Technical Report

NFS in NetApp ONTAP

Best practice and implementation guide

Justin Parisi & Elliott Ecton, NetApp
June 2023 | TR-4067

Abstract

This document provides basic concepts, support information, configuration tips, and best practices for NFS in NetApp® ONTAP®. This guide covers the latest available ONTAP versions for currency and length. In most cases, the commands and best practices listed in this guide apply to all ONTAP versions.

Results might vary in older ONTAP versions. Consult the product documentation for your ONTAP version when necessary.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Basic NFS concepts in NetApp ONTAP

Intended audience and assumptions

This technical report is for storage administrators, system administrators, and data center managers. It assumes basic familiarity with the following:

- The NetApp ONTAP data management software
- Network file sharing protocols (NFS in particular)

This document contains some advanced and diagnostic-level commands. Exercise caution when using these commands. If there are questions or concerns about using these commands, contact NetApp Support for assistance.

Stated support for NFS

The following section covers the stated support for NFS versions and client support with ONTAP.

NFS version support

Table 1 shows the NFS version support in ONTAP and the supported ONTAP release for the protocol.

Table 1) NFS version support in ONTAP.

NFS version	Supported ONTAP version
NFSv2	ONTAP 8.2 and earlier
NFSv3	Supported in all ONTAP releases
NFSv4.0	ONTAP 8.1 and later
NFSv4.1 (with pNFS)	ONTAP 8.1 and later
NFSv4.2	ONTAP 9.8 (basic protocol support) ONTAP 9.9.1 (labeled NFS)

NFS client support

NFSv3 clients will not appear in the Interoperability Matrix (IMT), as ONTAP supports all NFS clients that comply with the standards defined in Request For Comments (RFC) documentation that is ratified by the Internet Engineering Task Force (IETF). Client support for NFSv4.x is also dependent on the stated client support for each vendor. We do show NFSv4.1 client support in the IMT.

The NetApp IMT can be found here.

These are the most recent RFCs for each supported NFS version in ONTAP:

- RFC-1813: NFSv3
- RFC-7530: NFSv4.0
- RFC-5661: NFSv4.1
- RFC-7862: NFS4.2

Windows NFS support

Starting in ONTAP 8.2.3 and 8.3.1, support for NFS using Windows clients was added. Windows NFS does not comply with RFC standards, so some extra considerations need to be made to use it. For details, see the section called “NFS on Windows.”

NFS security support

Table 2 illustrates the supported security methods for securing NFS, both at-rest and in-transit. For more information about specific security features in ONTAP, see TR-4569: Security Hardening Guide for NetApp ONTAP.

- **At-rest security** refers to security for data that is in-place.
- **In-flight security** refers to security for data that is being transmitted across the network.

Table 2) NFS security support details.

Supported at-rest security for NFS	Supported in-flight security for NFS
• NetApp Volume Encryption (NVE) • NetApp Aggregate Encryption (NAE) • Self-encrypting Drives (SED) • NetApp Storage Encryption Drives (NSE) • Export Policies and Rules • Access control lists (ACLs) • Identity management (users, groups, file ownership)	• Kerberos (krb5, krb5i, krb5p) – Supported encryption types include: AES-256, AES-128, 3DES, DES

Note: NFS over SSH and NFS over stunnel are not supported with ONTAP.

NFS features

Each NFS version adds new features to the protocol to enhance operations, performance and business use cases. The following table shows the supported NFS features in ONTAP, along with the associated NFS version information for that feature.

Table 3) NFS feature support.

NFS version	Features available
All versions	• Volume and qtree level export rules • 1,024 maximum auxiliary GIDs for NFS operations • 64-bit file IDs • Ability to junction volumes across a namespace to create pseudo-filesystems • UNIX-style mode bit permissions (rwx) • TCP and UDP (NFSv4.x is TCP only) • Changing ports • Limiting port range between 1-1024 (mount-rootonly, nfs-rootonly) • FSID changes across file systems • Showmount (see the “Showmount” section for information and limitations) • NFS client to volume mapping • Multiprotocol NAS access (CIFS/SMB and NFS) • Lightweight Directory Access Protocol (LDAP)/ Network Information Service (NIS) for UNIX Identity mapping • Netgroups • Windows NFS
NFSv3	• All RFC standard features for NFSv3 are supported
NFSv4.0/4.1	• ACLs (up to 1,024) • Delegations • Migrations

NFS version	Features available
	• Referrals • Lease timeout configuration • pNFS (v4.1)
NFSv4.2	• Basic protocol support in ONTAP 9.8 • Labeled NFS (ONTAP 9.9.1; Guest mode and Limited Server Mode only)

The following table shows features that are currently unsupported for NFS.

Table 4) Unsupported NFS features.

NFS version	Unsupported features
All versions	• POSIX ACLs • Subdirectory exports • SSSD Dynamic UIDs
NFSv3	• Extended attributes (not in RFC spec)
NFSv4.0/4.1	• Session trunking/multipath
NFSv4.2	• Live file migration (Flex Files) • Sparse file • Space reservation • IO_ADVISE • Application data holes • Server-side copy • Full mode for Labeled NFS

NFS server options

NFS servers in ONTAP are configurable via server options. By default, the options are configured with best practices in mind and should not require specific modification. However, in some cases (such as enabling NFSv4.x support), the NFS options will need to be changed.

These default values, as well as all of the available NFS options and descriptions, are covered in the man pages for your version of ONTAP or by running `man nfs modify` from the CLI.

The rootonly options – nfsrootonly and mountrootonly

The `rootonly` options are added to avoid untrusted client access. Untrusted clients (those not part of the export rules) can potentially access data by using SSH tunneling to trusted clients. However, those requests would come from untrusted ports (ports greater than 1,024). This can provide a back door for clients not intended to have access.

Therefore, the enabling or disabling of the `rootonly` options hinges upon need. Does the environment require more ports to allow NFS to function properly? Or is it more important to prevent untrusted clients from accessing mounts?

One potential compromise is to make use of NFSv4.x and/or Kerberos authentication for a higher level of secured access to NFS exports. TR-4616: NFS Kerberos in ONTAP covers how to use NFS Kerberos.

In these scenarios, using the `mount-rootonly` and/or `nfs-rootonly` options can alleviate these issues.

To check port usage on the client, run the following command:

```
# netstat -na | grep [IP address]
```

To check port usage on the cluster, run the following command:

```
cluster::> network connections active show -node [nodename] -vserver [vservername] -service nfs*
```

For more details about these options and an example of when these options can help environments with a large number of NFS clients, see “Network port exhaustion with a large number of NFS clients.”

Showmount

ONTAP earlier than 8.3 does not support the `showmount` command from NFS clients to expose export paths. This limitation was by design, due to performance considerations. ONTAP clusters can potentially have thousands of export rules, so a query for all exports can be process intensive. Additionally, exports are not in flat files and are applied to volumes as rules, so the export path and export rules would live in two different places.

Example of showmount –e from client with showmount disabled in ONTAP:

```
[root@nfsclient /]# showmount -e x.x.x.a
Export list for x.x.x.a:
/ (everyone)
```

The SVM has a vsroot volume mounted to `/`, which is the volume returned in the showmount query. All other volumes are mounted below that mount point and are not returned to the client when the showmount NFS option is disabled.

What happens during a Showmount query?

Showmount leverages the MOUNT protocol in NFSv3 to issue an EXPORT query to the NFS server. If the mount port is not listening or blocked by a firewall, or if NFSv3 is disabled on the NFS server, showmount queries fail:

```
# showmount -e x.x.x.a
mount clntudp_create: RPC: Program not registered
```

The following shows output from a packet trace of the `showmount` command being run against a data LIF in ONTAP with the option disabled:

```
x.x.x.x x.x.x.a MOUNT   170    V3 EXPORT Call (Reply In 17)
Mount Service
Program Version: 3
V3 Procedure: EXPORT (5)

x.x.x.a x.x.x.x MOUNT   202    V3 EXPORT Reply (Call In 16)
Mount Service
Export List Entry: /unix ->
```

Note: The trace shows that the server returns `/unix ->`. However, this export path has a specific client in the rule set:

```
cluster::> vol show -vserver NFS83 -junction-path /unix -fields policy
  (volume show)
vserver volume policy
------- ------ --------
NFS83   unix   restrict

cluster::> export-policy rule show -vserver NFS83 -policyname restrict
             Policy          Rule    Access   Client                RO
Vserver      Name            Index   Protocol Match                 Rule
------------ --------------- ------  -------- --------------------- ---------
NFS83        restrict        1       any      x.x.x.y        any
```

If client match is required in the showmount functionality, the showmount utility in the toolchest provides that functionality.

Showmount in ONTAP 8.3 and later

The showmount functionality is required for some applications to work properly, such as Oracle OVM, so showmount support was added in ONTAP 8.3 and later to properly support those applications.

This functionality is disabled by default. It can be enabled with the following command:

```
cluster::> nfs server modify -vserver NFS -showmount
    enabled  disabled
```

Note: To use showmount in ONTAP, the parent volume (including vsroot, or /) needs to allow read or traverse access to the client/user attempting to run showmount and vsroot (/) should use UNIX security style.

After this functionality is enabled, clients can query data LIFs for export paths. However, the clientmatch (access from clients, netgroups, and so on) information is not available. Instead, each path reflects everyone as having access, even if clients are specified in export policy rule sets.

Sample output of showmount in clustered ONTAP 8.3 and later:

```
# showmount -e x.x.x.a
Export list for x.x.x.a:
/unix       (everyone)
/unix/unix1 (everyone)
/unix/unix2 (everyone)
/           (everyone)
```

Note: If using Windows NFS, showmount should be enabled to prevent issues with renaming files and folders.

Showmount caching

When showmount is run from a client, it requests information from the NFS server on the cluster. Because export lists can be large, the cluster maintains a cache of this information to reduce the number of requests made to the NFS server.

When a volume is unmounted from the cluster namespace (see “The cluster namespace”) using the `volume unmount` command or from ONTAP System Manager, the cache does not update, so the exported path remains in cache until it expires or is flushed.

To flush the showmount cache:

```
cluster::> export-policy cache flush -vserver SVM -cache showmount
```

The cache only flushes on the node you are logged in to. For example, if you are logged in to node1’s management LIF, then the cache on node1 flushes. This means that only clients connecting to data LIFs local to node1 benefit from the cache flush. To flush the cache on other nodes, log into the node management LIF on those nodes. The node that is flushing is displayed when running the command.

```
cluster::> export-policy cache flush -vserver SVM -cache showmount

Warning: You are about to flush the "showmount" cache for Vserver "SVM" on node "node1", which
will result in increased traffic to the name servers. Do you want to proceed with flushing the
         cache? {y|n}: y
```

Namespace concepts

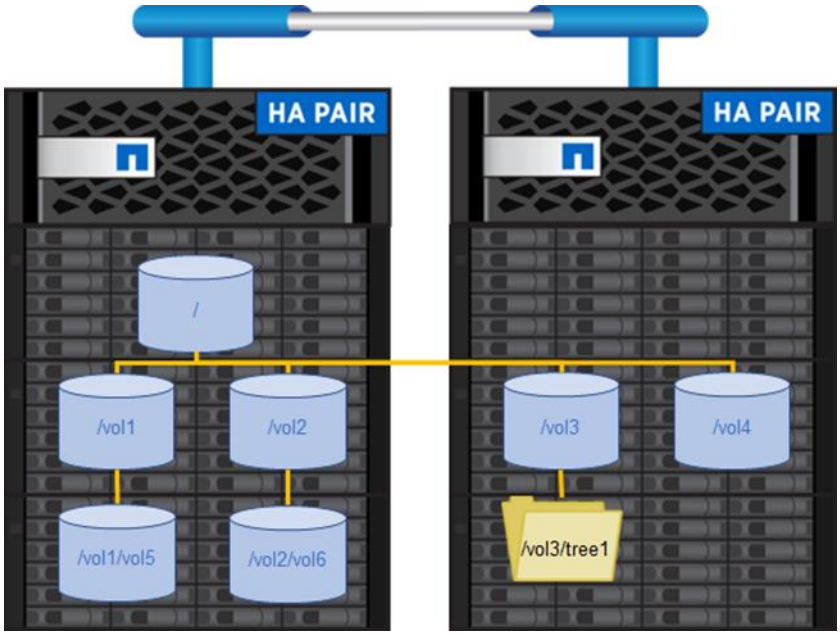
This section covers the concept of a “namespace” in NFS environments and how ONTAP offers multiple solutions for providing a global namespace for NFS clients.

The cluster namespace

A namespace in ONTAP is a collection of file systems hosted across different nodes in the cluster to provide scalable performance and capacity. Each SVM has a file namespace that consists of a single root volume. This namespace starts at the location of "/". Subsequent volumes and qtrees all traverse "/" and have their export paths defined by the volume option `-junction-path`. The SVM namespace can consist of one or more volumes linked by means of junctions that connect from a named junction inode in one volume to the root directory of another volume. A cluster can have more than one SVM, but each SVM only has one vsroot and one "/," which results in each SVM having a unique set of file system IDs. This prevents volumes in different SVMs from sharing file system IDs/file handles and avoids issues mounting NFS exports in multitenant environments.

All the volumes belonging to the SVM are linked into the global namespace in that cluster using the "/" export path. The cluster namespace is mounted at a single point in the cluster. The top directory of the cluster namespace within a cluster ("/") is a synthetic directory containing entries for the root directory of each SVM namespace in the cluster. The volumes in a namespace can be NetApp FlexVol® volumes or NetApp ONTAP FlexGroup volumes.

Figure 1) Cluster namespace.



Protecting your namespace

A vsroot volume lives only on a single node in a cluster, even though the SVM is accessible through multiple nodes. Because the vsroot is how NFS clients traverse the namespace, it is vital to NFS operations.

```
cluster::> vol offline -vserver NFS -volume vsroot

Warning: Offlining root volume vsroot of Vserver NFS will make all volumes on that Vserver
inaccessible.
Do you want to continue? {y|n}: y
Volume "NFS:vsroot" is now offline.
```

If the vsroot volume is somehow unavailable, then NFS clients will have issues whenever the vsroot volume is needed to traverse the file system.

This includes (but might not be limited to) the following behaviors:

- Mount requests hang.
- If / is mounted, traversal from / to another volume, running ls, and son on hang.
- Umount operations might fail because the mount is busy, even when the volume is back online.
- If a volume is already mounted below "/" (such as /vol1), then reads/writes/listings still succeed.

Load-sharing mirrors in ONTAP is a way to leverage the ONTAP SnapMirror capability to increase vsroot resiliency.

Note: Load-sharing mirrors are supported only with vsroot volumes. To share a load across data volumes, consider using NetApp FlexCache volumes instead.

When load-sharing mirrors are available for the vsroot volume, NFSv3 operations are able to leverage the load-sharing mirror destination volumes to traverse the file system. When load-sharing mirrors are in use, it is possible to access the source volume through the `.admin` folder within the NFS mount.

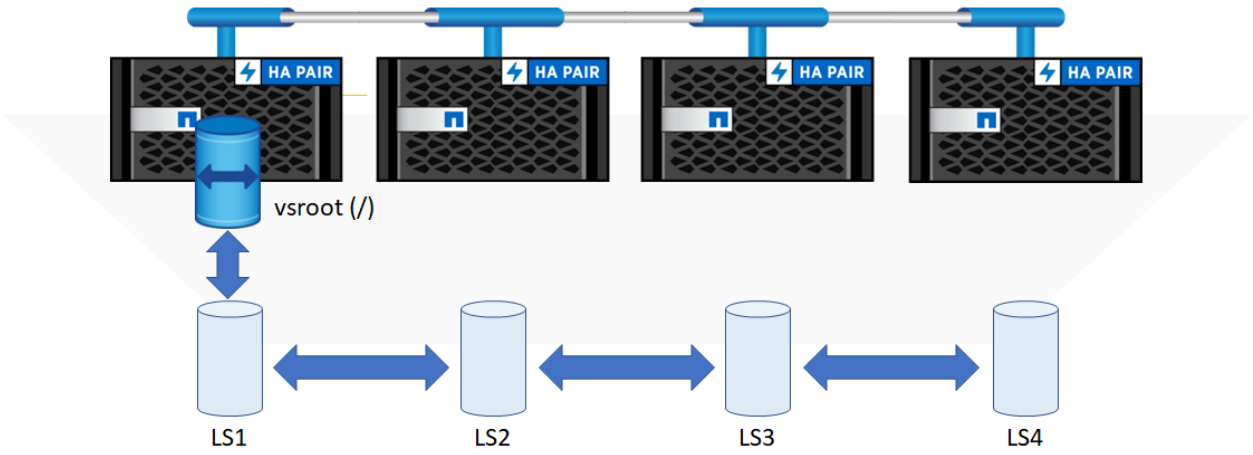
For more information, see Creating and initializing load-sharing mirror relationships.

NetApp highly recommends creating load-sharing mirror relationships for vsroot volumes in NFSv3 environments.

Note: NFSv4.x clients are unable to use load-sharing mirror volumes to traverse file systems due to the nature of the NFSv4.x protocol.

Figure 2 shows how load-sharing mirrors can provide access to "/" in the event the vsroot is unavailable.

Figure 2) Load-sharing mirror protection of vsroot volumes.



To create a load-sharing mirror for the vsroot volume, complete the following tasks:

- Typically, the vsroot volume is 1GB in size. Verify the vsroot volume size prior to creating new volumes and ensure the new volumes are all the same size.
- Create a destination volume to mirror the vsroot on each node in the cluster. For example, in a four-node cluster, create four new volumes with the -type DP.
- Create a new SnapMirror relationship from the vsroot source to each new DP volume you created. Specify a schedule for updates depending on the change rate of your namespace root. For example, hourly if you create new volumes regularly; daily if you do not.
- Initialize SnapMirror by using the `initialize-ls-set` command.

Pseudo file systems

The ONTAP architecture makes it possible to have a true pseudo-file system, which complies with the RFC 7530 NFSv4 standards.

```
Servers that limit NFS access to "shares" or "exported" file systems should provide a pseudo-file
system into which the exported file systems can be integrated, so that clients can browse the
server's namespace. The clients' view of a pseudo-file system will be limited to paths that lead
to exported file systems.
```

And in section 7.3:

```
NFSv4 servers avoid this namespace inconsistency by presenting all the exports within the
framework of a single-server namespace.  An NFSv4 client uses LOOKUP and READDIR operations to
browse seamlessly from one export to another.  Portions of the server namespace that are not
exported are bridged via a "pseudo-file system" that provides a view of exported directories
only.  A pseudo-file system has a unique fsid and behaves like a normal, read-only file system.
```

ONTAP has removed the `/vol` requirement for exported volumes seen in ONTAP operating in 7-Mode and instead uses a more standardized approach to the pseudo-file system. Because of this, you can now seamlessly integrate an existing NFS infrastructure with NetApp storage because "`/`" is truly "`/`" and not a redirector to `/vol/vol0`, as it was in 7-Mode.

A pseudo-file system applies only in 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 (such as `/volname`) does, then pseudo-file system concepts apply.

Having a pseudo-file system allows storage administrators to create their own file system namespaces, if they desire, by way of mounting volumes to other volumes using junction paths. This concept is illustrated in Figure 1) Cluster namespace..

Pseudo file system and -actual support

The use of -actual as an export option is supported only in NetApp ONTAP 7-Mode and not supported in ONTAP. The -actual option is used in cases where storage administrators wish to mask export paths and redirect to shorter names.

For example, if a storage administrator has a path of `/dept1/folder1`, then they may want to export that path as `/folder1`.

Working around lack of -actual support

In most cases, the `-actual` export option is not necessary in ONTAP. The design of the operating system allows natural pseudo file systems rather than those defined in export files. Everything is mounted beneath the SVM root volume, which is mounted to `/`. Exports can be set at the volume or qtree level, can be junctioned several levels deep, and can have names that do not reflect the actual volume names.

Table 5) Export examples.

Export path	Exported object
`/vol1`	Volume named vol1
`/NFSvol`	Volume named vol2
`/vol1/NFSvol`	Volume named vol2 junctioned to volume named vol1
`/vol1/qtree`	Qtree named qtree with parent volume named vol1
`/vol1/NFSvol/qtree1`	Qtree named qtree1 with parent volume named NFSvol junctioned to volume named vol1

One use case for `-actual` that is not inherently covered by the ONTAP NFS architecture is `-actual` for qtrees or folders. For instance, if a storage administrator wants to export a qtree or folder to a path such as `/folder1`, there is no way to do this natively using the NFS exports in the SVM. The path is instead `/volume/folder1`.

Sample export from 7-Mode:

```
/qtree –actual=/vol/vol1/qtree,rw,sec=sys
```

In ONTAP, the path for a qtree that NFS clients mount is the same path as the qtree is mounted to in the namespace. If this is not desirable, then the workaround is to leverage symlinks to mask the path to the qtree.

What is a symlink?

Symlink is an abbreviation for symbolic link. A symbolic link is a special type of file that contains a reference to another file or directory in the form of an absolute or relative path. Symbolic links operate transparently to clients and act as actual paths to data.

Relative paths versus absolute paths

Symlinks can leverage either relative or absolute paths. Absolute paths are paths that point to the same location on one file system regardless of the present working directory or combined paths. Relative paths are paths relative to the working directory.

For example, if a user is inside a directory at /directory and wants to go to /directory/user, that user can use a relative path:

```
# cd user/
# pwd
/directory/user
```

Or the user can use the absolute path:

```
# cd /directory/user
# pwd
/directory/user
```

When mounting a folder using NFS, it is better to use a relative path with symlinks, because there is no guarantee that every user mounts to the same mount point on every client. With relative paths, symlinks can be created that work regardless of what the absolute path is.

Using symlinks to simulate –actual support

In ONTAP, symbolic links can be used to simulate the same behavior that the export option `-actual` provided in 7-Mode.

For example, if a qtree exists in the cluster, the path can look like this:

```
cluster::> qtree show -vserver flexvol -volume unix2 -qtree nfstree

                      Vserver Name: flexvol
                       Volume Name: unix2
                        Qtree Name: nfstree
                        Qtree Path: /vol/unix2/nfstree
                    Security Style: unix
                       Oplock Mode: enable
                  Unix Permissions: ---rwxr-xr-x
                           Qtree Id: 1
                       Qtree Status: normal
                      Export Policy: volume
        Is Export Policy Inherited: true
```

The parent volume is `unix2 (/unix/unix2)`, which is mounted to volume `unix (/unix)`, which is mounted to `vsroot` (/).

```
cluster::> vol show -vserver flexvol -volume unix2 -fields junction-path
  (volume show)
vserver volume junction-path
------- ------ -------------
```

```
flexvol unix2  /unix/unix2
```

The exported path would be `/parent_volume_path/qtree`, rather than the `/vol/parent_volume_path/qtree` seen earlier. The following is the output from a `showmount -e` command from an NFS client:

```
/unix/unix2/nfstree                                                (everyone)
```

Some storage administrators might not want to expose the entire path of `/unix/unix2/nfstree`, because it can allow clients to attempt to navigate other portions of the path. To allow the masking of that path to an NFS client, a symlink volume or folder can be created and mounted to a junction path. For example:

```
cluster::> vol create -vserver flexvol -volume symlinks -aggregate aggr1 -size 20m -state online
-security-style unix -junction-path /NFS_links
```

The volume size can be small (minimum of 20MB), but that depends on the number of symlinks in the volume. Each symlink is 4k in size, so you may need to create larger volume sizes to accommodate the number of symlinks. Alternatively, create a folder under vsroot for the symlinks.

After the volume or folder is created, mount the vsroot to an NFS client to create the symlink.

```
# mount -o nfsvers=3 x.x.x.e:/ /symlink
# mount | grep symlink
x.x.x.e:/ on /symlink type nfs (rw,nfsvers=3,addr=x.x.x.e)
```

Note: If using a directory under vsroot, mount vsroot and create the directory.

```
# mount -o nfsvers=3 x.x.x.e:/ /symlink
# mount | grep symlink
x.x.x.e:/ on /symlink type nfs (rw,nfsvers=3,addr=x.x.x.e)
# mkdir /symlink/symlinks
# ls -la /symlink | grep symlinks
drwxr-xr-x.  2 root root 4096 Apr 30 10:45 symlinks
```

To create a symlink to the qtree, use the `-s` option (s = symbolic). The link path needs to include a relative path that directs the symlink to the correct location without needing to specify the exact path. If the link is inside a folder that does not navigate to the desired path, then ../ needs to be added to the path.

For example, if a folder named NFS_links is created under `/` and the volume unix is also mounted under `/`, then navigating to `/NFS_links` and creating a symlink cause the relative path to require a redirect to the parent folder.

Example of a symlink created in a symlink volume mounted to /NFS_links:

```
# mount -o nfsvers=3 x.x.x.e:/ /symlink/
# mount | grep symlink
x.x.x.e:/ on /symlink type nfs (rw,nfsvers=3,addr=x.x.x.e)
# cd /symlink/NFS_links
# pwd
/symlink/NFS_links
# ln -s ../unix/unix2/nfstree LINK
# ls -la /symlink/unix/unix2/nfstree/
total 8
drwxr-xr-x. 2 root root 4096 May 15 14:34 .
drwxr-xr-x. 3 root root 4096 Apr 29 16:47 ..
-rw-r--r--. 1 root root    0 May 15 14:34 you_are_here
# cd LINK
# ls -la
total 8
drwxr-xr-x. 2 root root 4096 May 15 14:34 .
drwxr-xr-x. 3 root root 4096 Apr 29 16:47 ..
-rw-r--r--. 1 root root    0 May 15 14:34 you_are_here
# pwd
```

```
/symlink/NFS_links/LINK
```

Note: Despite the fact that the symlink points to the actual path of `/unix/unix2/nfstree`, `pwd` returns the path of the symlink, which is `/symlink/NFS_links/LINK`. The file `you_are_here` has the same date and timestamp across both paths.

Because the path includes ../, this symlink cannot be directly mounted.

