to the server. It will do that before returning the delegation. In Data ONTAP 7.3 and later, when a client fails to return a delegation upon recall, the NetApp system revokes the delegation and will not hand out any delegations to that client for 300 seconds by default. This helps protect against misbehaving clients. The timeout value is controlled by the following option in Data ONTAP running 7-Mode:

```
filer> options nfs.v4.bad_client_reset_time [time in seconds]
```

It is not recommended to change this value without NetApp Support guidance. The options is currently not available for systems running clustered Data ONTAP.

When a conflicting request such as an OPEN for WRITE comes in for a file that has a read delegation, an NFSERR_DELAY/NFSERR_JUKEBOX error is returned if the request is coming from an NFSv4 client. If the request is coming over NFSv2/NFSv3/CIFS, the request is suspended waiting for the delegation to be recalled. When that is done, the suspended request is restarted and finally granted.

6.3 Getting a Write Delegation

The server grants a write delegation if no other client has the file open, giving exclusive access to a file to one client. After a client has a write delegation, it is guaranteed that no other client can access that file as long as the delegation remains valid.

When the client has a write delegation, it can be lazy about flushing the writes on file CLOSE, but only if the server guarantees space for those writes. Clients choose not to flush the writes on CLOSE when the server space reservation information is not available. Currently the NFSv4 server in Data ONTAP does not make any such guarantees, so the client is required to flush everything on CLOSE.

In keeping with the way delegations work, clients should do all OPEN, WRITE, LOCK, and UNLOCK requests locally. However, there are some clients (such as Solaris) that send LOCK and UNLOCK requests to the server even though they might be holding a write delegation for that file. The NetApp storage system associates such opens and locks with the delegation, and if the delegation is returned, these opens and locks are disassociated from the delegation state.

However, if a client decides to do locking or unlocking locally, it must send the lock state over when the delegation is being returned or recalled. That case is handled the same way as the one mentioned previously.

When low on resources, the NetApp storage does not recall a WRITE delegation, as it does with the READ delegation. The NetApp system reclaims the delegations as per RFC 3530. In Data ONTAP 7.3.5.1, two new hidden options (`nfs.v4.lock_laststate_cleanup` and `nfs.v4.open_laststate_cleanup`) were made available to clean up the lock and open states, respectively, that are no longer used by any user, file, or directory or when the controller is running low on memory resources. The recycle timeout for the state object is set at 45 seconds. UNIX clients such as Solaris 10 update 7 and later, RHEL5.6/6.2 and later, and AIX5.3/6.1 support delegations by default. No option is required to enable this feature on the clients as long as delegation is enabled on NetApp storage. These options do not currently exist in clustered Data ONTAP.

Performance Impact

Delegations provide aggressive caching that can help in the following environments:

- Frequent opens and closes
- File locking
- Read-only sharing
- High-latency environments
- Fast clients with a lot of memory
- Applications that are GETATTR-heavy in read only environments
- Multiple writers to a single file

6.4 Turning on Read/Write Delegations on the NetApp Storage System

The following two options enable the READ/WRITE delegations on a NetApp system running 7-Mode:

```
options nfs.v4.read_delegation       on
options nfs.v4.write_delegation      on
```

When these options are turned ON, the NetApp system delegates READ or ACCESS to the client upon OPEN.

In clustered Data ONTAP, READ/WRITE delegations are enabled/disabled by doing the following:

```
cluster::> nfs server modify –vserver vs0 -v4.0-write-delegation [enabled|disabled]
cluster::> nfs server modify –vserver vs0 -v4.0-read-delegation [enabled|disabled]
```

The following is a simple trace for a READ delegation where the client 172.28.2.82 is opening the file `b`. In the next line, the NetApp system issues a READ (DT=R) delegation to the client.

```
58      0.00030  172.28.2.82 -> 172.17.44.43 NFS C 4 (open) b PUTFH FH=5D51 OPEN b OT=NC SQ=1
CT=N AC=R DN=N OO=0045 GETFH GETATTR 10001a 30a23a
59      0.00337 172.17.44.43 -> 172.28.2.82  NFS R 4 (open) NFS4_OK PUTFH NFS4_OK OPEN NFS4_OK
ST=0AA7:0 DT=R DST=0AA1:0 GETFH NFS4_OK FH=5E2D GETATTR NFS4_OK
```

The following is a sample trace where the NetApp system is providing a WRITE delegation (DT=W) to the client after the client attempts to open file `b`.

```
32      0.00036  172.28.2.82 -> 172.17.44.43 NFS C 4 (open) PUTFH FH=5D51 OPEN b OT=CR(U) SQ=9
CT=N AC=W DN=N OO=0017 GETFH GETATTR 10001a 30a23a
33      0.00725 172.17.44.43 -> 172.28.2.82  NFS R 4 (open) NFS4_OK PUTFH NFS4_OK OPEN NFS4_OK
ST=0D03:0 DT=W DST=0D05:0 LB=SZ(0) GETFH NFS4_OK FH=5E2D GETATTR NFS4_OK
```

7 Kerberos

Data ONTAP currently supports Kerberos authentication on the DES-CBC-CRC and DES-CBC-MD5 cipher suite. For information on how to set up Kerberized NFS, see TR-4073 for clustered Data ONTAP and TR-3457 for 7-Mode.