Example of symlink created in vsroot:

```
# mount -o nfsvers=3 x.x.x.e:/ /symlink/
# mount | grep symlink
x.x.x.e:/ on /symlink type nfs (rw,nfsvers=3,addr=x.x.x.e)
# cd /symlink/
# pwd
/symlink
# ln -s unix/unix2/nfstree LINK1
# ls -la /symlink/unix/unix2/nfstree/
total 8
drwxr-xr-x. 2 root root 4096 May 15 14:34 .
drwxr-xr-x. 3 root root 4096 Apr 29 16:47 ..
-rw-r--r--. 1 root root    0 May 15 14:34 you_are_here
# cd LINK1
# ls -la
total 8
drwxr-xr-x. 2 root root 4096 May 15 14:34 .
drwxr-xr-x. 3 root root 4096 Apr 29 16:47 ..
-rw-r--r--. 1 root root    0 May 15 14:34 you_are_here
# pwd
/symlink/LINK1
```

Again, despite the fact that the actual path is `/unix/unix2/nfstree`, we see an ambiguated path of `/symlink/LINK1`. The file `you_are_here` has the same date and timestamp across both paths. Additionally, the symlink created can be mounted instead of the vsroot path, adding an extra level of ambiguity to the export path:

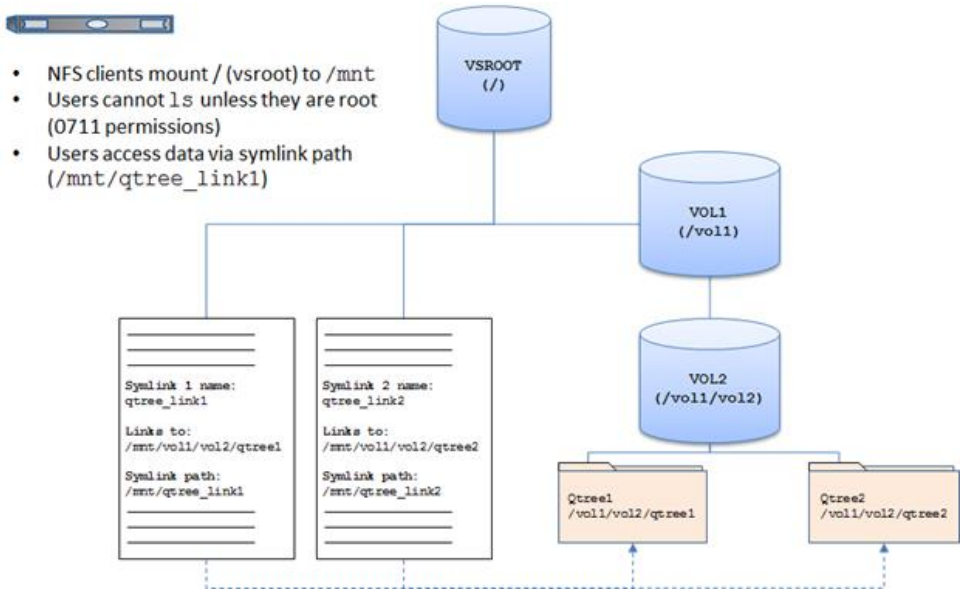
```
# mount -o nfsvers=3 x.x.x.e:/LINK1 /mnt
# mount | grep mnt
x.x.x.e:/LINK1 on /mnt type nfs (rw,nfsvers=3,addr=x.x.x.e
# cd /mnt
# pwd
/mnt
```

One use case for this setup is with automounters. Every client can mount the same path and never actually know where in the directory structure they are. `If clients mount the SVM root volume (/), be sure to lock down the volume to nonadministrative clients.`

For more information about locking down volumes to prevent listing of files and folders, see the section in this document called “Limiting access to the SVM root volume.”

Figure 3 shows a sample of how a namespace can be created to leverage symlinks to create ambiguation of paths for NAS operations.

Figure 3) Symlink example using vsroot.



Note: Export policies and rules can be applied to volumes and qtrees, but not folders or symlinks. This fact should be taken into consideration when creating symlinks for use as mount points. Symlinks instead inherit the export policy rules of the parent volume in which the symlink resides.

NetApp FlexGroup volumes

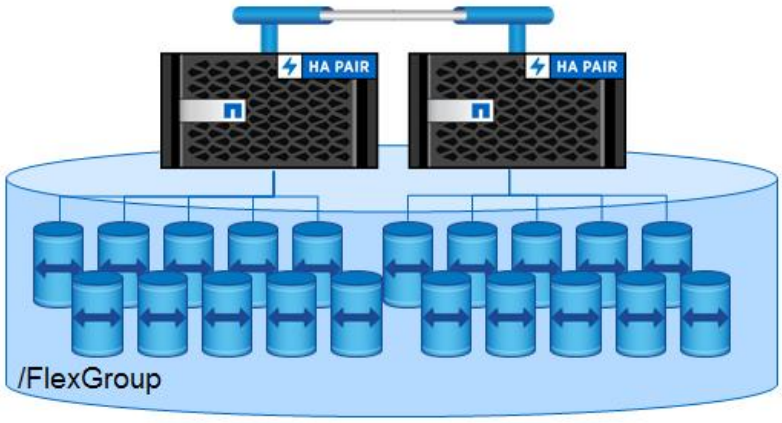
Beginning in ONTAP 9.1, a new way to present storage to NAS (CIFS/SMB and NFS) clients is introduced: NetApp FlexGroup volumes.

A FlexGroup volume is intended to provide the following benefits:

- Immense capacity for a single mount point (up to 20PB, 400 billion files)
- Performance gains over FlexVol by way of concurrency of ingest operations
- Ease of deployment and management

A NetApp FlexGroup volume provides NFS clients with a true global namespace — a single large bucket of storage that spans multiple nodes in a cluster and provides parallel metadata operations for NAS workloads.

Figure 4) NetApp FlexGroup volumes.



Ideal use cases

A FlexGroup volume works best with workloads that are heavy on ingest (a high level of new data creation), heavily concurrent, and evenly distributed among subdirectories:

- Electronic design automation (EDA)
- Log file repositories
- Software build/test environments (such as GIT)
- Seismic/oil and gas
- Media assets or HIPAA archives
- File streaming workflows
- Unstructured NAS data (such as home directories)
- Big data/data science
- Artificial Intelligence and Machine Learning
- Home directories

For more information about FlexGroup volumes, see the following resources:

- TR-4571: NetApp FlexGroup Volume Best Practices and Implementation Guide
- TR-4571-a: Top Best Practices – NetApp ONTAP FlexGroup Volumes
- TR-4617: Electronic Design Automation Best Practices
- TR-4678: Data Protection and Backup – NetApp ONTAP FlexGroup Volumes

NetApp FlexCache volumes

ONTAP 9.5 introduced a new feature for balancing read-heavy NFS workloads across multiple nodes called NetApp FlexCache volumes.

FlexCache in ONTAP provides a writable, persistent cache of a volume in a remote place. A cache is a temporary storage location that resides between a host and a source of data. The objective of a cache is to store frequently accessed portions of source data in a way that allows the data to be served faster than it would be by fetching the data from the source. Caches are beneficial in read-intensive environments where data is accessed more than once and is shared by multiple hosts. A cache can serve data faster in one of two ways:

- The cache system is faster than the system with the data source. This can be achieved through faster storage in the cache (for example, solid-state drives (SSD) versus HDD), increased processing power in the cache, and increased (or faster) memory in the cache.
- The storage space for the cache is physically closer to the host, so it does not take as long to reach the data.

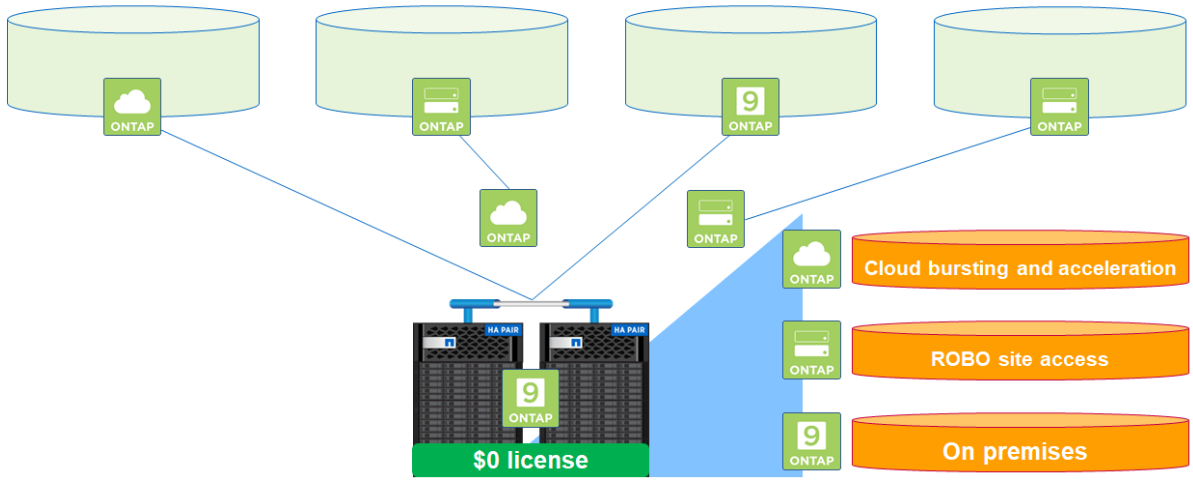
Caches are implemented with different architectures, policies, and semantics so that the integrity of the data is protected as it is stored in the cache and served to the host.

FlexCache offers the following benefits:

- Improved performance by providing load distribution
- Reduced latency by locating data closer to the point of client access
- Enhanced availability by serving cached data in a network disconnection situation

FlexCache provides all the above advantages while maintaining cache coherency, data consistency, and efficient use of storage in a scalable and high-performing manner.

Figure 5) NetApp FlexCache volumes.



A FlexCache is a sparse copy; not all files from the origin dataset can be cached, and, even then, not all data blocks of a cached inode can be present in the cache. Storage is used efficiently by prioritizing retention of the working dataset (recently used data).

With FlexCache, the management of disaster recovery and other corporate data strategies only needs to be implemented at the origin. Because data management is only on the source, FlexCache enables better and more efficient use of resources and simpler data management and disaster recovery strategies.

For more information on FlexGroup volumes, see the following resource:

- TR-4743: FlexCache in ONTAP

File locking concepts

File locking is a way that applications can preserve the integrity of a file when it is open and in use by notifying other clients that attempt to open the file that it is currently locked. With NFS, file locking mechanisms depend on the NFS version being used.

NFSv3 locking

NFSv3 uses ancillary protocols like Network Lock Manager (NLM) and Network Status Monitor (NSM) to coordinate file locks between the NFS client and server. NLM helps establish and release locks, while NSM notifies peers of server reboots. With NFSv3 locking, when a client reboots, the server has to release the locks. When a server reboots, the client reminds the server of the locks it held. In some cases, the lock mechanisms do not communicate properly and stale locks are leftover on the server and must be manually cleared.

NFSv4.x locking

NFSv4.x uses a lease-based locking model that is integrated within the NFS protocol. This means there are no ancillary services to maintain or worry about; all the locking is encapsulated in the NFSv4.x communication.

When a server or client reboots, if the lock cannot be re-established during a specified grace period, then the lock will expire. ONTAP NFS servers control this lock timeout period with the options `-v4-grace-seconds` and `-v4-lease-seconds`.

- `-v4-lease-seconds` refers to how long a lease is granted before the client has to renew the lease. The default is 30 seconds, with a minimum of 10 seconds and maximum of -1 second of the value of `-v4-grace-seconds`.
- `-v4-grace-seconds` refers to how long a client attempts to reclaim a lock from ONTAP during a reboot of a node (such as during failovers/givebacks). The default is 45 seconds and can be modified with a range of +1 second of the `-v4-lease-seconds` value and a maximum of 90 seconds.

In rare cases, locks may not be freed as quickly as stated by the lease seconds value, which results in the locks being freed over the course of two lease periods. For example, if grace seconds is set to 45 seconds, it may take 90 seconds to free the lock. For more information, see bug 957529. For information about the impact of these values to storage failovers, see “Impact of NFSv4.x locks on failover.”

Multiprotocol NAS lock behavior

When using file locks in multiprotocol NAS environments, you should be aware of a difference in behavior depending on the NAS protocol in use.

- If the NAS client is SMB, file locks are mandatory locks.
- If the NAS client is NFS, file locks are advisory locks.

What this means

Because of differences between the NFS and SMB file locks, an NFS client might fail to access a file previously opened by an SMB application.

The following occurs when an NFS client attempts to access a file locked by an SMB application:

- In mixed or NTFS volumes, file manipulation operations such as `rm`, `rmdir`, and `mv` can cause the NFS application to fail.
- NFS read and write operations are denied by SMB deny-read and deny-write open modes, respectively.
- NFS write operations fail when the written range of the file is locked with an exclusive SMB bytelock. In UNIX security-style volumes, NFS unlink and rename operations ignore the SMB lock state and allow access to the file. All other NFS operations on UNIX security-style volumes honor the SMB lock state.

For more information on multiprotocol NAS, see the section “Multiprotocol NAS.”

Lock types

There are several types of NFS locks, which include:

- **Shared locks.** Shared locks can be used by multiple processes at the same time and can only be issued if there are no exclusive locks on a file. These are intended for read-only work but can be used for writes (such as with a database).
- **Exclusive locks.** These operate the same as exclusive locks in CIFS/SMB – only one process can use the file when there is an exclusive lock. If any other processes have locked the file, an exclusive lock cannot be issued, unless that process was forked.
- **Delegations.** Delegations are used only with NFSv4.x and are assigned when the NFS server options are enabled, and the client supports NFSv4.x delegations. Delegations provide a way to cache operations on the client side by creating a "soft" lock to the file being used by a client. This helps improve some aspects of performance for operations by reducing the number of calls being made between the client and server and are similar to SMB opportunistic locks. For more information about delegations, see "NFSv4.1 delegations."
- **Byte-range locks.** Rather than locking an entire file, byte-range locks only lock a portion of a file.

  Note: Locking behavior is dependent on the type of lock, the client operating system version and the NFS version being used. Be sure to test locking in your environment to gauge the expected behavior.

For more information about file locking in ONTAP, see the product documentation section called "Managing file locks."

Manually establishing locks on a client

To test NFS locks, the client has to tell the NFS server to establish a lock. However, not all applications use locks. For example, an application like "vi" will not lock a file; instead, it creates a hidden swap file in the same folder and then commits writes to that file when the application is closed. Then the old file is deleted and the swap file gets renamed to the filename.

There are utilities to manually establish locks, however. For example, flock can lock files.

1. To establish a lock on a file, first run `exec` to assign a numeric ID.

```
# exec 4<>v4user_file
```

2. Use flock to create a shared or exclusive lock on the file.

```
# flock

Usage:
 flock [options] <file|directory> <command> [command args]
 flock [options] <file|directory> -c <command>
 flock [options] <file descriptor number>

Options:
 -s  --shared             get a shared lock
 -x  --exclusive          get an exclusive lock (default)
 -u  --unlock             remove a lock
 -n  --nonblock           fail rather than wait
 -w  --timeout <secs>     wait for a limited amount of time
 -E  --conflict-exit-code <number>  exit code after conflict or timeout
 -o  --close              close file descriptor before running command
 -c  --command <command>  run a single command string through the shell

 -h, --help     display this help and exit
 -V, --version  output version information and exit

# flock -n 4
```

3. Check the ONTAP SVM for the lock.

```
cluster::*> vserver locks show -vserver DEMO

Notice: Using this command can impact system performance. It is recommended
that you specify both the vserver and the volume when issuing this command to
minimize the scope of the command's operation. To abort the command, press Ctrl-C.

Vserver: DEMO
Volume   Object Path               LIF         Protocol  Lock Type   Client
-------- ------------------------- ----------- --------- ----------- ----------
home     /home/v4user_file         data2       nlm       byte-range  10.x.x.x
               Bytelock Offset(Length): 0 (18446744073709551615)
```

4. Unlock the file.

```
# flock -u -n 4
```

Note: Manually locking files allows you to test file open and edit interactions, as well as seeing how file locks handle storage failover events.

Export concepts

Volumes in ONTAP are shared to NFS clients by exporting a path that is accessible to a client or set of clients. When a volume is mounted to the SVM's namespace, a file handle is created and presented to NFS clients when requested in a mount command. Permissions to these exports are defined by export policies and rules, which are configurable by storage administrators.

Export policy and rule concepts

ONTAP offers export policies as containers for export policy rules to control security. These policies are stored in a replicated database, thus making exports available across every node in the cluster, rather than isolated to a single node.

To provide or restrict NFS access to these volumes, export policy rules are created. These rules can define read, write, and root access, as well as specifying client lists. Multiple rules can exist in a policy and multiple clients can exist in a single rule.

The default export policy

A newly created SVM contains an export policy called Default. This export policy cannot be deleted, although it can be renamed or modified. When an NFS server is created, the Default policy is automatically created and applied to the vsroot volume. However, the Default policy will have no export rules in it, so access to volumes using the default export policy will not be able to mount until rules are added. When new volumes are created, if no export policy is defined, then the export policy of the vsroot volume is inherited.

Vsroot and volume traversal

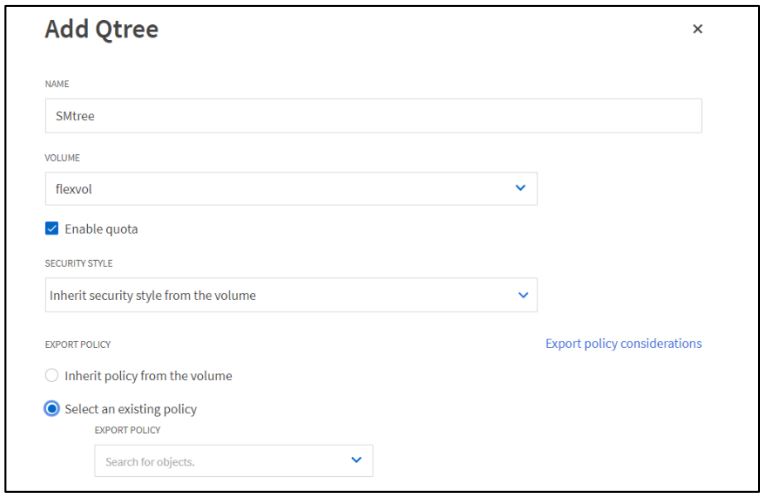
Because export policies are inherited by default, NetApp recommends opening read access to the root volume of the SVM (vsroot) to NFS clients when a rule is assigned. Setting any rules for the "default" export policy that restrict read access to the vsroot denies traversal to the volumes created under that SVM and will cause mounts to fail. That is because vsroot is "/" in the path to "/junction" and factors into the ability to mount and traverse.

Qtree exports

In ONTAP, it is possible to set export policies and rules for volumes, as well as underlying qtrees. This offers a way to restrict/allow client access to storage-managed directories in ONTAP, which can help storage administrators more easily manage workloads such as home directories.

By default, qtrees will inherit the export policy of the parent volume. You can explicitly choose or create an export policy and rule when creating qtrees in ONTAP System Manager, or by using the `-export-policy` CLI option.

Figure 6) Qtree export specification — ONTAP System Manager.



Qtree ID considerations

After a noninherited export policy is applied to a qtree, NFS file handles change slightly when dealing with operations between qtrees. For more information, see the section “Qtree IDs and rename behavior.”

Access control to vsroot

To control access to read/write to vsroot, use the volume unix-permissions and/or ACLs. NetApp recommends restricting the ability for nonowners of the volume to write to vsroot (at most, 0755 permissions).

When volumes are created, the following values are the defaults, unless specified otherwise:

- 0755 is the default UNIX security set on volumes
- The default owner is UID 0, and the default group is GID 1

To provide the ability to traverse vsroot that also prevents read/list access to NFS clients that can mount “/”, there are two approaches.

Option 1: Lock down UNIX mode bits on vsroot

The simplest way to lock down vsroot to users is to manage the ownership and permissions from the cluster:

1. Create a local UNIX user that is unique to the SVM. For example, this UNIX user could be the same name as the SVM itself.
2. Set the vsroot volume to the new UNIX user. Most NFS clients have a “root” user, which means, by default, the vsroot volume may have too much access by the root user.
3. Use UNIX permissions that limit groups and others to only traverse permissions but leaves desired permissions for the volume owner. (For example, 0611)

Option 2: Use NFSv4.x or NTFS ACLs to lock down vsroot

Another way to lock down vsroot is to leverage ACLs to limit permissions to traverse for everyone except a select few users or groups. This can be done via NFSv4.x ACLs (even if you mount using NFSv3) or

through NTFS permissions in environments that are also serving CIFS/SMB protocols. For information about using NFSv4.x ACLs with NFSv3 mounts, see “Using NFSv4.x ACLs with NFSv3 mounts.”

Export policy rules: Options

The Appendix of this document lists the various options used for export policy rules and what they are used for, as well as examples. Most export policy rule options can be viewed using the `export-policy rule show` command or using ONTAP System Manager.

Export policy rules: Inheritance

In ONTAP, export policy rules affect only the volumes and qtrees they are applied to. For example, if the SVM root volume has a restrictive export policy rule that limits root access to a specific client or subset of clients, the data volumes that exist under the SVM root volume (which is mounted at “`/`”) honor only the export policies applied to them. The one exception is if a volume has an export policy rule that denies read access to a client and that client must traverse that volume in that path. There is currently not concept of “bypass traverse checking” for NFS in ONTAP. For an example of export policy rule inheritance, see “Export policy rule inheritance example.”

Export policy rules: Index

ONTAP offers storage administrators the ability to set the priority for export policy rules so that they are honored in a specific order. The policy is evaluated when access is attempted and the rules are read in order from 0 to 999999999.

Note: A rule index of 999999999 is an absolute maximum, but NetApp does not recommend it. Use more sensible numbers for the index.

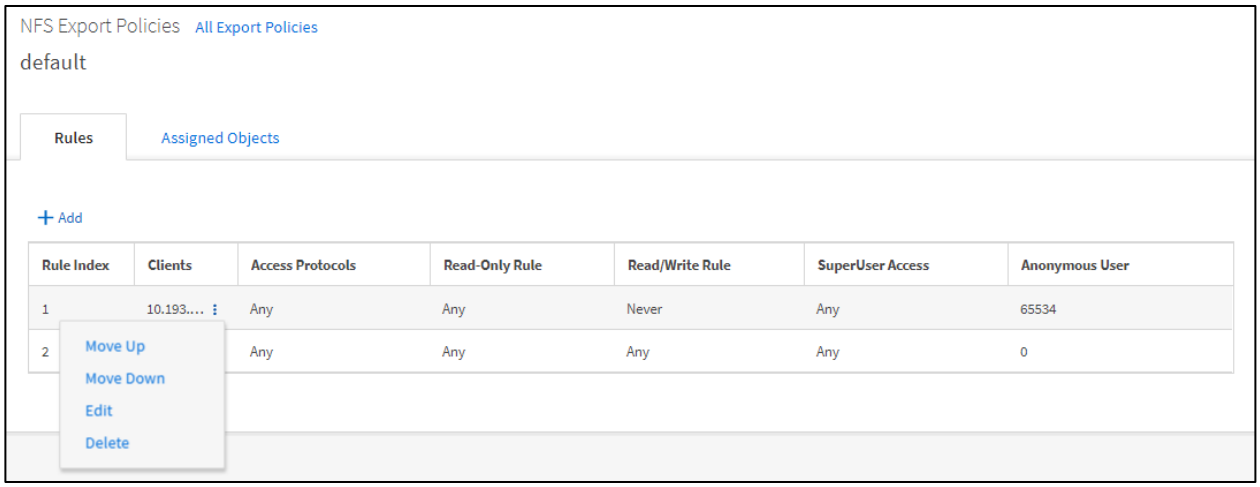
If a rule index with a higher number (such as 1) is read and has allowed access for a subnet but later a host that is in that subnet is denied access through a rule at a lower index (such as 99), then that host is granted access based on the rule that allows access being read earlier in the policy.

Conversely, if a client is denied access through an export policy rule at a higher index and then allowed access through a global export policy rule later in the policy (such as 0.0.0.0/0 client match), then that client is denied access.

It is possible to reorder the rule index for policy rules with the `export-policy rule setindex` command, or in ONTAP System Manager using “Move Up/Move Down.”

Figure 7 shows the reordering of the rule index in ONTAP System Manager.

Figure 7) Reordering the rule index in ONTAP System Manager.



It is important to consider the order of the export policy rules when determining the access that is and is not allowed for clients in ONTAP. If you use multiple export policy rules, be sure that rules that deny or allow access to a broad range of clients do not step on rules that deny or allow access to those same clients. Rule index ordering factors in when rules are read; higher-number rules override lower-number rules in the index.

Note: When you are using more granular rules (such as for a specific client, such as an administrative host), they should be placed higher up in the rule index. Broader access rules should be placed lower. For example, an administrative host rule would be at rule index 1 and a policy for 0.0.0.0/0 would be at index 99.

Export policy rules: Clientmatch

The clientmatch option in an export policy rule allows storage administrators to define an access list for mounting NFS exports, as well as a way to control access permissions at a high level after a client is able to mount the export.

Valid entries for the NFS export policy rule clientmatch include:

- IP addresses
- Host names
- Domains
- Subnets
- Netgroups

  Note: In ONTAP 9.1 and later, it's possible to define multiple comma-separated IP addresses or host names in a single rule, rather than needing to create unique policy rules for each.

The following considerations should be made:

- When host names are used for the clientmatch field or in netgroups, a working DNS server or manual host entries must be available to resolve the host names to IP addresses.
- When netgroups are used, an @ sign should be appended to the front of the netgroup to let ONTAP know that you are specifying a netgroup rather than a host name.
- If relying on name services for name resolution or netgroup lookups, ensure there is a data LIF in the SVM that can reach the necessary name services.
- For more information about name services, see TR-4668: Name Services Best Practice Guide.

Export policy rules: Caching

Export policy rules, client host names, and netgroup information are all cached in ONTAP to reduce the number of requests made to the cluster. This helps improve performance of requests, as well as alleviating load on networks and name service servers.

Clientmatch caching

When a clientmatch entry is cached, it is kept local to the SVM and then flushes after the cache timeout period is reached or if the export policy rule table is modified. The default cache timeout period is dependent on the version of ONTAP and can be verified using the command `export-policy access-cache config show` in admin privilege.

These are the default values:

```
TTL For Positive Entries (Secs): 3600
TTL For Negative Entries (Secs): 3600
Harvest Timeout (Secs): 86400
```

To view a specific client in the export policy access-cache, use the following advanced privilege command:

```
cluster::*> export-policy access-cache show -node node-02 -vserver NFS -policy default -address
x.x.x.x

                                        Node: node-02
                                     Vserver: NFS
                                  Policy Name: default
                                   IP Address: x.x.x.x
                     Access Cache Entry Flags: has-usable-data
                                  Result Code: 0
                First Unresolved Rule Index: -
                     Unresolved Clientmatch: -
              Number of Matched Policy Rules: 1
         List of Matched Policy Rule Indexes: 2
                                Age of Entry: 11589s
                  Access Cache Entry Polarity: positive
Time Elapsed since Last Use for Access Check: 11298s
      Time Elapsed since Last Update Attempt: 11589s
                Result of Last Update Attempt: 0
                 List of Client Match Strings: 0.0.0.0/0
```

Host name/DNS caching

When a clientmatch is set to a host name, the name is then resolved to an IP address. This happens based on the order the SVM's name service-switch (ns-switch) uses. For example, if the ns-switch host database is set to files,dns, then ONTAP searches for the client match in local host files and then searches DNS.

After a name lookup, ONTAP caches the result in the host cache. This cache's settings are configurable and can be queried and flushed from the ONTAP CLI in advanced privilege.

To query the cache, run the following commands:

```
cluster::*> name-service cache hosts forward-lookup show -vserver NFS
(vserver services name-service cache hosts forward-lookup show)
                    IP       Address IP                     Create
Vserver   Host      Protocol Family  Address        Source  Time       TTL(sec)
--------- --------- -------- ------- -------------- ------- ---------- --------
NFS       centos7.ntap.local
                    Any      Ipv4    x.x.x.x  dns     3/26/2020  3600
                                                           16:31:11
```

To view the hosts cache settings, run the following commands:

```
cluster::*> name-service cache hosts settings show -vserver NFS -instance
  (vserver services name-service cache hosts settings show)

                    Vserver: NFS
          Is Cache Enabled?: true
Is Negative Cache Enabled?: true
               Time to Live: 24h
     Negative Time to Live: 1m
     Is TTL Taken from DNS: true
```

In some cases, if an NFS client's IP address changes, the hosts entry might need to be flushed to correct access issues.

To flush a hosts cache entry, run the following commands:

```
cluster::*> name-service cache hosts forward-lookup delete -vserver NFS ?
     -host      -protocol  -sock-type -flags     -family
```

Netgroup caching

If using netgroups in the clientmatch field for export rules, then ONTAP does additional work to contact the netgroup name service server to unpack the netgroup information. The netgroup database in ns-switch determins the order in which ONTAP queries for netgroups. In addition, the method ONTAP uses for netgroup support is dependent on if netgroup.byhost support is enabled or disabled. For more information about netgroup.byhost, see TR-4835: How to Configure LDAP in ONTAP.

- If netgroup.byhost is disabled, then ONTAP queries the entire netgroup and populates the cache with all netgroup entries. If the netgroup has thousands of clients, then that process could take some time to complete. Netgroup.byhost is disabled by default.
- If netgroup.byhost is enabled, then ONTAP queries the name service only for the host entry and the associated netgroup mapping. This greatly reduces the amount to time needed to query for netgroups, as we don't need to look up potentially thousands of clients.

These entries are added to the netgroup cache, which is found in `vserver services name-service cache` commands. These cache entries can be viewed or flushed, and the timeout values can be configured.

To view the netgroups cache settings, run the following commands:

```
cluster::*> name-service cache netgroups settings show -vserver NFS -instance
  (vserver services name-service cache netgroups settings show)

                    Vserver: NFS
         Is Cache Enabled?: true
Is Negative Cache Enabled?: true
              Time to Live: 24h
     Negative Time to Live: 1m
  TTL for netgroup members: 30m
```

When an entire netgroup is cached, it gets placed in the members cache.

```
cluster::*> name-service cache netgroups members show -vserver DEMO -netgroup netgroup1
  (vserver services name-service cache netgroups members show)

            Vserver: DEMO
           Netgroup: netgroup1
              Hosts: sles15-1,x.x.x.x
        Create Time: 3/26/2020 12:40:56
Source of the Entry: ldap
```

When only a single netgroup entry is cached, the IP-to-netgroup and hosts reverse-lookup caches are populated with the entry.

```
cluster::*> name-service cache netgroups ip-to-netgroup show -vserver DEMO -host x.x.x.y
  (vserver services name-service cache netgroups ip-to-netgroup show)
Vserver    IP Address  Netgroup     Source  Create Time
---------- ----------- ------------ ------- -----------
DEMO       x.x.x.y
                       netgroup1    ldap    3/26/2020 17:13:09

cluster::*> name-service cache hosts reverse-lookup show -vserver DEMO -ip x.x.x.y
  (vserver services name-service cache hosts reverse-lookup show)
Vserver    IP Address      Host                 Source Create Time     TTL(sec)
---------- --------------- -------------------- ------ -------------- --------
DEMO       x.x.x.y   centos8-ipa.centos-ldap.local
                                                dns    3/26/2020 17:13:09
                                                                      3600
```

Cache timeout modification considerations

Cache configurations can be modified to different values if needed:

- **Increasing** the timeout values keeps cache entries longer but can result in inconsistencies in client access if a client changes its IP address (for example, if DHCP is used for client IP addresses and DNS does not get updated or the export rule uses IP addresses).
- **Decreasing** the timeout values flushes the cache more frequently for more up-to-date information but could add additional load to name service servers and add latency to mount requests from clients.

In most cases, leaving the cache timeout values intact is the best approach. For more information and guidance, see TR-4668: Name Services Best Practices and TR-4835: How to Configure LDAP in ONTAP.

Exportfs support

In ONTAP, `exportfs` is replaced by the `export-policy` and `name-service` cache commands. When running `exportfs`, the following would be seen:

```
  "exportfs" is not supported: use the "vserver export-policy" command.
```

Export policy rules: Access verification

ONTAP offers a command (`export-policy check-access`) that allows you to check an export policy's access rule set against a client's access to help determine if an export policy rule is working properly for pre-deployment as well as troubleshooting. Its functionality is similar to `exportfs -c` functionality. This command leverages all the normal name service communication and cache interaction that a standard mount from an NFS client would use.

Example of `export-policy check-access`:

```
cluster1::*> vserver export-policy check-access -vserver vs1 -client-ip 1.2.3.4 -volume flex_vol
-authentication-method sys -protocol nfs3 -access-type read

                                        Policy    Policy       Rule
Path                          Policy    Owner     Owner Type  Index Access
----------------------------- --------- --------- ---------- ------ ----------
/                             default   vs1_root  volume          1 read
/dir1                         default   vs1_root  volume          1 read
/dir1/dir2                    default   vs1_root  volume          1 read
/dir1/dir2/flex1              data      flex_vol  volume         10 read
```

The anon user

The anonymous (anon) user ID specifies a UNIX user ID or user name that is mapped to client requests that arrive without valid NFS credentials. This can include the root user. ONTAP determines a user's file access permissions by checking the user's effective UID against the SVM's specified name-mapping and name-switch methods. After the effective UID is determined, the export policy rule is leveraged to determine the access that UID has.

The `-anon` option in export policy rules allows specification of a UNIX user ID or user name that is mapped to client requests that arrive without valid NFS credentials (including the root user). The default value of `-anon`, if not specified in export policy rule creation, is 65534. This UID is normally associated with the user name "nobody" or "nfsnobody" in Linux environments. NetApp appliances use 65534 as the user "pcuser," which is generally used for multiprotocol operations, Because of this difference, if using local files and NFSv4, the name string for users mapped to 65534 might not match. This discrepancy might cause files to be written as the user specified in the `/etc/idmapd.conf` file on the client (Linux) or `/etc/default/nfs` file (Solaris), particularly when using multiprotocol (CIFS and NFS) on the same datasets.

The root user

The root user must be explicitly configured in ONTAP to specify which machine has root access to a share, or else anon=0 must be specified. Alternatively, the `-superuser` option can be used if more

granular control over root access is desired. If these settings are not configured properly, permission denied might be encountered when accessing an NFS share as the root user (0). If the `-anon` option is not specified in export policy rule creation, the root user ID is mapped to the Nobody user (`65534`).

AUTH types

When an NFS client authenticates, an AUTH type is sent. An AUTH type specifies how the client is attempting to authenticate to the server and depends on client-side configuration. Supported AUTH types include:

- **AUTH_NONE/AUTH_NULL.** This AUTH type specifies that the request coming in has no identity (NONE or NULL) and is mapped to the anon user. For more information, see http://www.ietf.org/rfc/rfc1050.txt and http://www.ietf.org/rfc/rfc2623.txt.
- **AUTH_SYS/AUTH_UNIX.** This AUTH type specifies that the user is authenticated at the client (or system) and comes in as an identified user. For more information, see http://www.ietf.org/rfc/rfc1050.txt and http://www.ietf.org/rfc/rfc2623.txt.
- **AUTH_RPCGSS.** This is Kerberized NFS authentication.

There are several ways to configure root access to an NFS share. For examples, see "Examples of controlling the root user."

Limiting access to the SVM root volume

By default, when an SVM is created, the root volume is configured with 755 permissions and owner:group of root (0): root (0). This means that:

- The user root (0) has effective permissions of 7, or Full Control.
- The Group and Others permission levels are set to 5, which is Read & Execute.

When this is configured, everyone who accesses the SVM root volume can list and read junctions mounted below the SVM root volume, which is always mounted to "/" as a junction-path. In addition, the default export policy rule that is created when an SVM is configured by using System Manager or `vserver setup` commands permits user access to the SVM root.

Example of the default export policy rule created by Vserver setup:

```
cluster::> export-policy rule show -vserver nfs_svm -policyname default -instance
  (vserver export-policy rule show)

                                Vserver: nfs_svm
                            Policy Name: default
                             Rule Index: 1
                          Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0
                           RO Access Rule: any
                           RW Access Rule: any
User ID To Which Anonymous Users Are Mapped: 65534
                  Superuser Security Types: none
              Honor SetUID Bits in SETATTR: true
                 Allow Creation of Devices: true
```

In the preceding export policy rule, all clients have any RO and RW access. Root is squashed to anon, which is set to 65534.

For example, if an SVM has three data volumes, all would be mounted under "/" and could be listed with a basic `ls` command by any user accessing the mount.

```
# mount | grep /mnt
x.x.x.e:/ on /mnt type nfs (rw,nfsvers=3,addr=x.x.x.e)
# cd /mnt
# ls
nfs4  ntfs  unix
```

In some environments, this behavior might be undesirable, because storage administrators might want to limit visibility to data volumes to specific groups of users. Although read and write access to the volumes themselves can be limited on a per-data-volume basis using permissions and export policy rules, users can still see other paths using the default policy rules and volume permissions.

To limit the ability to users to be able to list SVM root volume contents (and subsequent data volume paths) but still allow the traversal of the junction paths for data access, the SVM root volume can be modified to allow only root users to list folders in SVM root. To do this, change the UNIX permissions on the SVM root volume to 0711 using the volume modify command:

```
cluster::> volume modify -vserver nfs_svm -volume rootvol -unix-permissions 0711
```

After this is done, root still has Full Control using the 7 permissions, because it is the owner. Group and Others get Execute permissions as per the one-mode bit, which only allows them to traverse the paths using cd.

When a user who is not the root user attempts `ls`, that user has access denied.

```
sh-4.1$ ls
ls: cannot open directory .: Permission denied
```

In many cases, NFS clients log into their workstations as the root user. With the default export policy rule created by System Manager and Vserver setup, root access is limited.

```
# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023
# ls -la
ls: cannot open directory .: Permission denied
```

This is because the export policy rule attribute superuser is set to None. If root access is desired by certain clients, this can be controlled by adding export policy rules to the policy and specifying the host IP, name, netgroup, or subnet in the clientmatch field. When creating this rule, list it ahead of any rule that might override it, such as a clientmatch of 0.0.0.0/0 or 0/0, which is all hosts.

The following is an example of adding an administrative host rule to a policy:

```
cluster::> export-policy rule create -vserver nfs_svm -policyname default -clientmatch x.x.x.x -
rorule any -rwrule any -superuser any -ruleindex 1

cluster::> export-policy rule show -vserver nfs_svm -policyname default -ruleindex 1
  (vserver export-policy rule show)

                                Vserver: nfs_svm
                            Policy Name: default
                             Rule Index: 1
                          Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: x.x.x.x
                           RO Access Rule: any
                           RW Access Rule: any
User ID To Which Anonymous Users Are Mapped: 65534
                  Superuser Security Types: any
              Honor SetUID Bits in SETATTR: true
                 Allow Creation of Devices: true


cluster::> export-policy rule show -vserver nfs_svm -policyname default
  (vserver export-policy rule show)
             Policy          Rule    Access   Client                RO
Vserver      Name            Index   Protocol Match                 Rule
------------ --------------- ------  -------- --------------------- ---------
nfs_svm  default         1       any      x.x.x.x        any
nfs_svm  default         2       any      0.0.0.0/0              any
2 entries were displayed.
```

Now the client is able to see the directories as the root user.

```
# ifconfig | grep "inet addr"
          inet addr:x.x.x.x  Bcast:x.x.225.255  Mask:255.255.255.0
          inet addr:127.0.0.1  Mask:255.0.0.0
# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023
# ls
nfs4  ntfs  unix
```

Other clients are not able to list contents as root.

```
# ifconfig | grep "inet addr"
          inet addr:x.x.x.y  Bcast:x.x.225.255  Mask:255.255.255.0
# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023
# mount | grep mnt
x.x.x.e:/ on /mnt type nfs (rw,nfsvers=3,addr=x.x.x.e)
# ls /mnt
ls: cannot open directory .: Permission denied
```

For more information about export policy rules and their effect on the root user, review the "Root User" section of this document.

For more information about mode bits, see the following link: http://www.zzee.com/solutions/unix-permissions.shtml.

Mapping all UIDs to a single UID (squash_all)

In some cases, storage administrators might want to control which UID (such as root) some or all users map to when coming in through NFS to a UNIX-security-style volume. If a volume has NTFS security style, doing so is as simple as setting a default Windows user in the NFS server options. However, when the volume is UNIX security style, no name mapping takes place when coming in from NFS clients. To control this situation, you can create an export policy rule.

Squashing all UIDs to 65534

The following export policy rule example shows how to force all UIDs (including root) coming into the system from a specific subnet to use the 65534 UID. This rule can be used to create guest access policies for users to limit access. This is done with the RO, RW, and superuser authentication type of None, anon value of 65534, and the clientmatch value specifying the subnet.

```
                                  Vserver: nfs_svm
                              Policy Name: default
                               Rule Index: 1
                          Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: x.x.225.0/24
                            RO Access Rule: none
                            RW Access Rule: none
User ID To Which Anonymous Users Are Mapped: 65534
                   Superuser Security Types: none
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true
```

Making all UIDs root

The following export policy rule example shows how to force all UIDs (including root) coming into the system from a specific subnet to use the UID associated with root (0). This rule can be used to allow full access to users in a specific subnet to avoid overhead on permissions management. This access is enabled with the RO and RW authentication type of None, superuser value of None, anon value of 0, and the clientmatch value specifying the subnet.

For example:

```
                                    Vserver: nfs_svm
                                Policy Name: default
                                 Rule Index: 1
                            Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: x.x.225.0/24
                             RO Access Rule: none
                             RW Access Rule: none
User ID To Which Anonymous Users Are Mapped: 0
                    Superuser Security Types: none
                Honor SetUID Bits in SETATTR: true
                   Allow Creation of Devices: true
```

Special character considerations

Most common text characters in Unicode (when they are encoded with UTF-8 format) use encoding that is equal to or smaller than three bytes. This common text includes all modern written languages, such as Chinese, Japanese, and German. However, with the popularity of special characters such as the emoji, some UTF-8 character sizes have grown beyond three bytes. For example, a trophy symbol is a character that requires four bytes in UTF-8 encoding.

Special characters include, but are not limited to, the following:

- Emojis
- Music symbols
- Mathematical symbols

When a special character is written to a FlexGroup volume, the following behavior occurs:

```
# mkdir /flexgroup4TB/🏆
mkdir: cannot create directory ‘/flexgroup4TB/\360\237\217\206’: Permission denied
```

In the preceding example, `\360\237\217\206` is hex `0xF0 0x9F 0x8F 0x86` in UTF-8, which is a trophy symbol.

ONTAP software did not natively support UTF-8 sizes that were greater than three bytes in NFS, as indicated in bug 229629. To handle character sizes that exceeded three bytes, ONTAP placed the extra bytes into an area in the operating system known as `bagofbits`. These bits were stored until the client requests them. Then the client interprets the character from the raw bits. FlexVol supports `bagofbits`, in all ONTAP releases and FlexGroup volumes added support for `bagofbits` in ONTAP 9.2.

Also, ONTAP has an event management system message for issues with `bagofbits` handling.

```
Message Name: wafl.bagofbits.name
Severity: ERROR

Corrective Action: Use the "volume file show-inode" command with the file ID and volume name
information to find the file path. Access the parent directory from an NFSv3 client and rename
the entry using Unicode characters.

Description: This message occurs when a read directory request from an NFSv4 client is made to a
Unicode-based directory in which directory entries with no NFS alternate name contain non-Unicode
characters.
```

Support for utf8mb4 volume language

As previously mentioned, special characters might exceed the supported three bytes UTF-8 encoding that is natively supported. ONTAP then uses the `bagofbits` functionality to allow these characters to work.

This method for storing inode information is not ideal, so starting in ONTAP 9.5, utf8mb4 volume language support was added. When a volume uses this language, special characters that are four bytes in size will be stored properly and not in `bagofbits`.

Volume language is used to convert names sent by NFSv3 clients to Unicode, and to convert on-disk Unicode names to the encoding expected by NFSv3 clients. In legacy situations in which NFS hosts are configured to use non-UTF-8 encodings, you will want to use the corresponding volume language. Use of UTF-8 has become almost universal these days, so the volume language is likely to be UTF-8.

NFSv4 requires use of UTF-8, so there is no need to use non-UTF-8 encoding for NFSv4 hosts. Similarly, CIFS uses Unicode natively, so it will work with any volume language. However, use of utf8mb4 is recommended because files with Unicode names above the basic plane are not converted properly on non-utf8mb4 volumes.

Potential issues with UTF-8 characters

In some instances, UTF-8 file names with character representation containing 0x80 (octal \0200) are not able to be managed using NFS mounts. Many of these characters occur in the Unicode General Punctuation block. For example, the name '`test•file`' is encoded as '`test\xe2\x80\xa2file`' and that name might be affected because it contains 0x80 in the UTF-8 sequence. See bug 998468 for details.

Note: This issue only affects files created prior to ONTAP 8.3.2

The option `-v3-search-unconverted-filename` was added in ONTAP 9 to avoid this issue.

```
[-v3-search-unconverted-filename {enabled|disabled}] - Lookup for the filename in unconverted
language if converted language lookup fails (privilege: advanced)
This optional parameter specifies whether to continue the search with unconverted name while
doing lookup in a directory.
```

NFS version considerations

NFSv2 considerations

NFSv2 support was deprecated in ONTAP 8.2 and is no longer available to use.

NFSv3 considerations

The following section covers functionality, known issues, and considerations with NFSv3 in ONTAP.

What happens during NFSv3 mounts?

When mounting a file system over NFSv3, the following steps occur:

1. A Remote Procedure Call (RPC) is made to port 111 (portmapper) of the NFS server to attempt a TCP connection through the portmapper.
2. When the RPC has been acknowledged, portmapper issues a GETPORT call to port 111 of the NFS server data LIF to obtain which port NFS is allowed on.
3. The NFS server returns the port 2049 (NFS) to the client.
4. The client then closes the connection to port 111.
5. A new RPC call is made to port 2049 of the NFS server data LIF.
6. The NFS server returns the call successfully, and the client sends an NFS NULL call to port 2049 of the NFS server's data LIF. This checks whether the parent volume export policy rule allows access to the mount. In this case, the parent volume is mounted to `/`, or the SVM root.
7. The NFS NULL call is returned successfully, and the client proceeds with the mount attempt.

8. Portmapper sends another GETPORT call to the NFS server's data LIF asking for the `mountd` port and provides the credentials, NFS version number, and whether the mount uses TCP or UDP.
9. The cluster checks the NFS settings and verifies whether the credentials supplied are allowed to mount based on the exported volume's export policy rules. Name service servers (such as DNS, NIS, LDAP) are contacted as necessary. Caches are populated. If the NFS version or TCP/UDP is not allowed, the client reports the error.
10. The NFS server replies successfully if the version provided is supported and if the mount can use the specified TCP or UDP connection. It also replies if the AUTH security provider is supported (AUTH_SYS or AUTH_GSS, for example).
11. When the GETPORT call passes, the client issues a V3 MNT call to the junction path specified in the `mount` command through port 635 (`mountd`) of the NFS server data LIF.
12. ONTAP looks for the provided path to the export. If the entry exists, the cluster gathers the file handle information, which is unique to each volume.
13. The NFS server returns the file handle to the client, as well as replies which AUTH varieties are supported by the export policy rule. If the AUTH variety provided by the server matches what the client sent, the mount succeeds.
14. Portmapper from the client then sends another GETPORT call for NFS, this time providing the client's host name.
15. The NFS server replies with port 2049, and the call succeeds.
16. Another NFS NULL call is made from the client over port 2049 of the NFS data LIF and is acknowledged by the NFS server.
17. A series of NFS packets with FSINFO and PATHCONF information is traded between the client and the NFS server.

Effects of file system ID changes in ONTAP

NFS uses a file system ID (FSID) when interacting between client and server. This FSID lets the NFS client know where data lives in the NFS server's file system. Because ONTAP can span multiple file systems across multiple nodes by way of junction paths, this FSID can change depending on where data lives. Some older Linux clients can have problems differentiating these FSID changes, resulting in failures during basic attribute operations, such as `chown` and `chmod`.

An example of this issue can be found in bug 671319. If you disable the FSID change with NFSv3, be sure to enable the `-v3-64bit-identifiers` option in ONTAP 9. But keep in mind that this option could affect older legacy applications that require 32-bit file IDs.

For information about how this impacts high file count environments, see TR-4571: NetApp FlexGroup Volumes Best Practices and Implementation.

How FSIDs operate with NetApp Snapshot copies

When a NetApp Snapshot™ copy of a volume is created, a copy of a file's inodes is preserved in the file system for late access. The file theoretically exists in two locations.

With NFSv3, even though there are two copies of essentially the same file, the FSIDs of those files are not identical. FSIDs of files are formulated by using a combination of NetApp WAFL inode numbers, volume identifiers, and Snapshot copy IDs. Because every Snapshot copy has a different ID, every Snapshot copy of a file has a different FSID in NFSv3, regardless of the setting of the `-v3-fsid-change` option. The NFS RFC specification does not require FSIDs for a file to be identical across file versions.

FSID changes with storage virtual machine disaster recovery

ONTAP 8.3.1 introduces a new feature to enable disaster recovery for entire SVMs called storage virtual machine disaster recovery (SVM DR). This feature is covered in TR-4015: SnapMirror Configuration and Best Practices Guide.

When SVM DR is used with NFS exports in versions prior to ONTAP 9.0, the FSID of those exports changes, and clients have to remount the exports on the destination system. Otherwise, the clients show `stale` for NFS operations on those mounts. If the mount's FSID should be preserved by the SVM DR relationship, then the destination SVM must be created with the `-is-msid-preserve` option set to True in Diag Privilege mode. When this option is set, SnapMirror relationships used in SVM DR show `-msid-preserve` as True in their `snapmirror show` output. This should be used with caution, because SVM DR updates are asynchronous. The source SVM should be confirmed as down before attempting to write to the destination SVM with the same FSID.

NFSv3 security considerations

NFSv3 is considered less secure than NFSv4.x, but that does not mean there aren't security measures you can put into place. The following sections cover some of the NFSv3 security considerations as the apply to ONTAP.

NFSv3 export rules

One way to secure NFS mounts is through export rules. Export rules allow storage administrators to fence access to volumes or qtrees to specific clients in the environment. For example, if an NFS client is not listed in an export rule that allows access, then that client will not be able to mount the export.

Export rules also provide ways to limit whether a specific client is allowed to read, write, setUID permissions, and access to change file ownerships. Rules can define which specific NFS versions are allowed to mount, as well as what security flavors are allowed (such as AUTH_SYS or Kerberos).

Security limitations of export rules

Export rules are just one way to secure NFS mounts, but by themselves, don't offer enough security to approach enterprise standards. Export rules depend on clientmatch entries to limit access. This means someone could spoof an IP address in the clientmatch list and gain access to the export without any other authentication requirements, unless other security concepts, such as Kerberos authentication, are also required by the export rule.

NFSv3 permissions

NFSv3 offers basic file and folder permissions to control access to end users. These permissions follow the NFS mode bits definitions in RFC 1813 starting on page 22.

Security limitations of NFSv3 permissions

Mode bit permissions cover only owner, group and everyone else in the semantics – meaning that there are no granular user access controls in place for basic NFSv3. ONTAP does not support POSIX ACLs, so granular ACLs are only possible in the following scenarios with NFSv3:

- NTFS security style volumes (CIFS server required) with valid UNIX to Windows user mappings.
- NFSv4.x ACLs applied using an admin client mounting NFSv4.x to apply ACLs.

Additionally, mode bits don't provide the same level of granularity in permissions that NTFS or NFSv4.x ACLs provide. The following table compares the permission granularity between NFSv3 mode bits and NFSv4.x ACLs. For information about NFSv4.x ACLs, see: https://linux.die.net/man/5/nfs4_acl.

- Table 6 compares the NFSv3 mode bits against the NFSv4.x ACL granularity.

Table 6) NFSv3 mode bits versus NFSv4.x ACL granularity.

NFSv3 mode bits	NFSv4.x ACLs
• Set user ID on execution • Set group ID on execution • Save swapped text (not defined in POSIX) • Read permission for owner • Write permission for owner • Execute permission for owner on a file; or look up (search) permission for owner in directory • Read permission for group • Write permission for group • Execute permission for group on a file; or look up (search) permission for group in directory • Read permission for others • Write permission for others • Execute permission for others on a file; or look up (search) permission for others in directory	• ACE types (Allow/Deny/Audit) • Inheritance flags: – directory-inherit – file-inherit – no-propagate-inherit – inherit-only • Permissions: – read-data (files) / list-directory (directories) – write-data (files) / create-file (directories) – append-data (files) / create-subdirectory (directories) – execute (files) / change-directory (directories) – delete – delete-child – read-attributes – write-attributes – read-named-attributes – write-named-attributes – read-ACL – write-ACL – write-owner • Synchronize

Finally, NFS group membership (in both NFSv3 and NFSV4.x) is limited to a maximum of 16 as per the RPC packet limits. Options exist in ONTAP to help extend beyond those limits in “Auxiliary GIDs — addressing the 16 GID limitation for NFS.”

NFSv3 user and group IDs

NFSv3 user and group IDs come across the wire as numeric IDs, rather than names. ONTAP does no name resolution for these numeric IDs in UNIX security style volumes and qtrees with no NFSv4.x ACLs present. With NTFS security styles and NFSv4.x ACLs, ONTAP will attempt to resolve a numeric ID to a valid Windows user to negotiate access. For more information, see “Viewing and managing NFS credentials.”