8 Mount Option Best Practices with NFSv4

When specifying a mount, there are a variety of mount options that can be leveraged to help resiliency and performance. The following is a list of some of those options, as well as information to assist with setting these options

8.1 Mount Options

The following is a list of typical mount options and suggestions on how to apply them with NetApp storage using NFSv4. In most cases, mount options are standardized, Mount options might vary depending on the version and variety of Linux being used. Always consult the man pages of the Linux kernel being used to make sure the mount options exist for that version.

Mount options are specified using the -o flag. Mount options such as noacl and nolock do not apply to NFSv4 and are not recommended.

If NFSv4 is enabled on the NetApp storage system, then newer clients will negotiate NFSv4 on their own without mount options. Older clients will use NFSv3 unless specified. If NFSv4 is disabled on the NetApp storage system, clients fall back to using NFSv3.

hard or soft

`hard or soft` specifies whether the program using a file using NFS should stop and wait (`hard`) for the server to come back online, if the NFS server is unavailable, or if it should report an error (`soft`).

If `hard` is specified, processes directed to an NFS mount that is unavailable cannot be terminated unless the intr option is also specified.

If `soft` is specified, the `timeo=<value>` option can be specified, where `<value>` is the number of seconds before an error is reported.

For business-critical NFS exports, NetApp recommends using `hard` mounts. NetApp strongly discourages the use of `soft` mounts.

intr

`intr` allows NFS processes to be interrupted when a mount is specified as a hard mount. This is deprecated in new clients such as RHEL 6.4 and is hard coded to “nointr.” Kill -9 will be the only way to interrupt a process in newer kernels.

For business-critical NFS exports, NetApp recommends using `intr` with `hard` mounts in clients that support it.

nfsvers

`nfsvers` does not apply to NFSv4 mounts. To specify NFSv4, use the –t option for “type.”

noexec

`noexec` prevents the execution of binaries on an NFS mount.

NetApp recommends use of this option only when advised by the application or client vendor.

nosuid

`nosuid` prevents the setting of set-user-identifier or set-group-identifier bits. This prevents remote users from gaining higher privileges by running a setuid program.

NetApp recommends use of this option for better security on NFS mounts.

port

`port` allows the specification of which port an NFS mount will leverage. By default, NFS uses port 2049 for communication with NetApp storage. If a different port is specified, firewall considerations should be considered, as communication can be blocked if an invalid port is specified.

NetApp does not recommend changing this value unless necessary.

In the case of automounter, the following change is recommended in the auto.home or auto.misc or auto.* files:

-fstype=nfs4, rw, proto=tcp,port=2049

rsize=num and wsize=num

`rsize` and `wsize` are used to speed up NFS communication for reads (`rsize`) and writes (`wsize`) by setting a larger data block size, in bytes, to be transferred at one time. Be careful when changing these values; some older Linux kernels and network cards do not work well with larger block sizes.

NetApp recommends use of this option only when advised by the application or client vendor. NetApp highly recommends using 64k rsize and wsize for better performance.

sec

`sec` specifies the type of security to utilize when authenticating an NFS connection.

`sec=sys` is the default setting, which uses local UNIX UIDs and GIDs by means of AUTH_SYS to authenticate NFS operations.

`sec=krb5` uses Kerberos V5 instead of local UNIX UIDs and GIDs to authenticate users.

`sec=krb5i` uses Kerberos V5 for user authentication and performs integrity checking of NFS operations using secure checksums to prevent data tampering.

`sec=krb5p` uses Kerberos V5 for user authentication and integrity checking and encrypts NFS traffic to prevent traffic sniffing. This is the most secure setting, but it also has the most performance overhead involved.

Data ONTAP 7-Mode supports all security varieties specified.

Clustered Data ONTAP supports only sys and krb5 currently.

NetApp recommends using `sec` only when clients have been configured to use the specified security mode.

tcp or udp

`tcp` or `udp` is used to specify whether the mount will use TCP or UDP for transport.

NFSv4 only supports TCP, so this option does not apply to NFSv4.

NetApp recommends TCP for all mounts, regardless of version, provided the client supports mounting using TCP.

Appendix

NFSv4 Support in Data ONTAP

The following table describes the features that have been added to Data ONTAP NFSv4 support since its release.

Table 2) NFSv4 features for Data ONTAP operating in 7-Mode.