Security limitations of NFSv3 user and group IDs

The client and server never have to confirm that the user attempting a read or write with a numeric ID is actually a valid user. It’s just implicitly trusted. This opens the filesystem up to potential breaches simply by spoofing a known numeric ID. To prevent this from creating a security hole with NFSv3, implementing Kerberos for NFS forces users to authenticate with a user name and password or keytab file to get a Kerberos ticket to allow access into a mount. Using this in conjunction with NFSv4.x or NTFS ACLs can help mitigate performance risk with NFSv3 numeric IDs.

Advanced NFSv3 security concepts

The following section covers two advanced security concepts for NFSv3. The term Advanced refers to the concepts being either nonstandard or containing multiple steps for configuration.

Using ACLs with NFSv3

As previously mentioned, POSIX ACLs are not supported for use with NFSv3 in ONTAP. However, there are two options for controlling NFSv3 user and group access with ACLs:

- **NTFS ACLs.** ONTAP provides a way to present datasets in the same volume to multiple NAS protocols simultaneously. For example, a volume can host clients through NFS and SMB, while protocol-specific features such as locking and permissions are negotiated by ONTAP to ensure data resiliency and security.

  NFSv3 client access can be managed through file and folder permissions as configured from Windows clients when using ONTAP for both CIFS/SMB and NFS access. When a CIFS/SMB server is created and volume security styles are using NTFS permission semantics, NFSv3 clients will adhere to those NTFS ACLs by way of UNIX to Windows user name mapping that ONTAP requires for NFS access to volumes containing NTFS ACLs. For more information on this functionality, see "Multiprotocol NAS."
- **NFSv4.x ACLs.** NFSv4.x is supported in ONTAP along with NFSv3 and offers granular file and folder permissions in the same vein as NTFS ACLs. ONTAP also allows storage administrators to set NFSv4.x ACLs through an administrator client that is honored by NFSv3 clients. When an NFSv3 client attempts access to a mount with NFSv4.x ACLs, ONTAP forces the numeric user ID to resolve to a valid UNIX user through the NFSv4.x name mapping requirements so the NFSv4.x ACL can be honored properly. For more information about NFSv4.x ACLs, see "NFSv4.x ACLs."

Using NFS Kerberos with NFSv3

Kerberos is possible to use with NFSv3, but there are several caveats:

- NFSv3 is made up of several ancillary protocols in addition to the NFS protocol. When Kerberos is used with NFSv3, only the NFS packets use Kerberos. Mount, portmap, and so on do not use Kerberos.
- Export policy rules with NFSv3 and Kerberos should use both sys and krb5* in their settings to reflect the lack of Kerberos support for ancillary protocols when using ONTAP versions 8.2P5 or earlier. Modern releases do not require this modification to the export policy rules. For more information, see bug 756081.

  Note: For more information about using Kerberos for NFS, see TR-4616: NFS Kerberos in ONTAP.

Blocking portmap with ONTAP firewall policies

In ONTAP 9.3 and earlier, the portmap service (rpcbind) was always accessible on port 111 in network configurations that relied on the built-in ONTAP firewall rather than a third-party firewall, creating a potential security vulnerability. Starting in ONTAP 9.4, you can modify firewall policies to control whether the portmap service is accessible on particular LIFs. The new firewall policy is called mgmt-nfs and has the following rules by default:

```
        mgmt-nfs
                     dns         0.0.0.0/0, ::/0
                     http        0.0.0.0/0, ::/0
                     ndmp        0.0.0.0/0, ::/0
                     ndmps       0.0.0.0/0, ::/0
                     ntp         0.0.0.0/0, ::/0
                     portmap     0.0.0.0/0, ::/0
                     snmp        0.0.0.0/0, ::/0
```

Considerations:

- On upgrade, ONTAP adds the portmap service to all existing firewall policies, default or custom.
- When you create a new cluster or new IPspace, ONTAP adds the portmap service only to the default data policy, not to the default management or intercluster policies.
- You can add the portmap service to default or custom policies as needed, and remove the service as needed.

Inheriting the group owner from the parent

When migrating from some competitor systems, there might be an option in use to inherit a parent directory's group owner when creating new files and directories. ONTAP does not have this particular option, but similar functionality can be achieved using the sticky bit flag for the permissions.

1. To set the sticky bit, run the following commands:

```
# chmod 2775 setguid/
```

2. Change the owner of the parent directory.

```
# chown prof1:ProfGroup setguid/
```

The folder now has an "s" present in the execute portion of the group mode bits.

```
# ls -la | grep setguid
drwxrwsr-x   2 prof1            ProfGroup                  4096 Oct  6 16:10 setguid
```

When a new file is created, it inherits the group from the parent directory. In this case, root is the user.

```
# cd setguid/
# id
uid=0(root) gid=0(root) groups=0(root)
# touch newfile
# ls -la
total 8
drwxrwsr-x  2 prof1 ProfGroup 4096 Oct  6  2020 .
drwxrwxrwx 20 root  root      4096 Oct  6 16:10 ..
-rw-r--r--  1 root  ProfGroup    0 Oct  6  2020 newfile
```

NFSv4.x considerations

The following section covers functionality, known issues, and considerations with NFSv4.x in ONTAP.

Enabling NFSv4.x

To start using NFSv4.x with ONTAP in your environment, there is a list of steps to perform/review:

1. Set the NFS ID domain string in `/etc/idmapd.conf` to the same value as the `-v4-id-domain` option on the NFS SVM.
2. Ensure that the users and groups accessing the NFSv4.x client also exist (or can be queried from) the ONTAP SVM. These users and groups need to have the same names and case sensitivity.
   - For example, *john@DOMAIN.COM* on the NFS client needs to match *john@DOMAIN.COM* on the ONTAP NFS server.
3. If using LDAP or NIS for UNIX identities, ensure the user and group lookups return the expected IDs and group members.
4. Export policy rules for volumes should be set to allow NFSv4 as a protocol.
5. Data LIFs in the environment should have NFS as an allowed protocol (net int show -fields allowed-protocols).
6. The desired NFSv4.x versions should be enabled. If only version 4.1 is desired, enable only that version and disable version 4.0.

7. If NFSv4.x ACLs are desired, you must enable them for the specific NFSv4.x version you wish to use them with. (`-v4.0-acl`, `-v4.1-acl`)
8. Clients will negotiate the highest NFS version the NFS server supports. If some clients require NFSv3, they will need to change how they mount.

Advantages of using NFSv4.x

The following are some advantages to using NFSv4.x in your environment:

- Firewall-friendly because NFSv4 uses only a single port (2049) for its operations
- Advanced and aggressive cache management, such as delegations in NFSv4.x
- Strong RPC security choices that employ cryptography
- Internationalization
- Compound operations
- Works only with TCP
- Stateful protocol (not stateless like NFSv3)
- Kerberos configuration for efficient authentication mechanisms:
  - Support for DES and 3DES for encryption in clustered ONTAP 8.2.x and earlier
  - AES support in 8.3 and later
- Migration (for dNFS) using referrals
- Support of access control that is compatible with UNIX and Windows
- String-based user and group identifiers
- Parallel access to data through pNFS (does not apply for NFSv4.0)

It is important that you treat each specific use case differently. NFSv4.x is not ideal for all workload types. Be sure to test for desired functionality and performance before rolling out NFSv4.x en masse.

Note: ONTAP currently does not support NFSv4.x session trunking.

Performance enhancements for NFSv4.x operations

NetApp is constantly striving to improve performance for each ONTAP release. NFSv4.x performance is a priority, as it is the future for NFS. Performance enhancements are listed below, along with the ONTAP release they were introduced. For the best possible performance with NFS, always run the latest patched ONTAP release available.

NFSv4.x fastpath (introduced in ONTAP 8.2)

Starting in ONTAP 8.2, NFS fastpath was introduced to potentially improve NFSv4 performance for reads and writes. This improvement was made by bypassing the internal processing of NFSv4 packets into ONTAP-centric packets when the data request is made on a LIF that is local to the node hosting the volume. When combined with other features such as pNFS or referrals, localized data can be guaranteed for each read and write request, thus allowing consistent use of the NFSv4 fastpath. NFSv3 has always had an NFS fastpath concept. NFS fastpath is enabled by default.

NFSv4.x multithreaded operations (introduced in ONTAP 8.2)

In ONTAP 8.2 and later, multiprocessor support was added for NFSv4.x read and write operations. Metadata operations, however, still use a single threaded approach. In previous releases, NFSv4.x read and write operations were single threaded, thus allowing a potential bottleneck at the CPU for the protocol domain. Using multiple processors for read and write operations can greatly increase throughput on NetApp systems that contain more than one CPU for NFSv4.x workloads that are read and write heavy.

Note: NFSv3 has always used multiple processors for reads and writes. NFSv3 also uses multiple processors for metadata operations.

NFSv4.x — Performance improvements for streaming workload types (introduced in ONTAP 9.0)

ONTAP 9.0 introduced an improvement to how streaming workload types such as VMware, Oracle, and SAP HANA performed with NFSv4.1 by adding large I/O support. This allowed NFS (both v3 and 4.x) to use up to 1MB for both reads and writes.

NFSv4.x — Performance improvements for metadata workloads (introduced in ONTAP 9.5)

Many improvements were added into ONTAP 9.5 to improve metadata workloads, including:

- NFSv4.0 cache I/O support
- Optimizations of NFSv4.x metadata operations
- Increased caching
- Improved locking performance
- Increased StorePool limits

NFSv4.x FlexGroup volume support (introduced in ONTAP 9.7)

In addition to the metadata performance improvements added in ONTAP 9.5, FlexGroup volume support can help increase metadata workload performance as well, by way of parallelization of file ingest operations. For more information about FlexGroup volumes, see TR-4571: NetApp FlexGroup Volumes Best Practices and Implementation.

NFSv4.x performance enhancements for metadata operations (ONTAP 9.8)

Some enhancements were added to ONTAP 9.8 to improve the overall performance for NFSv4.x when dealing with high metadata workloads, such as what would be seen with standard NAS benchmark tests for software builds. These changes resulted in a marked improvement for these workloads, bringing the peak IOPS achieved at 1ms latency a bit closer to what NFSv3 provides.

These enhancements include:

- Improved write lock usage
- Optimized QoS performance
- Improved handling of replay operations (see bug 1281571)
- Consolidation of compound operations
- Storepool optimizations
- OPEN/CLOSE enhancements
- Nconnect support (for more information, see the section “Nconnect”)
- Parallelization of UNLINK operations
- Parallelization of READDIR operations

NFSv4.x performance optimizations in ONTAP 9.9.1

Additional optimizations were added to ONTAP 9.9.1 to improve the overall performance for NFSv4.x when dealing with high metadata workloads, such as what would be seen with standard NAS benchmark tests for software builds. These changes resulted in a marked improvement over ONTAP 9.8 for these workloads, bringing the peak IOPS achieved at 1ms latency within ~22% of what NFSv3 offers:

- Prefetch of attributes during LOOKUP and CLOSE operations
- Prefetch of ACCESS calls during OPEN to reduce metadata operations
- GETATTR after WRITE optimizations

- Memory pool to store prefetched attributes
- FlexGroup optimizations (cache FlexGroup volume details to optimize LOOKUP)
- Path length reduction for export checks

NFSv4.0

The NetApp ONTAP NFSv4.x implementation provides the following capabilities:

- **Write order.** The implementation provides the capability to write data blocks to shared storage in the same order as they occur in the data buffer.
- **Synchronous write persistence.** Upon return from a synchronous write call, ONTAP (clustered and 7-Mode) guarantees that all the data has been written to durable, persistent storage.
- **Distributed file locking.** The implementation provides the capability to request and obtain an exclusive lock on the shared storage, without assigning the locks to two servers simultaneously.
- **Unique write ownership.** ONTAP (clustered and 7-Mode) guarantees that the file lock is the only server process that can write to the file. After ONTAP transfers the lock to another server, pending writes queued by the previous owner fail.

Transitioning from NFSv3 to NFSv4.x: Considerations

The following section covers some considerations that need to be addressed when migrating from NFSv3 to NFSv4.x. When choosing to use NFSv4.x after using NFSv3, you cannot simply turn it on and have it work as expected. There are specific items to address, such as:

- Domain strings/ID mapping
- Storage failover considerations
- Name services
- Firewall considerations
- Export policy rule considerations
- Client support
- NFSv4.x features and functionality

For an in-depth look at the NFSv4.x protocol, including information about NFSv4.2, see the SNIA overview of NFSv4.

NFSv4.x ID domain mapping

While customers prepare to migrate their existing setup and infrastructure from NFSv3 to NFSv4, some environmental changes must be made before moving to NFSv4. One of them is ID domain mapping.

When NFSv3 clients access a mount, the numeric ID is passed to the NFS server and no further ID lookups are needed, provided all portions of the access are using UNIX security with NFSv3 semantics.

When NFSv4.x clients access a mount, a name string is passed from the client to the server that contains the user or group principal (name@DOMAIN.COM).

The server then will attempt to verify that it knows about a principal with the same exact name (case sensitive) through its name service and NFS server configuration. If that name string does not exist on both client and server, then the user is squashed to a nobody user as defined in the client NFSv4.x configuration file. This provides an extra layer of security over NFSv3.

Bypassing the name string — Numeric IDs

In some cases, storage administrators might not want to use NFSv4.x for its security, but instead for its locking mechanisms. Or they might not want to deal with the configuration overhead of client/server name strings.

In those cases, there is an option called `v4-numeric-ids`. With this option enabled, if the client does not have access to the name mappings, numeric IDs can be sent in the user name and group name fields. The server accepts them and treats them as representing the same user as would be represented by a v2/v3 UID or GID having the corresponding numeric value. If the client does have a name string that matches, then the client uses the name string rather than the numeriCID. If the client and server have matching user names but mismatched domain strings, then numerics won't be used; instead, the user name/group name reverts to `nobody`. This is a common scenario with the root user, because that user always exists on client and server, while the ID strings for NFSv4 in ONTAP defaults to `defaultv4iddomain.com`. Because the NFS clients default to no domain string setting in the `idmapd.conf` file (and instead falls back to the DNS domain for NFSv4 domain), mismatches often occur in that scenario.

Essentially, this option makes NFSv4.x behave more like NFSv3. The default value of this option is Enabled. For considerations regarding the use of extended group support, see "Considerations for numeric ID authentication (NFSv3 and NFSv4.x)."

Storage failover considerations

NFSv4.x uses a completely different locking model than NFSv3. Locking in NFSv4.x is a lease-based model that is integrated into the protocol rather than separated as it is in NFSv3 (NLM).

From the ONTAP documentation:

```
In accordance with RFC 3530, ONTAP "defines a single lease period for all state held by an NFS
client. If the client does not renew its lease within the defined period, all states associated
with the client's lease may be released by the server." The client can renew its lease explicitly
or implicitly by performing an operation, such as reading a file.
Furthermore, ONTAP defines a grace period, which is a period of special processing in which
clients attempt to reclaim their locking state during a server recovery.
```

Table 7) NFSv4.x lock terminology.

Term	Definition (per RFC 3530)
Lease	The time period in which ONTAP irrevocably grants a lock to a client
Grace period	The time period in which clients attempt to reclaim their locking state from ONTAP during server recovery
Lock	Refers to both record (byte-range) locks as well as file (share) locks unless specifically stated otherwise

For more information about NFSv4.x locking, see the section in this document on "NFSv4 locking." Because of this new locking methodology, as well as the statefulness of the NFSv4.x protocol, storage failover operates differently as compared to NFSv3. For more information, see the section in this document called "Nondisruptive operations with NFS."

Name services

When deciding to use NFSv4.x, it is a NetApp best practice to centralize the NFSv4.x users in name services such as LDAP or NIS. Doing so allows all clients and ONTAP NFS servers to leverage the same resources and guarantees that all names, UIDs, and GIDs are consistent across the implementation. For more information about name services, see TR-4835: How to Configure LDAP in ONTAP and TR-4668: Name Services Best Practices.

Firewall considerations

NFSv3 required several ports to be opened for ancillary protocols such as NLM, NSM, and so on in addition to port 2049. NFSv4.x requires only port 2049. If you want to use NFSv3 and NFSv4.x in the same environment, open all relevant NFS ports. These ports are referenced in “Default NFS ports in ONTAP.” For more information and guidance on firewalls, see “NFS security best practices.”

Volume language considerations

In ONTAP, volumes can have specific languages set. This capability is intended to be used for internationalization of file names for languages that use characters not common to English, such as Japanese, Chinese, German, and so on. When using NFSv4.x, RFC 3530 states that UTF-8 is recommended.

```
11.  Internationalization

   The primary issue in which NFS version 4 needs to deal with
   internationalization, or I18N, is with respect to file names and
   other strings as used within the protocol.  The choice of string
   representation must allow reasonable name/string access to clients
   which use various languages.  The UTF-8 encoding of the UCS as
   defined by [ISO10646] allows for this type of access and follows the
   policy described in "IETF Policy on Character Sets and Languages",
   [RFC2277].
```

When changing a volume's language, every file in the volume must be accessed after the change to make sure that they all reflect the language change. Use a simple `ls -lR` to access a recursive listing of files. If the environment is a high file count environment, consider using XCP to scan the files quickly.

Export policy rules

If an environment was configured for NFSv3 and the export policy rule option `-protocol` was limited to allow NFSv3 only, then the option must be modified to allow NFSv4. Additionally, policy rules could be configured to allow access only to NFSv4.x clients.

For example:

```
cluster::> export-policy rule modify -policy default -vserver NAS -protocol nfs4
```

For more information, consult the product documentation for your specific version of ONTAP.

Client considerations

When you use NFSv4.x, clients are as important to consider as the NFS server. For specific questions about the NFSv4.x configuration, contact the operating system vendor.

Follow these client considerations when implementing NFSv4.x:

Note: Other considerations might be necessary.

- NFSv4.x is supported.
- The fstab file and NFS configuration files are configured properly. When mounting, the client negotiates the highest NFS version available with the NFS server. If NFSv4.x is not allowed by the client or fstab specifies NFSv3, then NFSv4.x is not used at mount.
- The idmapd.conf file is configured with the proper settings, including the correct NFSv4.x ID domain.
- The client either contains identical users/groups and UID/GID (including case sensitivity) in local passwd and group files or uses the same name service server as the NFS server/ONTAP SVM.
- If using name services on the client, ensure the client is configured properly for name services (nsswitch.conf, ldap.conf, sssd.conf, and so on) and the appropriate services are started, running, and configured to start at boot.

- The NFSv4.x service is started, running, and configured to start at boot.

Making NFSv4.x act like NFSv3

In some cases, choosing NFSv4.x is not due to any security concern, but rather because an application vendor or service provider requires its use. Without an infrastructure to provide valid NFSv4.x ID domain information and to map users properly, things can become complicated. In scenarios where you want NFSv4.x to act like NFSv3, you can do the following:

- Ensure that `-v4-numeric-ids` is enabled on the NFS SVM (default is enabled)
- Ensure that the NFSv4.x ID domains match on the SVM (`-v4-id-domain`) and in the client's `idmapd.conf` file (or is the same as the DNS domain name of the client)
- Set `/sys/module/nfsd/parameters/nfs4_disable_idmapping` (or its equivalent) to Y on the NFS client

When these steps are followed, the clients bypass ID mapping and instead fall back to numeric IDs when a user name is not present on both the server and client.

NFSv4 features and functionality

NFSv4.x is the next evolution of the NFS protocol and enhances NFSv3 with new features and functionality, such as referrals, delegations, pNFS, and so on. These features are covered throughout this document and should be factored into any design decisions for NFSv4.x implementations.

NFSv4 user ID mapping

As previously mentioned in this document (in the "NFSv4.x ID domain mapping" section), NFSv4.x clients and servers attempt to map the user ID domain strings for added security. If numeric IDs are preferred, ONTAP has an NFS option (`-v4-numeric-ids`) to avoid requirements for name strings.

NFSv4.x ACLs

The NFSv4.x protocol can provide access control in the form of ACLs, which are similar in concept to those found in CIFS. An NFSv4 ACL consists of individual Access Control Entries (ACEs), each of which provides an access control directive to the server. ONTAP defaults to 400 ACEs and supports a maximum of 1,024 ACEs with a configurable NFS option (`-v4-acl-max-aces`).

Benefits of Enabling NFSv4 ACLs

The benefits of enabling NFSv4 ACLs include the following:

- Granular control of user access to files and directories
- Better NFS security
- Improved interoperability with CIFS
- Removal of the NFS limitation of 16 groups per user with AUTH_SYS security
  - ACLs bypass the need for GID resolution, which effectively removes the GID limit

Compatibility between NFSv4 ACLs and SMB clients

NFSv4 ACLs are different from Windows file-level ACLs (NTFS ACLs) but carry similar functionality. However, in multiprotocol NAS environments, if NFSv4.x ACLs are present, clients using SMB2.0 and later won't be able to view ACLs from Windows security tabs – even if the NFS option `ntacl-display-permissive-perms` and .the CIFS/SMB option `is-unix-nt-acl-enabled` are set. See bug 928026 for details.

How NFSv4 ACLs work

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, constructs a list of ACEs, and returns the list 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.

Access is denied if a DENY ACE is present in the ACL; 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.

If a basic chmod is run on a file or folder with NFSv4.x ACLs set, the ACLs will be removed unless the NFS option `v4-acl-preserve` is enabled.

A client using NFSv4 ACLs can set and view ACLs for files and directories on the system. When a new file or subdirectory is created in a directory that has an ACL, the new file or subdirectory inherits all ACEs in the ACL that have been tagged with the appropriate inheritance flags. For access checking, CIFS users are mapped to UNIX users. The mapped UNIX user and that user's group membership are checked against the ACL.

If a file or directory has an ACL, that ACL is used to control access no matter which protocol—NFSv3, NFSv4, or CIFS—is used to access the file or directory. The ACL is also used even if NFSv4 is no longer enabled on the system.

Files and directories inherit ACEs from NFSv4 ACLs on parent directories (possibly with appropriate modifications) as long as the ACEs have been tagged with the correct inheritance flags.

When a file or directory is created as the result of an NFSv4 request, the ACL on the resulting file or directory depends on whether the file creation request includes an ACL or only standard UNIX file access permissions. The ACL also depends on whether the parent directory has an ACL.

- If the request includes an ACL, that ACL is used.
- If the request includes only standard UNIX file access permissions and the parent directory does not have an ACL, the client file mode is used to set standard UNIX file access permissions.
- If the request includes only standard UNIX file access permissions and the parent directory has a noninheritable ACL, a default ACL based on the mode bits passed into the request is set on the new object.
- If the request includes only standard UNIX file access permissions but the parent directory has an ACL, the ACEs in the parent directory's ACL are inherited by the new file or directory as long as the ACEs have been tagged with the appropriate inheritance flags.

  Note: A parent ACL is inherited even if `-v4.0-acl` is set to `off`.

NFSv4 ACL behavior with umask and ACL inheritance

NFSv4 ACLs provide the ability to offer ACL inheritance. ACL inheritance means that files or folders created beneath objects with NFSv4 ACLs set can inherit the ACLs based on the configuration of the ACL inheritance flag.

Umask is used to control the permission level at which files and folders are created in a directory. For more information, see the section called "Umask."

By default, ONTAP allows umask to override inherited ACLs, which is expected behavior as per RFC 5661. To adjust the behavior ACL inheritance with umask, you can enable the option `-v4-inherited-acl-preserve`.

ACL formatting

NFSv4.x ACLs have specific formatting. The following example is an ACE set on a file:

```
A::ldapuser@domain.netapp.com:rwatTnNcCy
```

The preceding example follows the ACL format guidelines of:

```
type:flags:principal:permissions
```

A type of A means allow. The flags are not set in this case, because the principal is not a group and does not include inheritance. Also, because the ACE is not an AUDIT entry, there is no need to set the audit flags. For more information about NFSv4.x ACLs, see http://linux.die.net/man/5/nfs4_acl.

If an NFSv4.x ACL is not set properly, the ACL might not behave as expected, or the ACL change may fail to apply and throw an error.

Sample errors include:

```
Failed setxattr operation: Invalid argument
Scanning ACE string 'A::user@rwaDxtTnNcCy' failed.
```

Explicit DENY

NFSv4 permissions can 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. Explicit DENY attributes will override any ACCESS ACEs, explicit or not.

DENY ACEs are set with an attribute tag of D.

For example:

```
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 because they can be confusing and complicated. When DENY ACEs are set, users might be denied access when they expect to be granted access. This is because the ordering of NFSv4 ACLs affects how they are evaluated.

The preceding set of ACEs is equivalent to 755 in mode bits, which means:

- The owner has full rights.
- Groups have read only.
- Others have read only.

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

For an example of explicit DENY, see “NFSv4.x ACL Explicit DENY example.”

NFSv4 ACL preservation

By default, NFSv4 ACLs can be affected by setting mode bits on a file or folder. If an NFSv4 ACE has been configured and a `chmod` is used, the ACE is removed. This is controlled through the `v4-acl-preserve` option. For an example, see “NFSv4.x ACL preservation example” in this document.

Note: By default, this option is set to Enabled. NetApp recommends not modifying this option unless a specific use case arises.

Mode bit display behavior from Windows created files with NFSv4 ACLs

In multiprotocol NAS environments (where both SMB and NFS are used to access the same data), NFSv4 ACLs can create some undesirable behavior. In one such case, ONTAP 9.7 and earlier versions would show “----------” for files created via SMB on NFS clients when NFSv4 ACLs are in use. ONTAP 9.8 and later introduces the `is-inherit-modebits-with-nfsv4acl-enabled` (defaults to disabled) CIFS server option to resolve this issue. Set that option to enabled for proper mode bit display.

For more information, see bug 820848.

Preventing chmod on files and folders with NFSv4 ACLs present

In some cases, you might want to prevent `chmod` commands from working on any files or folders that have NFSv4 ACLs present – even from the file owners. This is controlled with the `-restrict-chmod-acl` option in ONTAP 9.8 (diag privilege). The default is unrestricted; to prevent `chmods` from being allowed on these files and folders, set the option to Restricted.

NFSv4 delegations

NFSv4 introduces the concept of delegations that provide an aggressive local client cache, which is different from the ad hoc caching that NFSv3 provides. There are two forms of delegations: read and write. Delegations value cache correctness over improving performance. For delegations to work, a supported UNIX client is required along with the NFS 4.x version-specific delegation options enabled on the NetApp controller. These options are disabled by default.

When a server determines to delegate a complete file or part of the file to the client, the client caches it locally and avoids additional RPC calls to the server. This reduces GETATTR calls in the case of read delegations because there are fewer requests to the server to obtain the file’s information. However, delegations do not cache metadata, which means that high file count workloads will not see as much of a benefit with delegations as a streaming file workload might see.

Read delegations can be granted to numerous clients but write delegations can be granted only to one client at a time, as any new write to a file invalidates the delegation. The server reserves the right to recall the delegation for any valid reason. The server determines to delegate the file under two scenarios: a confirmed call-back path from the client that the server uses to recall a delegation if needed and when the client sends an OPEN function for a file.

Why use read or write delegations?

Delegations can be used to improve the read and write performance of certain applications. For example, web applications that have numerous readers of one or more files on the same client and across clients that also generate copious amounts of metadata operations such as GETATTRs and LOOKUPs could request read delegations from the storage system for local access to improve performance and response time. Delegating the whole file or certain ranges of bytes to the client's local memory avoids additional RPC calls over the wire for metadata operations.

If the file or byte offset is rewritten by any client during the delegation, the delegation is recalled. Although this process is necessary to acquire updates, the delegation recall can affect read performance. Therefore, write delegations should be used for single writer applications. Read and write delegations can improve I/O performance, but that depends on the client hardware and operating system. For instance, low-memory client platforms do not handle delegations very well.

How can I tell if I'm using delegations?

If you enable delegations and the client supports them, they will be used. You can verify that delegations are being used by checking `vserver locks show -type delegation`.

```
cluster::*> vserver locks show -type delegation

Vserver: DEMO
Volume   Object Path               LIF         Protocol  Lock Type   Client
-------- ------------------------- ----------- --------- ----------- ----------
flexgroup_16
         /flexgroup_16/files/topdir_82/subdir_268/file3
                                   data        nfsv4.1   delegation  -
                Delegation Type: write
         /flexgroup_16/files/topdir_27/subdir_249/file2
                                   data        nfsv4.1   delegation  -
                Delegation Type: write
```

You can also review delegations with performance statistics.

```
cluster::*> statistics show -object nfsv4_1 -counter *del*

Object: nfsv4_1
Instance: DEMO
Start-time: 4/9/2020 15:05:44
End-time: 4/9/2020 15:31:13
Elapsed-time: 1529s
Scope: cluster
Number of Constituents: 2 (complete_aggregation)
    Counter                                                     Value
    -------------------------------- --------------------------------
    delegreturn_avg_latency                                  1366us
    delegreturn_percent                                          4%
    delegreturn_success                                      311465
    delegreturn_total                                        311465
```

NFSv4 locking

For NFSv4 clients, ONTAP supports the NFSv4 file-locking mechanism, maintaining the state of all file locks under a lease-based model. In accordance with RFC 3530, ONTAP "defines a single lease period for all state held by an NFS client. If the client does not renew its lease within the defined period, all state associated with the client's lease may be released by the server." The client can renew its lease explicitly or implicitly by performing an operation, such as reading a file. Furthermore, ONTAP defines a grace period, which is a period of special processing in which clients attempt to reclaim their locking state during a server recovery.

Locks are issued by ONTAP to the clients on a lease basis. The server checks the lease on each client by default every 30 seconds. In the case of a client reboot, the client can reclaim all the valid locks from the server after it has restarted. If a server reboots, then upon restarting it does not issue any new locks to the clients for a default grace period of 45 seconds (tunable in ONTAP to a maximum of 90 seconds). After that time the locks are issued to the requesting clients. The lease time of 30 seconds can be tuned based on the application requirements. For information about managing NFS locks, see Managing file locks in the product documentation.

Table 8) NFS lease and grace periods.

Term	Definition*
Lease	The time period in which ONTAP irrevocably grants a lock to a client
Grace period	The time period in which clients attempt to reclaim their locking state from ONTAP during server recovery

*For more information, see RFC 3530.

Specifying the NFSv4 locking lease period

To specify the NFSv4 locking lease period (the time period in which ONTAP irrevocably grants a lock to a client), you can modify the `-v4-lease-seconds` option. By default, this option is set to 30. The minimum value for this option is 10. The maximum value for this option is the locking grace period, which you can set with the `locking.lease_seconds` option.

NFSv4.x referrals

An NFS referral directs a client to another LIF in the SVM upon initial NFS mount request. The NFSv4.x client uses this referral to direct its access over the referred path to the target LIF from that point forward. Referrals are issued when there is a LIF in the SVM that resides on the node where the data volume resides. In other words, if a cluster node receives an NFSv4.x request for a nonlocal volume, the cluster node is able to refer the client to the local path for that volume by means of the LIF. Doing so allows clients faster access to the data using a direct path and avoids extra traffic on the cluster network.

How they work

When a mount request is sent, the request acts as a normal NFSv4.x mount operation. However, after the DH LOOKUP call is made, the server (NetApp cluster) responds with the GETFH status of `NFS4ERR_MOVED` to notify the client that the volume being accessed does not live where the LIF being requested lives. The server then sends a LOOKUP call to the client, notifying it of the IP (using the `fs_location4 value`) on the node where the data volume lives. This process works regardless of whether a client is mounting using a DNS name or IP. However, the client reports that it is mounted to the IP specified rather than the IP returned to the client from the server.

If a volume moves to another aggregate on another node, the NFSv4.x clients must unmount and remount the file system manually if volume locality is desired. Remounting makes sure that the client is referred to the new location of the volume. If the manual mount/unmount process is not followed, the client can still access the volume in its new location, but I/O requests would then take a remote path. However, remote I/O requests might not be impactful enough to an environment to matter enough to remount clients, which is a disruptive and potentially involved operation. The decision to remount clients should be made on a case-by-case basis.

Note: NFSv4.x referrals were introduced in RHEL as early as 5.1 (2.6.18-53), but NetApp recommends using no kernel older than 2.6.25 with NFS referrals and no version earlier than 1.0.12 of nfs-utils.

If a volume is junctioned below other volumes, the referral uses the volume being mounted to refer to as the local volume. For example:

- A client wants to mount vol2
- Vol2's junction is /vol1/vol2
- Vol1 lives on node1; vol2 lives on node2
- A mount is made to cluster:/vol1/vol2
- The referral returns the IP address of a LIF that lives on node2, regardless of what IP address is returned from DNS for the host name cluster
- The mount uses the LIF local to vol2 on node2

In a mixed client environment, if any of the clients do not support referrals, then the `-v4.0-referrals` option should not be enabled. If the option is enabled and a client that does not support referrals gets a referral from the server, that client is unable to access the volume and experiences failures. For more information about referrals, see RFC 3530.

NFSv4.x stateless migration – Oracle dNFS

NFSv4 referrals also brought NFSv4 stateless migration support in ONTAP 8.1 and later and includes support only for Oracle dNFS.

Migration is an NFSv4.x feature that allows a file system to move from one server to another without client disruption. Migration enablement requires enabling referrals and the option `-v4-fsid-change` on the NFS server. Migration is a diag-level option. Enabling migration assumes the following about the solution:

- All clients accessing the NFSv4.x server on the SVM are stateless.
- All clients accessing the NFSv4.x server on the SVM support migrations.
- The NFSv4.x clients do not use the following:
  - Locking
  - Share reservations
  - Delegations
  - OPEN for file access
- The NFSv4.x clients do use the following:
  - READ, WRITE, and SETATTR with special stateid of all bits 0
  - OPEN only to create a file and close it right away
- The NFSv4.x clients do not have a state established on the NFS server.

NFS migration support can be useful in the following scenarios in ONTAP:

- Volume moves
- LIF migration/failover

Table 9) Referrals versus migration versus pNFS.

	Referrals	Stateless migration	pNFS
When does redirect take place?	At mount	Any operation (I/O and metadata)	I/O only (read, write)
Traffic that is redirected	All traffic	All traffic	I/O only (read, write)
Use case	Automounter	Oracle dNFS	Guaranteed data locality for I/O
Drawback	Only on mount	Only stateless operations (no lock state)	Non-I/O traffic is not redirected

Snapshot copies with NFSv4.x

In NFSv3, the `.snapshot` directory is visible to clients by default. (For information about hiding the `.snapshot` directory in NFSv3, see “Hiding Snapshot copies.”) Because NFSv4.x does not use the MOUNT protocol, the `.snapshot` directory is not visible. However, it is accessible from anywhere in the NFSv4.x mount.

To access Snapshot copies using NFSv4.x, simply navigate to the `.snapshot` directory manually.

```
# ls -la /nfs3 | grep snapshot
drwxrwxrwx  16 root                 root                   4096 Mar 31 14:05 .snapshot

# ls -la /nfs4 | grep snapshot
#
# ls -la /nfs4/.snapshot
total 64
drwxrwxrwx 16 root root   4096 Mar 31 14:05 .
drwxr-xr-x 14 root root   4096 Apr 11  2018 ..
drwx--x--x  8 root daemon 4096 Jan 19  2017 base
drwxrwxrwx 11 root root   4096 Jul 10  2017 clone_home_clone.0
drwxr-xr-x 14 root root   4096 Apr 11  2018 daily.2020-03-30_0010
drwxr-xr-x 14 root root   4096 Apr 11  2018 daily.2020-03-31_0010
drwxr-xr-x 14 root root   4096 Apr 11  2018 hourly.2020-03-31_0905
drwxr-xr-x 14 root root   4096 Apr 11  2018 hourly.2020-03-31_1005
```

NFSv4.x behavior with load-sharing mirrors

Load-sharing mirrors in ONTAP are read-only copies of volumes that are replicated through SnapMirror (with the load sharing type specified) to multiple nodes in a cluster to provide resiliency for volumes. Load sharing mirrors are not supported for use with data volumes, but instead are used to protect the SVM root volume, which contains the root of the namespace. This is covered in more detail in “Protecting your namespace.”

However, NFSv4.x operations do not leverage load-sharing mirror destinations when mounting volumes and instead will use the source volume file handle. As a result, load-sharing mirrors don’t provide protection against SVM root volume failures.

NFSv4.1

NFSv4.1 is considered a minor version update of NFSv4. This section covers NFSv4.1 specifics. The previous section covered NFSv4.0, as well as topics that apply to both NFSv4.0 and NFSv4.1 (denoted by NFSv4.x in this document).

It is possible to enable NFSv4.1 and disable NFSv4.0. This is recommended if you wish to prevent clients from using NFSv4.0 for any reason.

To mount a client using NFSv4.1, there must be client support for NFSv4.1. Check with the client vendor for support for NFSv4.1. Mounting NFSv4.1 is generally done with the `minorversion` mount option, but newer Linux kernels will autonegotiate the highest supported NFS version.

Example:

```
# mount -o nfsvers=4,minorversion=1 NFSSERVER:/unix /unix
```

NFSv4.1 features

NFSv4.1 introduced a number of new features to the NFSv4 protocol standard, as covered in RFC-5661. These differences are covered in Section 1.8 of the RFC.

Some features are listed as Required, which means that the feature must be implemented in/supported by the NFS server to claim RFC standard. Other features are listed as Recommended or Optional features and are supported ad hoc by the NFS server but are not required to claim RFC compliance. For example, pNFS is listed as an Optional feature for NFSv4.1 and is supported by ONTAP, but NFS session trunking and directory delegations (also Optionals features) are not currently supported by ONTAP.

Parallel network file system

Parallel NFS (pNFS) is a part of NFS version 4.1 standards. With traditional NFS versions 3, 4, and 4.1, the metadata and data shared the same I/O path. pNFS handles metadata and data on different I/O paths. A metadata server (MDS) handles all the metadata activities from the client, while the data servers provide a direct path for data access.

As explained in RFC 5661:

> “Parallel data access is controlled by recallable objects known as ‘layouts,’ which are integrated into the protocol locking model. Clients direct requests for data access to a set of data servers specified by the layout using a data storage protocol which may be NFSv4.1 or may be another protocol.”

For pNFS, NetApp supports all clients that support pNFS and follow the RFC specifications. By default, the pNFS option is enabled, but it is only active if NFSv4.1 support also is enabled.

How pNFS works

pNFS defines the notion of a device that is generated by the server (that is, an NFS server running on ONTAP) and sent to the client. This process helps the client locate the data and send requests directly over the path local to that data. ONTAP generates one pNFS device per flexible volume. The metadata path does not change, so metadata requests might still be remote. In a ONTAP pNFS implementation, every data LIF is considered an NFS server, so pNFS only works if each node owns at least one data LIF per NFS SVM. Doing otherwise negates the benefits of pNFS, which is data locality regardless of which IP address a client connects to. For information on the behavior of pNFS when a data LIF goes down, see "pNFS and LIF outages."

The pNFS device contains information about the following:

- Volume constituents
- Network location of the constituents

The device information is cached to the local node for improved performance.

To see pNFS devices in the cluster, run the following command in advanced privilege:

```
cluster::> set diag
cluster::*> vserver nfs pnfs devices cache show
```

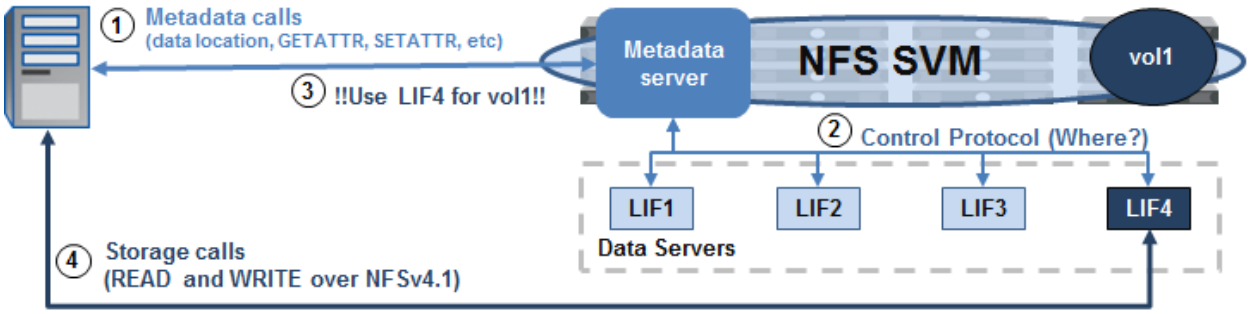
pNFS components

There are three main components of pNFS:

- MDS:
  - Handles all nondata traffic such as GETATTR, SETATTR, and so on
  - Responsible for maintaining metadata that informs the clients of the file locations
  - Located on the NetApp NFS server
- Data server:
  - Stores file data and responds to read and write requests
  - Located on the NetApp NFS server
  - Inode information also resides here
- Clients

These components leverage three different protocols. The control protocol is the way the metadata and data servers stay in sync. The pNFS protocol is used between clients and the MDS. pNFS supports file, block, and object-based storage protocols, but NetApp currently only supports file-based pNFS.

Figure 8) pNFS data workflow.



1. The client makes a data request to the cluster.
2. The metadata server works to find the location of the data if the location is not already cached.
3. The location of the data is returned to the client via the control path.
4. The client begins operations over the specified data LIF returned from the metadata server.

How can I tell pNFS is being used?

To verify whether pNFS is in use, you can run statistics counters to check for `pnfs_layout_conversions` counters. If the number of `pnfs_layout_conversions` are incrementing, then pNFS is in use.

```
cluster::*> statistics show -object nfsv4_1_diag -counter pnfs_layout_conversions

Object: nfsv4_1_diag
Instance: nfs4_1_diag
Start-time: 4/9/2020 16:29:50
End-time: 4/9/2020 16:31:03
Elapsed-time: 73s
Scope: node1

    Counter                                                     Value
    -------------------------------- --------------------------------
    pnfs_layout_conversions                                      4053
```

For basic comparison testing of pNFS with NFSv3 and NFSv4.x, see the following sections in this document:

- “Performance comparison: NFSv3 and NFSv4 using nconnect and pNFS”
- “NFSv3 vs. NFSv4.x — Performance comparisons”
- “Performance examples for different TCP maximum transfer window sizes”

Note: When using pNFS, be sure to use the latest available client and ONTAP release to avoid bugs, such as this one: SU323: Data corruption possible during pNFS I/O on Linux distributions.

How to blacklist clients

When you enable pNFS on the NFS server, any client that mounts using NFSv4.1 and later will leverage pNFS. In some cases, you might only want those clients to use basic NFSv4.x and not pNFS.

You can see if pNFS is in use on the client when an NFSv4.1 mount is established with the following command. You will see `nfs_layout_nfsv41_files` listed if pNFS is in use.

```
# lsmod | grep nfs
nfs_layout_nfsv41_files    32768  1
```

```
nfsv4                  790528  3 nfs_layout_nfsv41_files
dns_resolver            16384  2 cifs,nfsv4
nfsv3                   49152  1
nfs_acl                 16384  1 nfsv3
nfs                    360448  5 nfsv4,nfs_layout_nfsv41_files,nfsv3
lockd                  122880  2 nfsv3,nfs
fscache                385024  2 nfsv4,nfs
sunrpc                 479232  28
nfsv4,auth_rpcgss,lockd,nfs_layout_nfsv41_files,nfsv3,rpcsec_gss_krb5,nfs_acl,nfs
```

You can blacklist specific clients from using pNFS by blacklisting the pNFS module `nfs_layout_nfsv41_files` from loading on the client in the `/etc/modprobe.d/nfs_layout_nfsv41_files-blacklist.conf` file.

For example:

```
cat /etc/modprobe.d/nfs_layout_nfsv41_files-blacklist.conf
blacklist nfs_layout_nfsv41_files
```

Note: A reboot is required for the change to take effect.

To view the modules and verify the module is disabled/not loading when an NFSv4.1 mount is established:

```
# lsmod | grep nfs
nfsv4                  584056  3
dns_resolver            13140  1 nfsv4
nfs                    262045  16 nfsv4
fscache                 64980  2 nfs,nfsv4
nfsd                   351321  13
auth_rpcgss             59415  2 nfsd,rpcsec_gss_krb5
nfs_acl                 12837  1 nfsd
lockd                   98048  2 nfs,nfsd
grace                   13515  2 nfsd,lockd
sunrpc                 358543  46 nfs,nfsd,rpcsec_gss_krb5,auth_rpcgss,lockd,nfsv4,nfs_acl
```

For more information about blacklisting modules, see How do I prevent a kernel module from loading automatically?

NFSv4.1 delegations

NFSv4.1 delegations are very similar to NFSv4.0 delegations but are part of the v4.1 protocol rather than v4.0. Table 10 covers the new additions to NFSv4.1 and how they benefit an environment over NFSv4.0. These additions are covered in detail in RFC 5661, Section 10.2.

Table 10) NFSv4.1 delegation benefits.

NFSv4.1 delegation feature	Benefit versus NFSv4.0 delegation
EXCHANGE_ID is used	In NFSv4.0, SETCLIENTID was used. EXCHANGE_ID replaces SETCLIENTID and enables a client ID to be assigned before any other client operations take place. As per RFC 5661, "The only NFSv4.1 operations possible before a client ID is established are those needed to establish the client ID."
Callbacks use the same TCP connection as the forechannel	In NFSv4.0, callbacks use different TCP connections than the forechannel. Using the same TCP connection for callbacks provides better performance for delegations and is more firewall friendly.
New OPEN request options: • OPEN4_SHARE_ACCESS_WANT_DELEG_MASK	NFSv4.1 provides more precise control to clients for acquisition of delegations than NFSv4.0.

NFSv4.1 delegation feature	Benefit versus NFSv4.0 delegation
• OPEN4_SHARE_ACCESS_WANT_NO_PREFERENCE • OPEN4_SHARE_ACCESS_WANT_READ_DELEG • OPEN4_SHARE_ACCESS_WANT_WRITE_DELEG • OPEN4_SHARE_ACCESS_WANT_ANY_DELEG • OPEN4_SHARE_ACCESS_WANT_NO_DELEG	These new options enable more OPEN scenarios to be covered to prevent problems issuing or reclaiming delegations.

NFSv4.1 sessions

As per RFC 5661:

> A session is a dynamically created, long-lived server object created by a client and used over time from one or more transport connections. Its function is to maintain the server's state relative to the connection(s) belonging to a client instance. This state is entirely independent of the connection itself, and indeed the state exists whether or not the connection exists. A client may have one or more sessions associated with it so that client-associated state may be accessed using any of the sessions associated with that client's client ID, when connections are associated with those sessions. When no connections are associated with any of a client ID's sessions for an extended time, such objects as locks, opens, delegations, layouts, and so on, are subject to expiration. The session serves as an object representing a means of access by a client to the associated client state on the server, independent of the physical means of access to that state.
>
> A single client may create multiple sessions. A single session MUST NOT serve multiple clients.

From SNIA:

> Sessions NFSv4.1 has brought two major pieces of functionality: sessions and pNFS. Sessions bring the advantages of correctness and simplicity to NFS semantics. In order to improve the correctness of NFSv4, NFSv4.1 sessions introduce “exactly-once” semantics. This is important for supporting operations that were non-idempotent (that is, operations that if executed twice or more return different results, for example the file RENAME operation). Making such operations idempotent is a significant practical problem when the file system and the storage are separated by a potentially unreliable communications link, as is the case with NFS. Servers maintain one or more session states in agreement with the client; a session maintains the server's state relative to the connections belonging to a client. Clients can be assured that their requests to the server have been executed, and that they will never be executed more than once. Sessions extend the idea of NFSv4 delegations, which introduced server-initiated asynchronous callbacks; clients can initiate session requests for connections to the server. For WAN based systems, this simplifies operations through firewalls.

NFSv4.1 session trunking

ONTAP’s NFSv4.1 server currently does not offer support for the session trunking (or multipathing) feature. This feature is commonly asked for in VMware environments.

NFSv4.2

NFSv4.2 is the latest NFSv4.x version available and is covered in RFC-7862. ONTAP 9.8 introduced basic support for the NFSv4.2 protocol. ONTAP 9.9 added support for labeled NFS, but no other ancillary features are currently supported. NFSv4.2 does not have an independent option to enable/disable it, but it is enabled/disabled when you enable NFSv4.1 via the `-v4.1` NFS server option in ONTAP. If a client supports NFSv4.2, it negotiates the highest supported version of NFS during the mount command if not specified. Otherwise, use the `minorversion=2` mount option. There is no difference in performance between NFSv4.1 and NFSv4.2.

Labeled NFS 4.2

ONTAP 9.9.1 introduces support for the NFSv4.2 feature called labeled NFS, which is a way to manage granular file and folder access by using SELinux labels and Mandatory Access Control (MAC). These MAC labels are stored with files and folders, and they work in conjunction with UNIX permissions and NFSv4.x ACLs. You can enable and disable this feature with the following advanced privilege option:

```
[-v4.2-seclabel {enabled|disabled}] - NFSV4.2 Security Label Support (privilege: advanced)
This optional parameter specifies whether to enable security labels for NFSv4.2. The default
setting is disabled at the time of creation.
```

Support for labeled NFS means that ONTAP now recognizes and understands the NFS client's SELinux label settings.

Labeled NFS is covered in RFC-7204.

Use cases include:

- MAC labeling of virtual machine (VM) images
- Data security classification for the public sector (secret, top secret, and so on)
- Security compliance
- Diskless Linux

In this release, ONTAP supports the following enforcement modes:

- Limited Server Mode. ONTAP cannot enforce the labels but can store and transmit them.

  Note: The ability to change MAC labels is also up to the client to enforce.

- Guest Mode. If the client is not labeled NFS-aware (v4.1 or lower), MAC labels are not transmitted.

ONTAP does not currently support Full Mode (storing and enforcing MAC labels).

Name services

In enterprise NAS environments, thousands of clients, users, and groups are interacting with storage systems every day. These clients, users, and groups require easy management that is consistent across all NAS clients. User1 in client A should not be different than user1 in client B, and client A and client B should not use the same host names or IP addresses.

That's where name services come in.

DNS

DNS servers provide a centralized way to create and manage IP addresses, host names, and business-critical service records. When all clients and storage systems point to the same DNS configurations, then there is consistency in host name <-> IP mapping without the need to manage thousands of local files.

DNS is critical in many applications and network services, such as:

- Kerberos
- LDAP
- Active Directory

Use of DNS in NFS environments is highly recommended – especially when Kerberos and LDAP are involved.

Dynamic DNS

In DNS, it's possible to have clients send DNS updates to the DNS servers when IP addresses are added, deleted, or modified. This feature reduces the amount of management overhead required for DNS, but is only possible if the DNS server supports it.

ONTAP provides a method for data LIFs to send updates to DNS servers through dynamic DNS. This is managed with the `vserver services name-service dns dynamic-update` commands.

DNS load balancing

In some cases, a host name might not just be a single IP address but might be a front-end for multiple IP addresses. In ONTAP, an SVM might have multiple data LIFs. NFS clients generally access these network interfaces with a DNS host name that load balances connections across the multiple IP addresses. ONTAP supports a few methods to load balance connections with DNS:

- Off-box DNS (round-robin through A records)
- On-box DNS (DNS forwarding/delegation to the ONTAP DNS server)
- Third-party load balancer (hardware or software gateway)

On-box DNS or off-box DNS?

ONTAP provides a method to service DNS queries through an on-box DNS server. This method factors in a node's CPU and throughput to help determine which available data LIF is the best one to service NAS access requests:

- Off-box DNS is configured by way of the DNS administrator creating multiple A name records with the same name on an external DNS server that provides round-robin access to data LIFs.
- For workloads that create mount-storm scenarios, the ONTAP on-box DNS server cannot keep up and balance properly, so it's preferable to use off-box DNS.

For more information, see TR-4523: DNS Load Balancing in ONTAP.

Identity management name services

For identity management, LDAP and NIS provide a central repository for users and groups, as well as netgroup functionality. These centralized services offer a way for clients and servers to maintain the same information to ensure predictable, consistent identities when accessing NAS file systems.

ONTAP supports both LDAP and NIS for name services, but LDAP is recommended over NIS for its security and replication support.

LDAP

The recommended method for identity management for users, groups, and netgroups is to use a LDAP server. Not only does LDAP centralize the source for name services across NFS clients and servers, it also provides a way to secure communication via secure binds and searches using SSL or Kerberos to encrypt the LDAP packets. NIS servers don't offer support for this by default.

Additionally, LDAP servers offer easier ways to replicate their information across multiple servers – particularly when using Active Directory for UNIX identity management. For more information about setting up LDAP for use with ONTAP NAS environments, see TR-4835: How to Configure LDAP in ONTAP.

NIS

NIS databases can also be used for users, groups, and netgroup name services with ONTAP. ONTAP uses the standard NIS *.byname functionality for lookups using ypserv calls. Any NIS server that leverages standard NIS functionality can be used for lookups – including Windows Active Directory.

ONTAP can also enable local NIS group.byname and netgroup.byname functionality (similar to NIS slave) for chatty NIS environments. This helps reduce the overall load on the network and NIS servers when enabled.

Local files

Local files (such as passwd, group and netgroup) are also supported as name service sources. With ONTAP, a storage administrator can either build the files with ONTAP commands, through the UI (UNIX user and group creation), or they can import flat files from servers by using the `load-from-uri` commands. By default, ONTAP SVMs support up to 64,000 entries for local UNIX users and groups.

If local files are going to be the primary name service and there will need to be more than 64,000 entries, then enabling scaled/file-only mode would be an option.

Scaled mode/file-only mode

Scaled mode/file-only mode for local users and groups in ONTAP 9.1 and later allows storage administrators to expand the limits of local users and groups by enabling a diag-level name service option and then using the `load-from-uri` functionality to load files into the cluster to provide larger numbers of users and groups. Scaled mode/file-only mode can also add performance improvements to name service lookups, because there is no longer a need to have external dependencies on name service servers, networks, and so on. However, this performance comes at the expense of ease of management of the name services, because file management adds overhead to the storage management and introduces more potential for human error. Additionally, local file management must be done per cluster, adding an extra layer of complexity.

To enable this option for users and groups, run the `vserver services name-service unix-user file-only` and `vserver services name-service unix-group file-only` commands.

After the mode is enabled, run the following command to load the user and group file from URI:

```
cluster::*> vserver services name-service unix-user load-from-uri
```

Note: If loading files larger than 10MB for users and 25MB for groups, use the `-skip-file-size-check` option.

When using file-only mode, individual operations on users and groups are not allowed. This configuration is not currently supported in NetApp MetroCluster or SVM disaster recovery (SVM DR) scenarios.

Can you still use external name services when using file-only mode?

File-only mode does not mean you cannot use LDAP or NIS as a name service; it means that the local users and groups are managed with files only (as opposed to replicated database entries). LDAP and NIS lookups will still work properly when file-only mode is enabled.

Default local users

When an SVM is created by using a Vserver setup or System Manager, the default local UNIX users and groups are created, along with default UIDs and GIDs.

The following example shows these users and groups:

```
cluster::> vserver services unix-user show -vserver vs0
              User            User   Group  Full
```

```
Vserver         Name             ID     ID     Name
--------------- ---------------- ------ ------ --------------------------------
nfs             nobody           65535  65535  -
nfs             pcuser           65534  65534  -
nfs             root             0      0      -

cluster::> vserver services unix-group show -vserver vs0
Vserver         Name                 ID
--------------- -------------------- ----------
nfs             daemon               1
nfs             nobody               65535
nfs             pcuser               65534
nfs             root                 0
```

Note: When using file-only mode, be sure the preceding users exist in the files being used to manage the cluster. After file-only mode is enabled, the default users are removed if the uploaded file does not include them.

Local user impact

When file-only mode is enabled, the default local users of `root`, `pcuser`, and `nobody` are removed if the file being loaded does not have the users. Be sure to include the local users and groups in your passwd/group files when using file-only mode.

Limits

The following section covers the limits for using local users and groups in ONTAP. These limits are cluster-wide.

Table 11) Limits on local users and groups in ONTAP.

	Local UNIX users/groups	Scaled-mode users/groups
Local users and groups maximum entries	65,536	Users: 400,000 Groups: 15,000 Group memberships: 3,000 SVMs: 6
Scaled-mode users and groups maximum file sizes	Not applicable	Passwd file size (users): 10MB* Group file size: 25MB* *Group and passwd file sizes can be overridden with `-skip-file-size-check` but larger file sizes have not been tested.

As previously mentioned, the local UNIX user and group limits are cluster-wide and affect clusters with multiple SVMs. Thus, if a cluster has four SVMs, then the maximum number of users in each SVM must add up to the maximum limit set on the cluster.

For example:

- SVM1 has 2,000 local UNIX users.
- SVM2 has 40,000 local UNIX users.
- SVM3 has 20 local UNIX users.
- SVM4 would then have 23,516 local UNIX users available to be created.

Any attempted creation of any UNIX user or group beyond the limit would result in an error message.

For example:

```
cluster::> unix-group create -vserver NAS -name test -id 12345
```

```
Error: command failed: Failed to add "test" because the system limit of {limit number}
       "local unix groups and members" has been reached.
```

Multiprotocol NAS

ONTAP supports multiprotocol NAS access to the same datasets in an SVM. With ONTAP multiprotocol NAS access, users and groups can use CIFS/SMB and NFS to access volumes or qtrees and leverage ACLs for users and groups, along with setting file and folder ownership as desired. For details on multiprotocol NAS, see TR-4887: Multiprotocol NAS in ONTAP – Overview and Best Practices.

Qtrees

Qtrees allow a storage administrator to create folders from the ONTAP UI or CLI to provide logical separation of data within a volume. Qtrees provide flexibility in data management by enabling unique export policies, unique security styles, quotas, and granular statistics.

Qtrees have multiple use cases and are useful for home directory workloads because qtrees can be named to reflect the user names of users accessing data, and dynamic shares can be created to provide access based on a user name.

The following list provides more information about qtrees in FlexGroup volumes:

- Qtrees appear as directories to clients.
- Qtrees can be created at the volume level; you cannot currently create qtrees below directories to create qtrees that are subdirectories.
- Qtrees are created and managed the same way as a FlexVol qtree is managed.
- Qtrees cannot be replicated using SnapMirror. Currently, SnapMirror only performed at the volume level. If you want more granular replication with a volume, use junction paths.
- A maximum of 4,995 qtrees is supported per volume. Quota monitoring and enforcement (enforcement in ONTAP 9.5 and later for FlexGroup volumes) can be applied at the qtree or user level.

Qtrees and file moves

A qtree is considered a unique file system in ONTAP. Although it looks like a directory from a NAS client perspective, some operations might behave differently than if it were an actual directory. One example of that is moving a file between qtrees in the same volume.

When a file move is performed in a volume across directories, the file is simply renamed to a new name, and the file moves happens within seconds because the move is inside of the same file system.

When a file move occurs between two qtrees, the file is copied to the new location rather than being renamed. This causes the operation to take much longer.

This is a behavior that occurs whether the qtree lives in a FlexVol or a FlexGroup volume.

Qtree IDs and rename behavior

After a noninherited export policy is applied to a qtree, NFS file handles change slightly when dealing with operations between qtrees. ONTAP validates qtree IDs in NFS operations, which affect things such as file renames and moves when moving to or from a qtree in the same volume as the source folder or qtree. This is considered a security feature, which helps prevent unwanted access across qtrees, such as in home directory scenarios. However, simply applying export policy rules and permissions can achieve similar goals.

For example, a move or rename to or from a qtree in the same volume results in an Access Denied error. The same move or rename to or from a qtree in a different volume results in the file being copied. With larger files, the copy behavior can make it seem like a move operation is taking an unusually long time, whereas most move operations are near-instantaneous because they are simple file renames when in the same file system or volume.

This behavior is controlled by the advanced privilege option and is covered in detail in the NetApp Knowledge Base article Permission denied while moving files between qtrees when NFS option '-validate-qtree-export' is enabled.

From the Knowledge Base article, these are the behaviors of different operations.

```
Assuming that file permissions allow and that client is allowed by export policies to access both
source and destination volume/qtree, these are the current permutations with the 'validate-qtree-
export' flag enabled or disabled:

Enabled:
- Rename in same volume and qtree: SUCCESS
- Rename in same volume, different qtrees: EACCESS
- Rename between volumes where qtree IDs differ: EACCESS
- Rename between volumes where qtree IDs match: XDEV

Disabled:
- Rename in same volume and qtree: SUCCESS
- Rename in same volume, different qtrees: SUCCESS
- Rename between volumes where qtree IDs differ: XDEV
- Rename between volumes where qtree IDs match: XDEV
```

Note: NFS3ERR_XDEV and NFS3ERR_ACCESS are defined in RFC-1813.

To change the behavior of renames and moves across qtrees, modify `-validate-qtree-export` to `disabled`. For more information, see Validating qtree IDs for qtree file operations.

Note: There is no known negative effect caused by disabling the `-validate-qtree-export` option, outside of allowing renames across qtrees.

File handle effect for qtree exports

Normally, the NFS export file handles that are handed out to clients are 32 bytes or less in size. However, with qtree exports, an extra few bytes are added to create 40-byte file handles. In most clients, this is not an issue, but older clients (such as HPUX 10.20, introduced in 1996) might have problems mounting these exports. Be sure to test older client connectivity in a separate test SVM before enabling qtree exports, because there is currently no means to change the file handle behavior after qtree exports have been enabled.

Mounting multiple Qtrees in the same volume on the same NFS client

Although qtrees effectively act as independent file systems, if they live in the same volume, then the NFS conversation between client and server involves the same MSID/file handle from the parent volume. This can result in the NFS client seeing the qtrees as the same file system mounted twice, and the used space is the same regardless of what is actually being used in each qtree.

For example, these two qtrees are mounted to the same client at different mount points.

```
# mount | grep qtree
10.193.67.214:/testvol/qtree1 on /mnt/qtree1 type nfs
10.193.67.214:/testvol/qtree2 on /mnt/qtree2 type nfs
```

They both show the same space usage before we copy a file.

```
# df -h | grep qtree
10.193.67.214:/testvol/qtree1  973G  2.0M  973G   1% /mnt/qtree1
10.193.67.214:/testvol/qtree2  973G  2.0M  973G   1% /mnt/qtree2
```

Then we copy a 3.8GB file to qtree1. Both qtrees show the same space used.

```
# cp debian-8.2.0-amd64-DVD-1.iso /mnt/qtree1/
# df -h | grep qtree
10.193.67.214:/testvol/qtree1  973G  3.8G  970G   1% /mnt/qtree1
10.193.67.214:/testvol/qtree2  973G  3.8G  970G   1% /mnt/qtree2
```

We can get around this by applying a simple monitoring quota to one of the qtrees. Just by doing this, the proper space usage is seen.

```
cluster::*> quota report -vserver NFS
Vserver: NFS

                                    ----Disk----  ----Files-----   Quota
Volume   Tree      Type    ID        Used  Limit    Used   Limit   Specifier
-------  --------  ------  --------  -----  -----  ------  ------   ---------
testvol  qtree1    tree    1
                                    3.73GB      -       2       -   qtree1
testvol  qtree2    tree    2           0B      -       1       -   qtree2
testvol            tree    *           0B      -       0       -   *

# df -h | grep qtree
10.193.67.214:/testvol/qtree1  973G  3.8G  970G   1% /mnt/qtree1
10.193.67.214:/testvol/qtree2  970G     0  970G   0% /mnt/qtree2
```

Subdirectory exports

Qtrees can be exported through NFS, which provides a single-level subdirectory path to define unique export policies and rules for clients. However, individual directories cannot have export policies and rules applied to them, and qtrees currently can only be created at the volume level in ONTAP. If your environment requires exports lower in the directory tree, a combination of volumes, qtrees, and junction paths can be used to simulate subdirectory exports. However, this does not secure the entire path, because each level in the junction path has to allow read access to the export policy rules for the clients to allow traversal.

For example, you could create a subdirectory export like this:

```
/volume1/qtree1/volume2/qtree2/volume3/qtree3
```

Each object in this path can be exported to the NFS clients with unique policies and rules. If you need greater levels of security for these folders, consider using NTFS security styles/ACLs or Kerberos for NFS.

User and group owners

Starting in ONTAP 9.8, you can set the user and group owner of a qtree from the ONTAP CLI with `qtree create` or `qtree modify`. In previous releases, this was done through the NAS protocol from a client. This is currently only available through the CLI or a REST API. There is no ZAPI or ONTAP System Manager support.

```
[ -user <user name> ]               User ID
[ -group <group name> ]             Group ID
```

Nondisruptive operations with NFS

This section covers nondisruptive operations (NDO) with NFS in ONTAP and scenarios with NDO behavior for NFS clients. In some cases, even NFSv3 can be disrupted by specific planned and unplanned events. The reason for this occurrence is that even though NFSv3 is a stateless protocol, there are still underlying mechanisms such as locking and NFS server-side caches that can come into play during disruptive events.

Replay/reply cache

The replay (or reply) cache in ONTAP is crucial to preventing NFS requests from trying nonidempotent requests twice. Nonidempotent requests are requests that can change data structures. For example, reading a document twice at the same time is an idempotent operation because it's harmless. Editing that document twice at the same time is a nonidempotent operation and can be harmful if the document doesn't have locking in place to protect it. The replay/reply cache in ONTAP helps keep track of which operations have arrived on the storage in case a network issue causes a client to resend the same operation. The cache is used to reply to the operation rather than retrying in the storage layer.

This cache is stored at the data layer with the volumes. When this cache is lost, CREATE operations can fail with E_EXIST and REMOVE operations can fail with E_NOENT. Table 12 shows different scenarios in which replay cache is kept or lost, which determines the disruptiveness of the operation.

Table 12) Replay/reply cache NDO behavior.

Operation	Result (NFSv3 and NFSv4.x)
Volume move	Replay cache is moved with volume.
Aggregate relocation or storage giveback operation	Replay cache is lost.
LIF migrate (same node)	Replay cache remains intact.
LIF migrate (different node)	Replay cache is lost.
Unplanned takeover	Replay cache is lost.
Planned takeover	Replay cache is lost.

File locking

File locking mechanisms were created to prevent a file from being accessed for write operations by more than one user or application at a time. NFS leverages file locking either using the NLM process in NFSv3 or by leasing and locking, which is built in to the NFSv4.x protocols. Not all applications leverage file locking, however; for example, the application "vi" does not lock files. Instead, it uses a file swap method to save changes to a file.

When an NFS client requests a lock, the client interacts with the ONTAP system to save the lock state. Where the lock state is stored depends on the NFS version being used. In NFSv3, the lock state is stored at the data layer. In NFSv4.x, the lock states are stored in the NAS protocol stack.

In NFSv3 environments, locks are managed by the NLM protocol, which is ancillary to the NFS protocol. As a result, when locks are used in NFSv3, there might be stale locks left over after failovers that need to be cleaned up manually. NFSv4.x locks are reclaimed based on a lease model and do not need manual cleanup. For more information about NFS file locking, see "File locking concepts."

To view or remove file locks in an SVM, run the following commands in Advanced Privilege:

```
cluster::> set advanced
cluster::*> vserver locks
    break show
```

When potentially disruptive operations occur, lock states do not transfer in some instances. As a result, delays in NFS operations can occur as the locks are reclaimed by the clients and reestablished with their new locations. Table 13covers the scenarios in which locks are kept or lost.

Table 13) Lock state NDO behavior.

Operation	NFSv3 result	NFSv4.x result
Volume move	Lock state is moved with volume.	Lock state is moved with volume.

Operation	NFSv3 result	NFSv4.x result
Aggregate relocation or storage giveback operation	Lock state is not moved; up to 45s outage when locks are in use.	Lock state is not moved; up to 90s outage when locks are in use.
LIF migrate (same node)	Lock state is not stored in NAS protocol stack; no disruption.	Lock state remains intact; still on local node; no disruption.
LIF migrate (different node)	Lock state is not stored in NAS protocol stack; nothing to move; no disruption.	Lock state is not moved; up to 90s outage when locks are in use.
Unplanned takeover	Lock state is not moved; up to 45s outage when locks are in use.	Lock state is not moved; up to 90s outage when locks are in use.
Planned takeover	Lock state is not moved; up to 45s outage when locks are in use.	Lock state is not moved; up to 90s outage when locks are in use.

Impact of NFSv4.x locks on failover scenarios

In failover scenarios (planned, unplanned or testing), you might notice that there is a noticeable pause in client I/O when NFSv4.x is in use. This pause is normal, because the lock states are being negotiated to maintain coherency and ensure that other clients aren't able to lock files that are already in use, which can result in unwanted writes to files.

This protocol-specific pause can take upwards of 45–90 seconds, which is untenable in some production environments. This pause is caused by a grace period, where NFS locks remain intact until the grace period times out or the client/server communication resumes.

This is covered in How does the NFSv4 grace periods work?

The grace period in ONTAP is active in two separate areas of the storage architecture and is set to 45 seconds by default:

- **Network.** When a data LIF is moved to another node (either manually or through port failure), the grace period for lock reclamation is set by the NFS server option `-v4-grace-seconds`. Because this is set at the NFS server level, all NFSv4.x operations to the SVM are affected.
- **Data/WAFL.** When a storage failover event occurs, the grace period for lock reclamation is set by the node-level option `locking.grace_lease_seconds`. Because this is set at the node level, this affects all NFSv4.x operations to that node. This option should be set on all nodes participating in data access using NFSv4.x.

In most instances, NetApp does not recommend lowering the grace period values, for the following reasons:

- The default value of 45 seconds helps ensure NFSv4.x clients have enough time to reclaim locks before resuming I/O. Currently, the data layer (for storage failovers) does not have the ability to break out of the default grace period before the timer runs out, even if all clients have sent RECLAIM_COMPLETE to the storage system. Bug 1392916 has been opened for this issue to enhance the failover logic for lock reclamation.
- If the grace period values are tuned too low, there is a risk that another client may try to write to the same file or byte-range that was previously locked, which can result in file corruption.
  - For single write workloads (where there is no risk of other clients trying to write to a locked file), lowering the grace periods doesn't present the same type of risk as workloads where multiple clients may be writing to a file at any given time.
- Reclaim times can vary depending on the number of locks and clients. In environments with many locks/clients, reclaims take longer than environments with fewer locks/clients. The number of clients/locks are a factor in how low you can safely set the grace period values.

However, if you need to modify the grace period for data LIF failover operations, run the following command:

```
cluster::> nfs server modify -vserver DEMO -v4-grace-seconds 45
```

To modify the grace period for storage failover operations, run the following command:

```
cluster::> node run [node names or *] options locking.grace_lease_seconds
```

Difference between grace seconds and lease seconds

There are two options in an ONTAP NFS server that control the timeout behavior of NFSv4 locks. Although they are similar, they are not identical. The section “NFSv4.x locking” covers these options as well.

- `v4-lease-seconds` – This is how long ONTAP grants a lock/lease to the client.
- `v4-grace-seconds` – This is how long ONTAP and the client will try to maintain that lock state during LIF failovers/migrations.

The minimum value set for `v4-grace-seconds` must be at least 1 second longer than the `v4-lease-seconds` value (enforced by ONTAP). This is because if a lease is granted for 45 seconds, you’ll want to ensure you keep trying for at least that long, plus additional time in case the failover event exceeds the lease seconds. These are the default values of those options:

```
cluster::*> nfs server show -vserver DEMO -fields v4-lease-seconds,v4-grace-seconds
vserver v4-lease-seconds v4-grace-seconds
------- ---------------- ----------------
DEMO    30               45
```

By default, an NFSv4 lease is granted to a client for 30 seconds. If a failure event occurs (such as network outage or storage failover), the lease will exist for 30 seconds. For an additional 15 seconds, ONTAP and the client will try to reestablish those locks. If network or storage failures exceed 45 seconds, those locks are released and the client/application must re-establish the lock on its own.

NFSv4.1 sessions

In ONTAP, NFSv4.1 sessions are supported. With NFSv4.1 sessions, LIF migrations can be disruptive to NFSv4.1 operations, but they are less disruptive than with NFSv4.0. For more information, see RFC-5661, section 2.10.13. For ways to get extra performance with sessions for both NFSv3 and NFSv4.x, see the section called “Nconnect.”

What happens during LIF migrations in NFSv4.x?

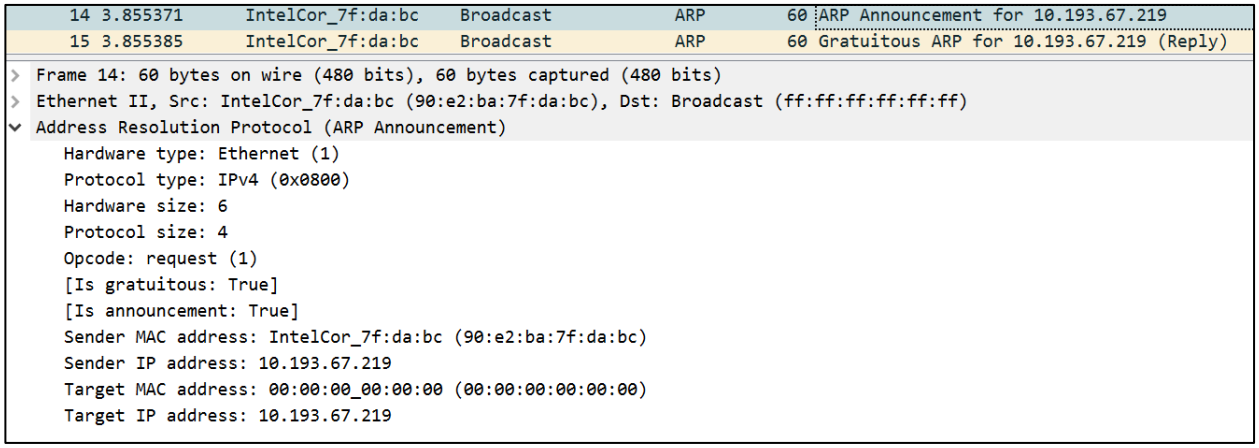
When a data LIF hosting NFSv4.x traffic is migrated in ONTAP, existing NFSv4.x traffic must be quiesced until a safe point in the process to move the LIF. After the NFS server is determined safe to allow the migration, the LIF is then moved to the new location and lock states are reclaimed by NFS clients. Lock state reclamation is controlled by the NFS option `-v4-grace-seconds` (45 seconds by default). With NFSv4.1 sessions, this grace period is not needed, because the lock states are stored in the NFSv4.1 session. Busier systems cause longer latency in LIF migrations, because the system must wait longer for the operations to quiesce and the LIF waits longer to migrate. However, disruptions occur only during the lock reclamation process.

LIF migrations with NFSv3

When a LIF migrates due to storage failover or port failure, ONTAP broadcasts an ARP announcement over the network for the IP address to inform clients that the MAC address has changed for the data LIF. As a result, the clients update their ARP tables to reflect that change. If a client cannot update the ARP entry (such as if a client firewall like AppArmor or SELinux is blocking ARP broadcasts), the NFS access still attempts to use the old MAC address, which causes access failures/mount hangs until the LIF fails

back to the original MAC address or the ARP cache is updated. Figure 9 shows the ARP during LIF migration.

Figure 9) Gratuitous ARP during LIF migration.



You can view the ARP caches on clients and compare the MAC entries with the port information on the NetApp cluster.

For example, this is what the client sees:

```
# arp -a x.x.x.a
demo.ntap.local (x.x.x.a) at 90:e2:ba:7f:d4:bc [ether] on eno16780032
```

This is what ONTAP sees:

```
cluster::*> net int show -vserver DEMO -address x.x.x.a -fields curr-port,curr-node
vserver lif   curr-node          curr-port
------- ----- ------------------ ---------
DEMO    data2 node2              e2a

cluster::*> net port show -node node2 -port e2a -fields mac
node               port mac
------------------ ---- -----------------
Node2              e2a  90:e2:ba:7f:d4:bc
```

When the LIF is migrated, the client's ARP cache gets updated with the new port's MAC address.