Release	Features
6.4.x	Basic NFSv4 support. All mandatory features except for LIPKEY and SPKM-3 and access and audit ACE.
6.5/6.5.1 and 6.5.2	Basic NFSv4 support. All mandatory features except for LIPKEY and SPKM-3 and access and audit ACE. Read delegation support added.
6.5.3/7.0	Basic NFSv4 support, including ACLs. All mandatory features except for LIPKEY and SPKM- and access and audit ACE. Shipped with write delegations.
7.0.1/7.1	ACL inheritance was introduced in Data ONTAP 7.0.1 and controlled through the nfs.v4.acl.enable option, but this was changed in Data ONTAP 7.1 and later releases, where ACL is always inherited if parent has inheritable ACL. Data ONTAP 7.1 also has more robust pseudo file system implementation.
7.3	Supports nested exports rules, UNICODE (UTF8), and Kerberized NFSv4 callbacks.
7.3.1	Starting from Data ONTAP 7.3.1, NFSv4 is supported over IPv6. A new option, `nfs.max_num_aux_groups`, was added that specifies the maximum number of auxiliary UNIX groups to which a user can belong. The default value is 32.The option `nfs.thin_prov.ejuke` was added in this release to specify whether the NFS server forces a client to retry a request by breaking the connection or by sending NFSERR_JUKEBOX (NFSv3) or NFS4ERR_DELAY (NFSv4). Does not affect NFSv2.
7.3.3	Beginning in Data ONTAP 7.3.3, the `exportfs` command was updated with options for managing entries in the NFS access cache (`-c` option to take multiple IP addresses as arguments and `-f` option includes a `-n` parameter, allowing the user to flush access cache entries that correspond to a specified host). Beginning with Data ONTAP 7.3.3, a new option `nfs nsdb flush` was introduced that flushes specified entries from the name server database cache (NSDB).
7.3.4	Option `nfs.v4.snapshot.active.fsid.enable`, which determines whether the FSID of objects in a Snapshot copy matches the FSID of the active file system for NFSv4, was added in this release.
7.3.5	Beginning with Data ONTAP 7.3.5, a new hidden option `nfs.v4.acl_preserve` is introduced to preserve the ACL when the mode bits are set.

Release	Features
7.3.5.1	Beginning with Data ONTAP 7.3.5.1, two new hidden options are "`nfs.v4.lock_laststate_cleanup`" and "`nfs.v4.open_laststate_cleanup`," which clean up the lock and open states respectively that are no longer used by any user, file, or directory when the controller is running low on memory resources.

Table 3) NFSv4 features for clustered Data ONTAP.

Release	Features
8.1	Basic NFSv4 support. All mandatory features except for LIPKEY and SPKM-3 and access and audit ACE.
8.1.1	Basic NFSv4.1 support.
8.2	NFSv4.1 READ/WRITE delegation support. NFSv4.x session slot replay cache size support. NFSv4.x session slot number configuration. NFSv4 max ACEs (1024 max).

Acronyms

Acronym	Definition
ACE	Access control entry
ACL	Access control list
CIFS	Common Internet File System
DACL	Discretionary access control list
DNS	Domain Name System
FTP	File Transfer Protocol
GID	Group ID
KDC	Key distribution center
LDAP	Lightweight Directory Access Protocol
NFS	Network File System
NIS	Network Information Service
NLM	Network Lock Manager
NSM	Network Status Monitor
RFC	Request for comments
RHEL	Red Hat Enterprise Linux
RPC	Remote procedure call
SACL	Security access control list
TCP	Transmission Control Protocol
UCS	Universal Character Set
UDP	User Datagram Protocol

Acronym	Definition
UID	User ID
UTF	Universal Character Set Transformation Format

References

The following references were used in this technical report:

- RFC 2847: LIPKEY - A Low Infrastructure Public Key Mechanism Using SPKM
  www.ietf.org/rfc/rfc2847.txt
- RFC 3530: Network File System (NFS) Version 4 Protocol
  www.ietf.org/rfc/rfc3530.txt
- RFC 1094: NFS: Network File System Protocol Specification
  www.ietf.org/rfc/rfc1094.txt
- RFC 4120: The Kerberos Network Authentication Service (V5)
  www.ietf.org/rfc/rfc4120.txt

Version History

Version	Date	Document Version History
Version 1.0	2007	Initial release
Version 2.0	May 2012	Major updates to all sections
Version 3.0	April 2013	Major updates to all sections

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

NetApp provides no representations or warranties regarding the accuracy, reliability, or serviceability of any information or recommendations provided in this publication, or with respect to any results that may be obtained by the use of the information or observance of any recommendations provided herein. The information in this document is distributed AS IS, and the use of this information or the implementation of any recommendations or techniques herein is a customer's responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. This document and the information contained herein may be used solely in connection with the NetApp products discussed in this document.

Go further, faster®

www.netapp.com