```
cluster::*> net int migrate -lif data2 -destination-node node1 -destination-port e2a

cluster::*> net port show -node node1 -port e2a -fields mac
node               port mac
------------------ ---- -----------------
node1              e2a  90:e2:ba:7f:da:bc

# arp -a x.x.x.a
demo.ntap.local (x.x.x.a) at 90:e2:ba:7f:da:bc [ether] on eno16780032
```

Safely decommissioning data LIFs in use with NFS

In some cases, you might want to decommission an IP address in a data SVM. However, if those IP addresses are actively being used by NFS mounts/SMB shares, then you face the potential of creating an unintentional outage in your environment. This becomes potentially more problematic if those IP addresses are a part of a DNS load balancing zone. Clients might still try to make active connections to IP addresses that have been removed, and clients that were previously already connected to those IP addresses will still try sending operations to that IP address, even if it has been removed from the SVM. A TCP connection does not automatically update the IP address until a remount is performed.

If you plan on decommissioning, removing, or even modifying the admin status of a data LIF to down, use the following gudielines:

- If the data LIFs are participating in a DNS load balance zone or round robin, remove those LIFs from the zone some time prior to the maintenance window. For ONTAP on-box DNS, use `net int modify -dns-zone none`. For off-box DNS, delete any A/AAAA records for those IP addresses. The goal is to ensure no new clients mount to the IP addresses.
- Before modifying the interfaces to `-status-admin down`, check for existing network connections:
  - In ONTAP 9.6 and earlier, use `network connections active show`.
  - For SMB/CIFS clients, use `cifs session show`.
  - For NFS clients in ONTAP 9.6 and later, use `nfs connected-clients show`.
- Keep in mind that there may be some inactive clients out there that have mounts connected; these lists only show recently accessed connections (~24 hours or so). However, those clients are likely not an issue when decommissioning interfaces, because they don't use the mounts. To be thorough, check the connections for a few days until the number reaches zero or you have notified those clients still connected about the maintenance window.
- After you have confidence that no mission-critical clients are accessing the IP addresses being decommissioned, you can remove the interfaces. If you are removing multiple interfaces, don't remove all of them at once in case you missed a stray client or two. This helps minimize disruption in case there's a mistake.
- For examples of network connections active show commands, see the section "Viewing active NFS connections in the cluster."
- For examples of NFS connected-clients show, see the section "NFS connected-clients sample output."

pNFS and LIF outages

pNFS provides a way for NFSv4.1 clients to redirect traffic to nodes that own the volumes containing the data being read or written, which offers performance benefits by way of data locality. Doing this requires a data LIF in the SVM on each node of the cluster that participates in data access so that there is always a possible local path. If nonroutable LIFs exist in the SVM, then pNFS might still select those data LIFs and an outage can occur. If using pNFS, avoid adding data LIFs to the SVM that cannot communicate with the NFSv4.1 clients.

In the event a data LIF goes offline, then data I/O falls back to the MDS, which is the IP address the mount is attached to. If the data LIF goes down during data I/O, then there is an up to 60 second delay until the client falls back to the MDS connection to resume data I/O.

Wdelay/No_wdelay

Some applications (such as RedHat OpenShift) make a specific requirement for the NFS server export option no_wdelay. This is to help protect against cache incoherency, performance, and other issues that some NFS servers that do not guarantee writes might have with application interoperability. ONTAP does not make use of this export option, because all writes in ONTAP are immediately available after the write has been acknowledged by the NFS client.

Direct connect NFS

Direct connect NFS refers to using a direct connection from an NFS client to the storage system. Because NFS is an Ethernet-based protocol, there is no technical reason this won't work. However, in a NetApp cluster, there are multiple nodes used for NFS connectivity. In the event of a storage failover, cable failure, port failure, and so on, the directly connected client will have no way to communicate with the NFS export.

It is highly recommended to use NFS over a full-fledged network for the best reliability and performance.

Choosing volume styles: FlexGroup or FlexVol?

When deploying volumes to use with NFS workloads, you have two choices of volume styles available:

- **FlexVol volumes** are the standard volume type available in ONTAP and span a single node's hardware.
- **FlexGroups** are volumes that are made of up multiple FlexVol member volumes spanning multiple hardware domains in a cluster that provide a number of advantages over FlexVol volumes including:
  - Volume sizes greater than 100TB (20PB tested).
  - File counts greater than 2 billion (400 billion tested).
  - Multi-threaded metadata operations that provide 2-6x performance for high-ingest workloads.
  - Ability to use multiple nodes in a cluster to automatically balance workloads.
  - FlexVol-like management for ease of use.
  - Nondisruptive expansion when a volume reaches capacity.

In most NFS workloads, a FlexGroup volume provides more benefits over FlexVol volumes. The main caveat when deciding is to check feature parity between the volume styles to see if necessary features in your environment are supported or not. For more information regarding FlexGroup volumes, including a deeper look at deployment and decision points, see TR-4571: NetApp FlexGroup Volumes Best Practices and Implementation.

NFS auditing

The following section covers the setup and use of NFS auditing, which can use either NFSv4.x audit ACEs (UNIX security styles) or Windows audit ACEs (NTFS security styles).

NFS audit setup

The main requirement for setting up NFS auditing is to have an audit ACE on the volume that requires auditing. The audit ACE can be Windows or NFSv4.x, but in NFS-only environments, the ACE must be NFSv4.x. As a result, NFSv4.x must be enabled in the NFS server, and an NFSv4.x admin client needs to be used to set up the auditing.

Enabling auditing on ONTAP systems

For more information about NFSv4.x ACLs, see the "NFSv4.x ACLs."

After NFSv4.x and NFSv4.x ACLs are enabled, enable NFS auditing with the following command:

```
cluster::> vserver audit create -vserver nfs -destination /unix -rotate-size 100MB
```

This command allows auditing for NFS and CIFS access on the junction path `/unix` for the SVM named `nfs`. After auditing is enabled on the ONTAP system, the AUDIT ACEs should be created.

Note: If using inheritable audit ACEs, be sure to create at least one inheritable allow or deny ACE on the parent directory to avoid access issues. For more information, see bug 959337.

Creating NFSv4 AUDIT ACEs

To create an NFSv4 AUDIT ACE, mount the volume on which auditing was enabled using NFSv4.x. After the volume is mounted, create an AUDIT ACE on the volume, files, and/or directories where auditing is required.

An AUDIT ACE can be used to track ALLOW or DENY for a variety of operations, including:

- Read
- Write
- Execute
- Append
- Delete

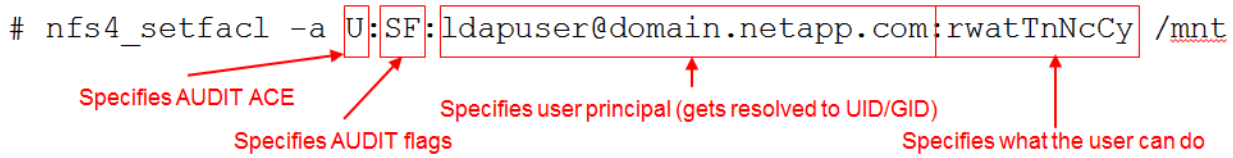
For information about all of the ACE permissions in NFSv4, see http://linux.die.net/man/5/nfs4_acl.

Each Linux client uses a different method of assigning NFSv4.x ACEs. In RHEL/CentOS/Fedora, the commands `nfs4_setacl` and `nfs4_getacl` are used.

An AUDIT ACE leverages flags to specify if auditing should be for successes, failures, or both. AUDIT ACEs use the ACE type of U.

Figure 10 shows an example of setting NFSv4 audit ACE.

Figure 10) Example of setting NFSv4 audit ACE.



After the AUDIT ACE is applied and the user that is being audited attempts access, and the events get logged to an XML file on the volume.

For an example of a logged NFS audit event, see “NFS audit event example.”

NFS best practices

The following section covers best practices with NFS environments. Best practices are simply recommendations that should be considered when using and deploying NFS. Best practices are not hard requirements, and in many cases are dependent on a variety of factors in the environment.

General ONTAP best practices

The following list provides general ONTAP best practices:

- Upgrade to the latest ONTAP version for bug fixes and newest NFS features.
- Use Active IQ for upgrade recommendations and proactive remediation.
- Enable Autosupport to allow for a seamless and efficient support experience.
- Configure load-sharing mirrors for the vsroot volume in NFSv3 environments.
- For the best performance, use NetApp All-Flash FAS systems.
- Enable all storage efficiencies for maximum storage utilization.
- Follow best practices for aggregate and storage configurations as per the product documentation.
- In most cases, leave the defaults unchanged unless there is a specific reason to change the settings.
- Install Active IQ Unified Manager and configure to monitor your ONTAP cluster.
- Set up proactive alerts for events such as storage failovers, volume capacity alerts, and so on.
- Consider deploying FlexGroup volume for your NFS workloads, particularly in high file count environments.

ONTAP data LIF best practices with NFS environments

ONTAP allows storage administrators to provide the following benefits:

- Seamless scale-out storage
- Multiprotocol unified access (NFS, CIFS, and SAN)
- Nondisruptive operations

This is done by way of a secure multitenant architecture with SVMs.

What are SVMs?

Storage virtual machines (SVMs) are logical storage containers that own storage resources such as flexible volumes, logical interfaces (LIFs), exports, CIFS shares, and so on. Think of them as a storage blade center in your cluster. These SVMs share physical hardware resources in the cluster with one another, such as network ports/VLANs, aggregates with physical disk, CPU, RAM, switches, and so on. Data access can occur on any data network interface owned by SVMs, regardless of the location in the cluster. As a result, load for SVMs can be balanced across a cluster for maximum performance and efficiency or to leverage SaaS functionality, among other benefits. Alternately, a single SVM can be used to present a monolithic storage device to an environment.

Data LIF considerations

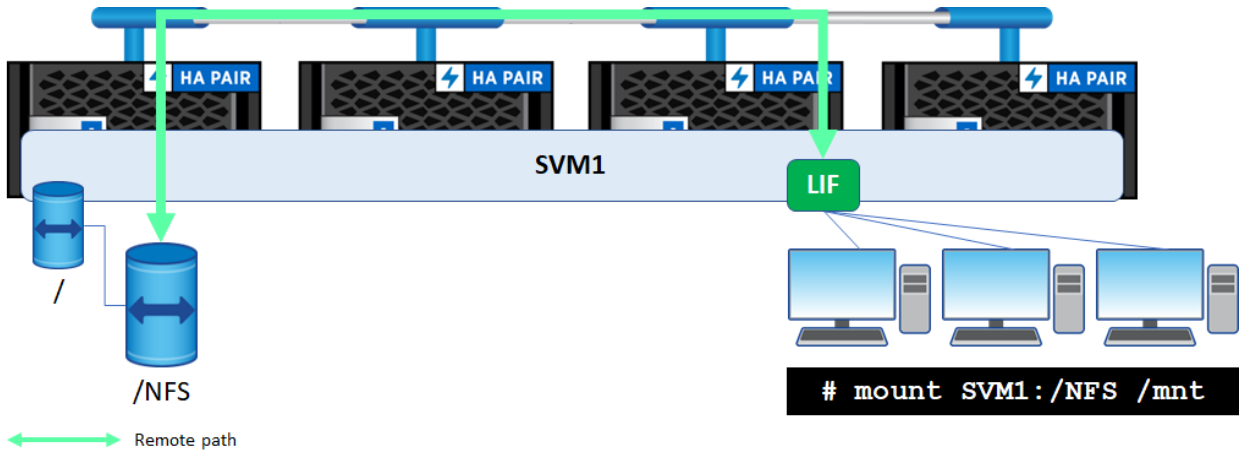
Data LIFs can live on any physical port in a cluster that is added to a valid broadcast domain. These data LIFs are configured with SVM-aware routing mechanisms that allow the correct pathing of Ethernet traffic in an SVM, regardless of where a valid data LIF lives in the cluster. When designing a network for NAS interaction, one of two approaches can be taken.

Option #1: Simplicity approach — single LIF per SVM

Essentially, all it takes to access NAS data in ONTAP is a single network IP address that is routable to network clients. In many environments, a single network interface will suffice for NAS workloads. If the underlying physical network port fails, or if a storage node is taken over by its HA partner, then the network IP address migrates to another working port in the cluster. Using a single network interface reduces the number of IP addresses needed, but it also limits the amount of potential network bandwidth that would be available to a workload. Sending all NAS traffic to a single node in the cluster also limits the number of resources (such as CPU and RAM) available, so if a workload is expected to require high throughput or will connect hundreds to thousands of clients, then option #2 might be a better choice.

Figure 11 shows a single LIF NAS interaction.

Figure 11) Single LIF NAS interaction.



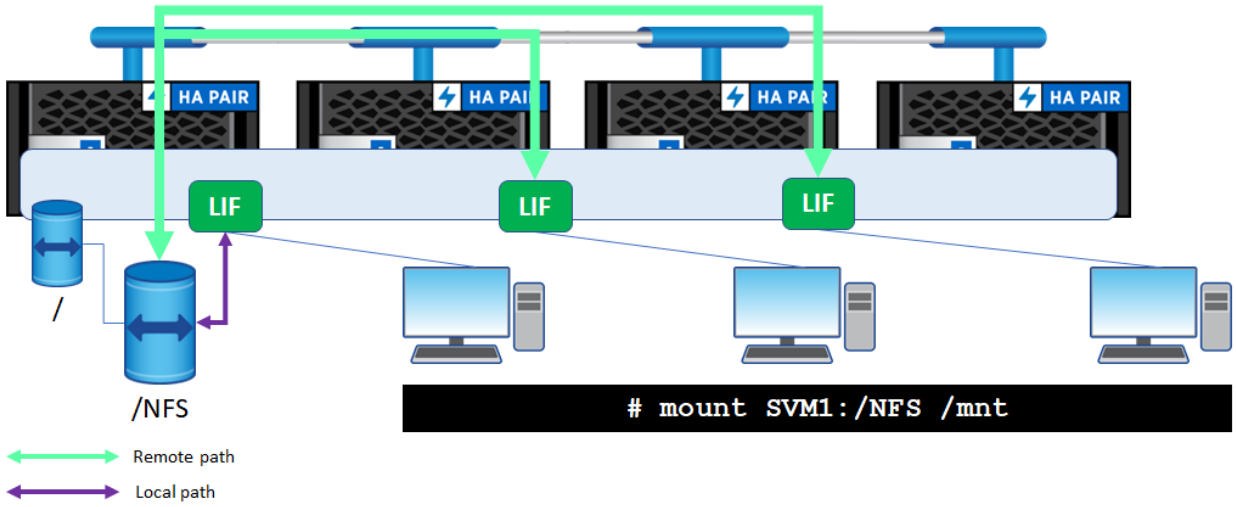
Option #2: Performance approach — multiple data LIFs per SVM

In an ONTAP cluster, multiple nodes can be made available for NAS connectivity and storage. Remember, ONTAP clusters using NAS can only scale up to 24 nodes. Multiple nodes mean multiple physical resources, such as CPU/RAM/network interfaces. As a result, having more than one data LIF in an SVM can add considerable performance to your NAS workloads. Spreading network connections across nodes alleviates CPU and network port contention, as well as avoiding scenarios where a node might have too many TCP connections. For network load balancing of NAS connections, round robin DNS, on-box DNS, or standard load balancing hardware can be leveraged. For more information regarding on-box DNS (and how to configure it), see TR-4523: DNS Load Balancing in ONTAP.

In situations where the best possible performance is required, or where many clients will be accessing a NAS device at the same time, creating multiple data LIFs per SVM is a sound approach. Additionally, use of load balancing NAS features such as NFS referrals, CIFS autolocation and pNFS will require a data LIF on each node where data resides.

Figure 12 shows multiple LIFs NAS interaction.

Figure 12) Multiple LIFs NAS interaction.



Data LIF locality recommendations

In ONTAP, you have the ability to leverage data locality features such as NFS referrals, CIFS autolocation, and pNFS for NAS traffic regardless of where the volumes live in a cluster. For NFS referrals and CIFS autolocation, the initial TCP connection is automatically redirected to a network interface that is local to the requested volume. If the volume being used is a FlexGroup volume, then NFS referrals and CIFS autolocation should not be used.

pNFS provides a metadata path on the initial mount request, but all reads and writes are automatically redirected to local volumes through pNFS layout calls. pNFS is only available for the NFSv4.1 protocol and only with NFS clients that support it. FlexGroup volumes can only use pNFS in ONTAP 9.7 and later. For more information about pNFS, see “Parallel network file system.”

Without autolocation features, managing data LIF locality to avoid the cluster network adds management complexity that might not be worth the effort, as the performance impact for most NFS workloads is negligible. Ideally, NAS connections connect to data LIFs that are local to the volumes, but with FlexGroup volumes/scale-out NAS and larger cluster backend networks, this becomes less important.

Data locality benefits and considerations

The following section describes the benefits and considerations of data locality in ONTAP and how to approach these concepts with simplicity in mind.

Ability to spread the load across nodes and leverage all the available hardware in a cluster

When creating volumes and network interfaces, consider deploying workloads across multiple nodes in the cluster to maximize the performance headroom. Why pay for hardware you do not use?

Simplicity approach: ONTAP provides automated provisioning of storage when using ONTAP System Manager. This takes into account available performance headroom and attempts to place new volumes on nodes that are less utilized. Additionally. FlexGroup volumes provision across multiple nodes in a cluster and automatically balance workloads to a single namespace.

Ability to balance network connections across multiple cluster nodes

Clusters are single entities, as are SVMs. But they do have underlying hardware that has its own maximums, such as number of connections and so on.

Simplicity approach: Create multiple data LIFs per SVM and mask those interfaces behind a DNS round robin name or DNS forwarding zone leveraging the ONTAP on-box DNS feature. In addition, leverage FlexGroup volumes to spread workloads across multiple nodes.

Ability to enable data locality in the event of a volume move

If you move a volume to another node, you can be certain you still have a local path to the data if every node has a data LIF for the SVM. When moving volumes in ONTAP to new nodes, existing TCP connections remain in place for NAS clients. As a result, these NAS operations traverse the cluster network.

Simplicity approach: Do nothing. In most cases, NAS clients notices no difference in performance to these NAS shares. If using NFSv4.1, consider using pNFS.

Network port exhaustion with a large number of NFS clients

In environments with a large number of clients connecting through NFS, it is important to keep in mind that, by default, the number of mount and NFS ports are limited to 1,024.

This number is controlled with the options:

```
mount-rootonly
nfs-rootonly
```

By default, `mount-rootonly` is set to Enabled and `nfs-rootonly` is set to Disabled.

In some circumstances, the number of ports used to mount or for NFS operations might be exhausted, which then causes subsequent mount and NFS operations to hang until a port is made available.

If an environment has thousands of clients that are mounted through NFS and generating I/O, it is possible to exhaust all ports on an NFS server. For example, there is one scenario with ESX using NFS datastores, because some legacy best practices would call for a data LIF/IP address per datastore. In environments with many volumes/datastores, this scenario created a situation where NFS ports were overrun. The remediation for this situation is to disable the `mount-rootonly` and/or the `nfs-rootonly` options on the NFS server. This solution removes the 1 to 1,024 port range limit and allows up to 65,534 ports to be used in a NFS server. For more information on these options, see “The rootonly options – nfsrootonly and mountrootonly.”

Note: For ESX/NFS best practices in ONTAP, see TR-4597: VMware vSphere with ONTAP.

This situation affects the source port (client-side) only. The server-side mountd, portmapper, NFS, and nlm ports are designated by ONTAP.

NFS behind network address translation

NFS maintains a reply cache to keep track of certain operations to make sure that they have been completed. This cache is based on the source port and source IP address. When network address translation (NAT) is used in NFS operations, the source IP or port might change in flight, which could lead to data resiliency issues. If NAT is used, static entries for the NFS server IP and port should be added to make sure that data remains consistent.

In addition, NAT could also lead to issues with NFS mounts hanging due to how NAT handles idle sessions. If using NAT, the configuration should take idle sessions into account and leave them open indefinitely to prevent issues. NAT can also create issues with NLM lock reclamation.

Ultimately, the best practice for NAT with NFS would be to avoid using it if possible and instead create a data LIF on the SVM. If NAT is necessary, work with the NAT vendor to configure it properly for NFS operations.

LIF service policies

ONTAP 9.6 and later introduces LIF service policies, which replace the concept of roles on network data interfaces in ONTAP. LIF policies can be applied or removed to a network interface to allow/disallow traffic without needing to recreate the network interface.

You can see which service policy your interfaces have by running the following command:

```
cluster::*> net int show -vserver DEMO -lif data -fields service-policy
  (network interface show)
vserver lif  service-policy
------- ---- ------------------
DEMO    data default-data-files
```

LIF service policies create several default policies, but you can add custom policies if you choose. These are the following default policies, which allow SAN, NAS, or management traffic. Only one policy can be assigned to a data LIF at a time.

```
cluster::*> network interface service-policy show -vserver DEMO
Vserver   Policy                     Service: Allowed Addresses
--------- -------------------------- ------------------------------------------
DEMO
          default-data-blocks        data-core: 0.0.0.0/0, ::/0
                                     data-iscsi: 0.0.0.0/0, ::/0
                                     data-fpolicy-client: 0.0.0.0/0, ::/0

          default-data-files         data-core: 0.0.0.0/0, ::/0
                                     data-nfs: 0.0.0.0/0, ::/0
                                     data-cifs: 0.0.0.0/0, ::/0
                                     data-flexcache: 0.0.0.0/0, ::/0
                                     data-fpolicy-client: 0.0.0.0/0, ::/0

          default-management         data-core: 0.0.0.0/0, ::/0
                                     management-ssh: 0.0.0.0/0, ::/0
                                     management-https: 0.0.0.0/0, ::/0
                                     data-fpolicy-client: 0.0.0.0/0, ::/0
```

To create policies that allow only NFS or only CIFS/SMB, you can use `network interface service-policy create`, or you can add or remove services with `network interface service-policy add-service` or `network interface service-policy remove-service`. This can all be done without taking an outage.

For more information, see LIFs and service policies in ONTAP 9.6 and later.

NFS security best practices

While NFS stands for Network File System, it has playfully been referred to as Not For Security. NFSv3 and earlier were limited in security scope, because client IP addresses, numeric user, and group IDs and other identifying information would be passed over networks in clear text.

Because the default security mechanisms for NFSv3 security are export rules and basic user mode-bit permissions, spoofing IP addresses and user identities for access is fairly simple and straightforward. As a result, use the following suggestions to better secure your NFS environments.

Choosing your NFS version

When deploying an NFS environment, the NFS version being used is one of the many things to consider. NFSv3 is by far the most popular NFS version, but NFSv4.x is increasing in popularity as more and more businesses are requiring more stringent security requirements:

- If security is the most important factor in your environment, you should at the very least test NFSv4.x, as it is designed for security.
- If raw performance is the priority, then NFSv3 would be the prudent choice, as its statelessness provides distinct performance advantages over NFSv4.x.
- If locking is important, then NFSv4.x offers the most robust and resilient options.
- If a blend of functionality is required, then it is best to test which NFS version makes the most sense for your workloads.

The section called “NFS version considerations” breaks down the details of all NFS versions, advantages, features, and migration considerations.

Securing NFSv3

While NFSv4.x is superior to NFSv3 for security, there are still some steps you can take to better secure the protocol.

Export policy rule considerations

Export policies and rules are the first gateway for securing an NFS export. These rules allow you to specify one or more clients and define the level of access they’ll receive when they mount an export. These rules override file-level permissions. For example, if you set “read/write” (rwrule) to Never, then even if the UNIX permissions are 777, that client won’t be able to write to the export. For more information about NFS export policies and rules, see “Export concepts.”

The way in which you set up your export policies and rules affects whether a client is allowed to mount the export at all. When securing NFSv3 exports, consider the following information:

- If an export policy has no rules and is applied to a volume or qtree, then no one will be allowed access to mount.
- If an export policy rule doesn’t include a specific client (either through IP, host, name, netgroup/, or ubnet), that client will be unable to mount.
- If the export policy rule on the vsroot volume does not allow read access to a specific client, then that client will be unable to mount the export. Read access is required on an export policy rule to allow clients to traverse the paths to the volume. Vsroot is “/” in the export path.
  - Vsroot uses the default export policy when it’s created. By default, the “default policy” has no export rules.
- When setting the clientmatch for the vsroot’s export policy rule to allow path traversal, consider setting it by netgroup, subnet, or a list of clients. Avoid setting the entry to 0.0.0.0/0 or 0/0; this allows all clients to traverse/read the export. For more information, see “Access control to vsroot.”

- To allow (or prevent) root access for a client, use the Superuser export policy rule option. For more information, see “Mapping all UIDs to a single UID (squash_all).”
- In multiprotocol NAS environments (CIFS/SMB and NFS access), the way ONTAP responds to permission or owner changes on NFS mounts of volumes with NTFS security depends on the setting of the export policy. The options are Fail (send an error) or Ignore (fail silently with no error).
- Setting `-rw-rule`, `-ro-rule` or `-superuser` to None will squash a client’s access to the user set with `-anon`. In most cases, `-anon` should be set to 65534. Exceptions would include setting `-anon` to 0 to allow root access to an admin host or setting it to an explicit user to control file ownership for a specific client/set of clients.
- Rather than setting host names and IP addresses in the export clientmatch, consider using netgroups to mask host/client lists in the `-clientmatch` field. Netgroups can live in local files on an SVM, in LDAP, or NIS.
- For clients that are doing read/write access, set `-superuser` to None and `-anon` to 65534 to prevent root access.
- Consider setting up a group of clients that are designated for administrative/root-level tasks and set the export policy to allow root access. See “Examples of controlling the root user” for details on how to allow root access.
- If you’re using Kerberos for NFSv3 mounts, ensure you’ve included both `sys` and `krb5*` in your rules. NFSv3 uses ancillary protocols and only uses Kerberos for the NFS portion. Limiting access to export policy rules in NFSv3 to only `krb5*` will result in failed mounts, because `sys` access is needed for portmapper, mount, and so on.
- Avoid using `any` for `-rw-rule`, `-ro-rule`, `-superuser`; allowing `any` reduces the security effectiveness of the policy rule. Instead, specify the authentication method you require in those options.
- Avoid using `any` for `-protocol`, because this allows access to protocols you might not want to use. If you require both NFSv3 an NFSv4.x access to mounts, set the protocol to `nfs`. If you only want NFSv3 or only NFSv4.x to access a mount, specify either `nfs3` or `nfs4`.

Client firewalls

By default, NFS clients enable firewalls (such as SELinux in RHEL or AppArmor in Ubuntu). In some cases, those firewalls can impact NFS operations. Always check the client firewall rules if NFS access issues occur, and, if necessary, during troubleshooting, stop the firewall service.

Firewall port rules

For NFSv3, there are a number of ports that need to be allowed by firewalls to provide full access to all NFSv3 operations. In general, firewalls have built-in rules, such as NFS that keeps a list of well-known NFS ports. However, many security environments recommend changing well-known ports for better security. The ports listed in “Default NFS ports in ONTAP” show which ports ONTAP uses by default for NFSv3, which ports can be modified and which NFS version they apply to.

The following list provides a set of general recommendations and guidance for firewall rules. These are not NetApp requirements; they are simply ideas to help secure NFS environments.

- Consider changing the default port numbers for NFS ancillary protocols to ports that are not well-known and adjusting firewall ports as needed.
- Consider allowing only the NFSv3 ports that are needed by your environment. For example, if rquota is not being used, don’t allow it though a firewall. There are several ports that are required for NFS mounts, however, so you should always allow those port numbers.
  - Portmapper (111), mountd (635) and NFS (2049) are required to allow NFS access. Other ancillary ports (NLM, NSM, rquota) are only needed if an application or client needs them. Test in your environment accordingly.

- NFS mounts generate a source port on the client. By default, ONTAP sets the range of allowed source ports between 1–1024 (`-mount-rootonly`). In some cases, this range of port numbers might not be large enough to accommodate the number of clients that are mounting. If more source port numbers are needed, set `-mount-rootonly` to Disabled and modify the firewall rules to accommodate that change. For more information, see "The rootonly options – nfsrootonly and mountrootonly."
- NFS operations also use a range of source ports that can be controlled through `-nfs-rootonly`. By default, this value is set to Disabled, which means the port range is 1–65536. If a low number of NFS clients is used (100 or less), consider setting this option to Enabled to reduce the port range to 1–1024 for better security.
- Avoid opening up NFS ports outside of your local network. If NFS over a WAN is required, strongly consider using NFS Kerberos with krb5p for end-to-end encryption. Alternately, consider using NetApp FlexCache volumes to localize NFS traffic to a remote site. NetApp FlexCache volumes use TLS 1.2 to encrypt communications. For more information about FlexCache volumes, see TR-4743: FlexCache in ONTAP.
- Export policy rules and name services for UNIX identity management might be dependent on external name servers (such as DNS, LDAP, NIS, Active Directory, and so on). Make sure the firewall rules allow traffic for those name servers.
- Some firewalls might drop idle TCP connections after a set amount of time. For example, if a client has an NFS mount connected, but doesn't use it for a while, it's considered Idle. When this occurs, client access to mounts can hang, because the network connection has been severed by the firewall. Keepalives can help prevent this, but it is better to address this either by configuring firewalls to actively reject packets from stale sessions or configuring the ONTAP NFS server options `-idle-connection-timeout` and `-allow-idle-connection`. These options were introduced in ONTAP 9.5. For more information, see Bug 1072144.

Permissions

While it's possible to set read and write permissions for clients in export policies, you cannot set permissions for specific users and groups with just export rules. In addition, you can't get more granular than read or write with export policies. As a result, file and folder permissions would be needed. NFSv3, by default, uses a basic permission model for files and folders via mode-bits, which means that permissions can only be explicitly controlled for the file/folder owner and everyone else. Additionally, the level of permissions is limited to what is defined in RFC-1813.

Figure 14 shows what each permission mode bit does from 0–7.

Table 14) UNIX Mode-bit levels.

Symbolic notation	Mode bit value	Access level
-	0	No access
rwx	7	All access
rw-	6	Read & write
r-x	5	Read and execute
r--	4	Read
-wx	3	Write and execute
-w-	2	Write
---x	1	Execute

It is also possible to leverage an NFSv4.x admin client to mount a file system and add NFSv4.x ACLs, which are then honored by NFSv3 clients. This provide more granular permissions for NFSv3 environments.

The following additional recommendations can help better secure NFSv3. These are general recommendations and do not apply to all environments, so be sure to test in your environment.

- For the vsroot volume, limit the ower and group to Read and Execute and everyone else to Execute only privileges (551). Vsroot is generally small, so you want to prevent users from writing to it when possible. Limiting everyone other than the owner and group to Execute permissions only allows traversal of the path and does not allow file/directory listings in the vsroot volume.
- Avoid setting permissions to 7 on volumes/qtrees, unless you want to allow root access to the volume owner or group.
- Follow standard UNIX permission best practices when managing access to NFS file systems.
- Consider using NFSv4.x ACLs or NTFS security style volume permissions (if multiprotocol NAS is in use) for more robust permission management.
- If using LDAP or NIS for UNIX identities, ensure the user and group lookups are returning the proper users and group memberships from the NFS client and server.

User authentication

When an NFS mount is accessed, the authentication method is negotiated between the client and server. For standard AUTH_SYS access, the user credentials are passed over the wire in plain text, which means anyone spying on the network can see those credentials and use them to gain access to the file system. With standard AUTH_SYS, there is no user name/password gateway to safeguard against unintended access.

For the best security result with authentication in NFSv3, consider enabling NFS Kerberos for your NFS mounts. AUTH_GSS (or Kerberos) configuration provides a few levels of protection for access, including:

- User name/password interaction with a Kerberos Key Distribution Center server (such as Windows Active Directory, and FreeIPA)
- Security settings protecting against changing the time on a client to reuse an expired Kerberos credential.
- Encryption of the NFS operations, ranging from logins (krb5), data integrity checks (krb5i) and end-to-end NFS encryption (krb5p).
- An increase of the maximum number of user group membership from 16 in AUTH_SYS to 32 in AUTH_GSS. (ONTAP provides a way to increase this number to 1,024 in the “Auxiliary GIDs — addressing the 16 GID limitation for NFS” section)

For more information, see TR-4616: NFS Kerberos in ONTAP.

Encryption

ONTAP provides in-flight encryption for NFS through krb5p, but also offers a few options for encrypting data at-rest. These options include:

- SED and NSE drives
- NVE
- NAE

For more information about encryption and security hardening, see TR-4569: Security Hardening Guide for ONTAP.

Showmount

Historically, `showmount` on NFS clients has been how users can see exported file systems on an NFS server. By default, NFS servers in ONTAP enable the `showmount` functionality to show exported paths, but do not list the allowed client access. Instead, `showmount` displays that (everyone) has access.

The “/” root path is not displayed in the `showmount` commands by default. To control the behavior of `showmount` and show the root path, use the following NFS server options:

```
-showmount
-showmount-rootonly
```

This functionality might cause security scanners to flag the NFS server that has a vulnerability, because these scanners often use `showmount` to see what is being returned. In those scenarios, you might want to disable `showmount` on the NFS server.

However, some applications make use of `showmount` for functionality, such as Oracle OVM. In those scenarios, inform the security team of the application requirements.

Securing NFSv4.x

NFSv4.x offers more robust security than NFSv3 and should be used in any NFS environment that will prioritize security over all else.

NFSv4.x security features include:

- Single port for all NFS operations
- ID domain string matching
- NFSv4.x ACLs
- Integrated Kerberos support for user name/password authentication and end-to-end encryption
- Client ID checks to help prevent client spoofing

Export policy rule considerations

The export policy rule considerations for NFSv4.x generally mirror those covered in “Securing NFSv3.” Review those considerations for guidance.

Client firewalls

NFS clients, by default, enable firewalls (such as SELinux in RHEL or AppArmor in Ubuntu). In some cases, those firewalls might impact NFS operations. Always check the client firewall rules if NFS access issues occur, and, if necessary during troubleshooting, stop the firewall service.

Firewall port rules

Unlike NFSv3, NFSv4.x combines all NFS operations into compound calls that all leverage the same NFS port – 2049. This port is not modifiable in ONTAP, so if you plan on using NFSv4.x, firewall port rules should be modified to allow port 2049.

Permissions

NFSv4.x provides permissions through mode-bits, as well as NFSv4.x ACLs. The advantages of ACLs are covered in “Benefits of Enabling NFSv4 ACLs,” but for the most robust and granular security with NFSv4.x, use ACLs.

User authentication

For user authentication in NFSv4.x, the same guidance discussed in “Securing NFSv3” should be followed – use Kerberos whenever the highest level of security is required.

Encryption

For encryption in-flight and at-rest in NFSv4.x, the same guidance discussed in “Securing NFSv3” should be followed.

Name services best practices

ONTAP provides support for external name service servers for DNS, LDAP/NIS, Kerberos and Active Directory. In NAS environments, name services can provide a wide array of benefits for performance, resiliency, and manageability. Detailed name service best practices can be found in TR-4668: Name Service Best Practices, but here are a few general name service best practices to follow:

- **Centralize name services whenever possible.** In environments with hundreds or thousands of users, groups, host names, and so on, trying to keep local copies of files in sync across the organization can result in inconsistencies, duplicate host names or user credentials, and general confusion across platforms. Pointing your clients and ONTAP cluster to the same set of name services vastly reduces the management overhead in these environments and simplifies access permissions and troubleshooting.
- **Always configure DNS.** In the same vein as centralization of name services, having a centralized service to keep track of host names, IP addresses, DNS zones, and domains and service records rather than trying to maintain these manually is essential. DNS in particular is critical to the functionality of many NAS services – particularly Windows Active Directory, Kerberos KDCs, LDAP (for service record lookups and load balancing) and NFSv4.x.
- **In NFSv4.x environments, use LDAP for UNIX identity management.** Because NFSv4.x leverages name strings for user authentication, and because user name matching across server and clients controls how NFSv4.x functions, having a centralized name service for UNIX identities for NFSv4.x clients and ONTAP to point to makes using NFSv4.x that much simpler.
- **When using name service servers, have more than one.** Name service servers might receive thousands, hundreds of thousands, or even millions of requests every day. Those requests have a performance cost – both on the network and on the name service servers. ONTAP does a good job of caching requests, but name service servers can still get overwhelmed unless you've set up multiple servers and added them to the client and ONTAP configuration. Even better, load balancing several servers behind the same DNS name allows for easier configuration, reduces the need to re-configure clients and servers if name service addresses change, and masks the number of servers that are being used to load balance requests.

General NAS networking considerations for ONTAP

A NAS connection in ONTAP follows these general guidelines.

- When a client connects to a data LIF in ONTAP, a unique ID is assigned to that TCP connection to keep track of the operations.
- If the same NAS client makes subsequent NAS connection requests to the same data LIF, that unique ID is reused, even if a different data volume is accessed.
- If the same NAS client makes a subsequent NAS connection request to a different data LIF (even if it is on the same node), then a new unique ID is assigned for that client.
- Each node in a cluster has a finite limit of possible TCP connections for NAS operations. When this limit is reached, new connections are denied until resources are freed up. The ONTAP event management system (EMS) logs will show errors if this happens.
- Using a single data LIF in a cluster provides only one access point for incoming NAS connections and adheres to the TCP connection limits on the single node. As a result, if you mount 1000 clients to a single node, you run the risk of exhausting system resources faster than if you mount 1000 clients to four different cluster nodes. Avoid pointing all NAS clients to a single data LIF in large scale environments—such as EDA—to mitigate resource exhaustion.
- In an HA pair, if each node has more than half of the maximum connection IDs allowed and a storage failover occurs, then you might hit the connection ID limit on the surviving node as clients attempt to reestablish connectivity to the NFS mounts and connections will fail. If possible, keep NAS connections to about 50% of the total allowed limit in case a failover occurs,

- NFSv3 mounts have ancillary protocols that are used during the connection process, such as mount and portmapper. These UDP-based protocols are also assigned unique IDs during the initial connection process and age out after 60 seconds. In mount storm scenarios (thousands of clients mounting at the same time), it might be possible to exhaust connection resources artificially and prematurely due to the temporary ID assignments for the ancillary protocols. Consider staggering client mounts and spreading your automounter workloads across multiple nodes in a cluster, or, if you are using Cloud Volumes ONTAP, across multiple Cloud Volumes ONTAP instances that use FlexCache volumes.
- Unmounts also generate connection IDs for the mount/portmap protocols.
- Both UDP and TCP connections count against the total network connections on a node.
- Specifying `tcp` as a mount option eliminates those extra UDP calls and reduces the total number of connections generated on each mount/unmount. In environments that generate a lot of mounts/unmounts, use `tcp` as a mount option.
- NFSv4.x always uses TCP and does not use mount/portmapper for mounts/unmounts, so no extra connection IDs are generated for that NFS version.
- When the maximum connection limits are exceeded, an EMS message is generated (maxCID.threshold.exceeded or maxCID.limit.exceeded).
- Modern NAS clients have features that attempt to add more parallel network connections to a single NAS connection, such as SMB multichannel and NFS nconnect. When using features that provide more TCP threads per NAS share, they generally improve performance, but they also use up more unique IDs in the NAS stack of ONTAP. For example, a normal NFSv3 mount might only use a single unique ID, but an NFSv3 mount using nconnect=4 might use up to four unique IDs per mount. Keep this in mind when designing for scale in EDA environments.
- Each unique ID in ONTAP has a limit of 128 concurrent NAS operations. If this limit is exceeded by the client sending more concurrent operations than ONTAP is able to manage, a form of flow control is enacted on the NAS stack in ONTAP until resources are freed up. You can mitigate this behavior with a client-side configuration or by using more unique IDs per client. For more information about network concurrency with NAS, see “Network connection concurrency and TCP slots: NFSv3”.
- In addition to per-node connection ID limits and per-connection concurrent NAS operation limits, there are also node-level limits for total available NAS operations at any given time. Each time a NAS operation is performed, a resource context is reserved until that operation is completed. At that time, the resource is released back to the system. If too many resources are requested at once on a single node, then performance issues can occur. See “Exec context throttling” for information about these resources and how to best prevent issues.

Examples of mount/unmount connection behavior

The following examples show how connection IDs are generated with NFSv3 mount/unmounts with and without the `tcp` mount option. You can see these connection IDs by using the command `network connections active show -remote-ip x.x.x.x.`

Note: `To reduce the total number of connections for NFSv3 mounts and reduce the chances of exceeding the node’s connection limits, specify the tcp mount option.`

Connection IDs generated with mounts/unmounts not using the “tcp” mount option

Before mount:

```
cluster::> network connections active show -remote-ip x.x.x.x
There are no entries matching your query.
```

Client mounts without `tcp` option:

```
# mount -o vers=3 DEMO:/home /mnt/client1
```

After mount:

```
cluster::> network connections active show -remote-ip x.x.x.x
Vserver     Interface              Remote
Name        Name:Local Port        Host:Port                    Protocol/Service
---------- ---------------------- ---------------------------- ----------------
Node: cluster-02
DEMO        data2:111              centos83-perf.ntap.local:56131
                                                               UDP/port-map
DEMO        data2:635              centos83-perf.ntap.local:44961
                                                               UDP/mount
DEMO        data2:635              centos83-perf.ntap.local:1022
                                                               UDP/mount
DEMO        data2:2049             centos83-perf.ntap.local:879 TCP/nfs
4 entries were displayed.
```

After ~60 seconds, the UDP connection IDs disappear and only the NFS connection remains:

```
cluster::> network connections active show -remote-ip x.x.x.x
Vserver     Interface              Remote
Name        Name:Local Port        Host:Port                    Protocol/Service
---------- ---------------------- ---------------------------- ----------------
Node: cluster-02
DEMO        data2:2049             centos83-perf.ntap.local:879 TCP/nfs
```

Client unmounts:

```
# umount /mnt/client1
```

NFS connection ID is gone, but now there are mount/portmap UDP connections for ~60 seconds:

```
cluster::> network connections active show -remote-ip x.x.x.x
Vserver     Interface              Remote
Name        Name:Local Port        Host:Port                    Protocol/Service
---------- ---------------------- ---------------------------- ----------------
Node: cluster-02
DEMO        data2:111              centos83-perf.ntap.local:36775
                                                               UDP/port-map
DEMO        data2:635              centos83-perf.ntap.local:33867
                                                               UDP/mount
DEMO        data2:635              centos83-perf.ntap.local:966 UDP/mount
3 entries were displayed.

cluster::> network connections active show -remote-ip x.x.x.x
There are no entries matching your query.
```

Connection IDs generated with mounts/unmounts with the “tcp” mount option

Before mount:

```
cluster::> network connections active show -remote-ip x.x.x.x
There are no entries matching your query.
```

Client mounts with `tcp` option:

```
# mount -o vers=3,tcp DEMO:/home /mnt/client1
```

After mount:

```
cluster::> network connections active show -remote-ip x.x.x.x
Vserver     Interface              Remote
Name        Name:Local Port        Host:Port                    Protocol/Service
---------- ---------------------- ---------------------------- ----------------
Node: cluster-02
DEMO        data2:2049             centos83-perf.ntap.local:931 TCP/nfs
```

Client unmounts:

```
# umount /mnt/client1
```

NFS connection ID is gone and there is no UDP mount/portmapper:

```
cluster::> network connections active show -remote-ip x.x.x.x
There are no entries matching your query.
```

Behavior of NFS connections with containers

Containerized environments (such as Docker or RedHat OpenShift) are becoming more prevalent in EDA environments. As a result, it is important to understand how containers impact NFS connections in ONTAP to ensure proper sizing and scale-out considerations when implementing containers using NFS mounts.

If there are NFS mounts on the container host, they will have their own connection IDs per data LIF. Containers on that host using NFS mounts often share the same network IP address as the host, but when initiating an NFS mount, it gets its own connection ID.

For example, this container host has a mount to a volume:

```
[root@centos7-docker ~]# mount | grep scripts
demo:/scripts/dockerfiles on /dockerfiles type nfs
(rw,relatime,vers=3,rsize=1048576,wsize=1048576,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec
=sys,mountaddr=x.x.x.x,mountvers=3,mountport=635,mountproto=udp,local_lock=none,addr=x.x.x.x)

cluster::> network connections active show -remote-ip x.x.x.y -fields cid,proto,service,remote-
ip,local-address,node
node                cid        vserver local-address remote-ip     proto service
------------------ ---------- ------- ------------- ------------- ----- -------
clutster-01         1011516163 DEMO    x.x.x.x       x.x.x.y       TCP   nfs

cluster::> nfs connected-clients show -node * -client-ip x.x.x.y -data-lif-ip x.x.x.x

     Node: cluster-01
  Vserver: DEMO
  Data-Ip: x.x.x.x
Client-Ip       Volume-Name      Protocol Idle-Time     Local-Reqs Remote-Reqs
--------------- ---------------- -------- ------------- ---------- -----------
x.x.x.y         scripts          nfs3     55s           73         0
```

If a container mount is initiated to the same data LIF, a new connection ID is generated for NFS along with new UDP connection IDs for mount and portmap:

```
[root@centos7-docker ~]# docker exec -it centos bash
[root@f8cac0b471dc /]# mount -o vers=3 10.193.67.237:/scripts /mnt

cluster::*> network connections active show -remote-ip x.x.x.y -fields cid,proto,service,remote-
ip,local-address,node
node                cid        vserver local-address remote-ip     proto service
------------------ ---------- ------- ------------- ------------- ----- -------
cluster-01          1011516163 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01          1011516166 DEMO    x.x.x.x       x.x.x.y UDP   port-map
cluster-01          1011516167 DEMO    x.x.x.x       x.x.x.y UDP   mount
cluster-01          1011516168 DEMO    x.x.x.x       x.x.x.y UDP   mount
cluster-01          1011516170 DEMO    x.x.x.x       x.x.x.y TCP   nfs
```

When you start a new container and mount within it, an additional NFS connection ID and UDP connection IDs are created from the same client IP, even though you are using the same data LIF, volume and node:

```
[root@centos7-docker ~]# docker run --name centos2 --rm -it --cap-add SYS_ADMIN -d
parisi/centos7-secure
16dec486692dbc1133b4c1f74c6e78aa7aab875c0aed71d0d087461a6bed8060
[root@centos7-docker ~]# docker exec -it centos2 bash
[root@16dec486692d /]# mount -o vers=3 10.193.67.237:/scripts /mnt
```

```
cluster::*> network connections active show -remote-ip x.x.x.y -fields cid,proto,service,remote-
ip,local-address,node
node                cid        vserver local-address remote-ip     proto service
------------------- ---------- ------- ------------- ------------- ----- --------
cluster-01          1011516163 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01          1011516170 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01          1011516177 DEMO    x.x.x.x       x.x.x.y UDP   port-map
cluster-01          1011516178 DEMO    x.x.x.x       x.x.x.y UDP   mount
cluster-01          1011516179 DEMO    x.x.x.x       x.x.x.y UDP   mount
cluster-01          1011516183 DEMO    x.x.x.x       x.x.x.y TCP   nfs
```

As a result, connection IDs can begin to add up quickly in environments that generate a lot of NFS mounts in a short period of time,particularly if containers are involved.

For an idea of how scaling container environments can create a potential impact on the NFS server, consider the following scenario: If a container host starts 1000 containers and each container makes an NFS mount request to the same data LIF in a cluster, then the total number of connection IDs that are generated on that single data LIF will range between 1000 (if specifying the `tcp` mount option) and 4,000 (3,000 UDP connection IDs for mount/portmap that age out after ~60 seconds if the `tcp` mount option is not specified).

If the containers mount to different data LIFs on the same node, the same connection ID dispersal applies.

If the containers mount to different data LIFs on different nodes, then the connection IDs are distributed across nodes, and it takes longer for the connection ID limits to be reached. The more cluster nodes and data LIFs used, the more connection ID dispersal takes place. For two nodes, 1000 connection IDs can divide into 500 per node. For a 4-node cluster, 1000 connection IDs can divide into 250 per node.

Impact of nconnect on total connections

When an NFS mount is established, a single NFS connection ID is used. However, with the new NFS option `nconnect`, it is possible to open multiple TCP connections per NFS mount. For instance, using `nconnect=8` with your NFS mount option creates as many as eight NFS connection IDs to the data LIF in the SVM. Although this can deliver extra performance benefits, it can also use up more connection IDs than expected.

Example:

```
# mount -o nconnect=8 DEMO:/home /mnt/client1

cluster::*> network connections active show -remote-ip x.x.x.x -fields cid,proto,service,remote-
ip,local-address,node
node                cid        vserver local-address remote-ip     proto service
------------------- ---------- ------- ------------- ------------- ----- -------
cluster-02          2328843073 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-02          2328843076 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-02          2328843077 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-02          2328843078 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-02          2328843079 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-02          2328843080 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-02          2328843081 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-02          2328843082 DEMO    x.x.x.y       x.x.x.x TCP   nfs
```

If you plan on using nconnect in environments that generate a large number of mounts, be aware of the connection ID limits per node for your platform, and plan to distribute connections across multiple cluster nodes to balance connections appropriately. In cases where a single node/data LIF is being used for NFS mounts, connection IDs run out much faster than if multiple nodes are used. Connection ID limits should be factored into scale discussions when architecting the solutions.

Effect of NetApp XCP on total connections

When using NetApp XCP for data migrations, it behaves similarly to nconnect when establishing connection IDs. While a single mount point might be specified, the `-parallel` option determines the total number of connection IDs established to the ONTAP node. If the `-parallel` option is not defined, XCP defaults to eight parallel connections.

Note: With XCP scans, only a single connection ID is used.

In the following example, an XCP host is used to copy data from an NFS mount with the following syntax:

```
# xcp copy -parallel 16 IP1,IP2:/files destination:/files
```

In the preceding command, XCP is creating 16 network threads per IP address specified. In this case, both IP1 and IP2 are on the same node of the source cluster. As a result, that node gets 34 total connection IDs established for this job (18 on the first IP specified; 16 on the second IP):

```
cluster::*> network connections active show -remote-ip x.x.x.y -fields cid,proto,service,remote-
ip,local-address,node
node                 cid        vserver local-address remote-ip     proto service
-------------------- ---------- ------- ------------- ------------- ----- -------
cluster-01           1011516323 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516327 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516328 DEMO    x.x.x.z       x.x.x.y TCP   nfs
cluster-01           1011516329 DEMO    x.x.x.z       x.x.x.y TCP   nfs
cluster-01           1011516330 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516331 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516332 DEMO    x.x.x.z       x.x.x.y TCP   nfs
cluster-01           1011516333 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516334 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516335 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516336 DEMO    x.x.x.z       x.x.x.y TCP   nfs
cluster-01           1011516337 DEMO    x.x.x.z       x.x.x.y TCP   nfs
cluster-01           1011516338 DEMO    x.x.x.z       x.x.x.y TCP   nfs
cluster-01           1011516339 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516340 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516342 DEMO    x.x.x.z       x.x.x.y TCP   nfs
cluster-01           1011516343 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516344 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516345 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516346 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516347 DEMO    x.x.x.z       x.x.x.y TCP   nfs
cluster-01           1011516348 DEMO    x.x.x.z       x.x.x.y TCP   nfs
cluster-01           1011516349 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516350 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516351 DEMO    x.x.x.z       x.x.x.y TCP   nfs
cluster-01           1011516352 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516353 DEMO    x.x.x.z       x.x.x.y TCP   nfs
cluster-01           1011516354 DEMO    x.x.x.z       x.x.x.y TCP   nfs
cluster-01           1011516355 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516356 DEMO    x.x.x.z       x.x.x.y TCP   nfs
cluster-01           1011516357 DEMO    x.x.x.z       x.x.x.y TCP   nfs
cluster-01           1011516358 DEMO    x.x.x.z       x.x.x.y TCP   nfs
cluster-01           1011516359 DEMO    x.x.x.x       x.x.x.y TCP   nfs
cluster-01           1011516360 DEMO    x.x.x.z       x.x.x.y TCP   nfs
34 entries were displayed.
```

When using XCP, try to spread the connections across multiple IP addresses on multiple nodes and be aware that the number of parallel threads means more total network connection IDs are in use.

Viewing connection ID maximums and allocations

In ONTAP, it is possible to view how many connection IDs (CIDs) are available for a node, how many connection IDs are currently in use, and which clients are using those connections.

To view connection ID information, note the following commands:

network connections active (admin privilege)

```
cluster::> network connections active ?
  delete                      *Delete an active connection in this cluster
  show                        Show the active connections in this cluster
  show-clients                Show a count of the active connections by client
  show-lifs                   Show a count of the active connections by logical interface
  show-protocols              Show a count of the active connections by protocol
  show-services               Show a count of the active connections by service
```

statistics start/show -object cid (diag privilege)

```
cluster::*> statistics start -object cid
```

node run netstat -na (diag privilege)

```
cluster::*> node run -node node1 netstat -na
```

Example of connection ID behavior and maximums using statistics

To gather statistics for this object, run the following command in diagnostic privilege:

```
cluster::*> statistics start -object cid
```

Statistics collection is being started for sample-id: sample_55

To view the output:

```
cluster::*> statistics show -object cid
```

The following is a sample of how those results might appear before an NFS mount is established:

```
Object: cid
Instance: cid
Start-time: 6/25/2021 16:54:27
End-time: 6/25/2021 16:56:37
Elapsed-time: 130s
Scope: cluster-01

    Counter                                                     Value
    -------------------------------- --------------------------------
    alloc_failures_nomem                                            0
    alloc_failures_reserved_toomany                                 0
    alloc_failures_toomany                                          0
    alloc_total                                                     0
    cid_max                                                    115904
    execs_blocked_on_cid                                            0
    in_use                                                       352
    in_use_max                                                      0
    instance_name                                                 cid
    node_name                                                cluster-01
    process_name                                                    -
    reserved_cid                                               10526
```

In this output the node has a `cid_max` of 115904. Of those, 10526 are `reserved_cid` for ONTAP system operations, which means the total number of available connection IDs for client operations is 115904 – 10526 = 105378. If a node exceeds that limit, the EMS triggers a maxCID.limit.exceeded message.

Currently, there are 352 `in_use` CIDs.

After an NFS mount with `nconnect=8` is established on node1, this is how the numbers change:

```
Object: cid
Instance: cid
Start-time: 6/25/2021 16:54:27
```

```
End-time: 6/25/2021 17:04:54
Elapsed-time: 627s
Scope: cluster-01

    Counter                                                     Value
    -------------------------------- --------------------------------
    alloc_failures_nomem                                            0
    alloc_failures_reserved_toomany                                 0
    alloc_failures_toomany                                          0
    alloc_total                                                    14
    cid_max                                                    115904
    execs_blocked_on_cid                                            0
    in_use                                                        360
    in_use_max                                                      0
    instance_name                                                 cid
    node_name                                                 cluster-01
    process_name                                                    -
    reserved_cid                                               10526
```

Now, instead of 352 CIDs in_use, there are 360 used. This aligns with the eight active network connections created with the nconnect mount.

When an NFSv3 mount is issued without the `tcp` mount option specified, four new CIDs are `in_use` and you can see it in the statistics.

```
    Counter                                                     Value
    -------------------------------- --------------------------------
    alloc_failures_nomem                                            0
    alloc_failures_reserved_toomany                                 0
    alloc_failures_toomany                                          0
    alloc_total                                                    28
    cid_max                                                    115904
    execs_blocked_on_cid                                            0
    in_use                                                        364
    in_use_max                                                      1
    instance_name                                                 cid
    node_name                                                 cluster-01
    process_name                                                    -
    reserved_cid                                               10526
```

These CIDs include the mount and portmap UDP entries seen with the `network connections active show` command.

```
cluster::*> network connections active show -remote-ip x.x.x.x -fields cid,proto,service,remote-
ip,local-address,node
node             cid        vserver local-address remote-ip     proto service
---------------- ---------- ------- ------------- ------------- ----- -------
cluster-01       1011516253 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-01       1011516256 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-01       1011516257 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-01       1011516258 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-01       1011516259 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-01       1011516260 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-01       1011516261 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-01       1011516262 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-01       1011516268 DEMO    x.x.x.z       x.x.x.x UDP   port-map
cluster-01       1011516269 DEMO    x.x.x.z       x.x.x.x UDP   mount
cluster-01       1011516270 DEMO    x.x.x.z       x.x.x.x UDP   mount
cluster-01       1011516272 DEMO    x.x.x.z       x.x.x.x TCP   nfs
```

After the UDP entries age out (at about ~60 seconds), the CIDs are released for use by new connections, and you only have one extra CID in use by the NFS connection on data LIF IP **x.x.x.z**.

```
    Counter                                                     Value
    -------------------------------- --------------------------------
    alloc_failures_nomem                                            0
    alloc_failures_reserved_toomany                                 0
    alloc_failures_toomany                                          0
```

```
    alloc_total                                                    28
    cid_max                                                    115904
    execs_blocked_on_cid                                            0
    in_use                                                        361
    in_use_max                                                      1
    instance_name                                                 cid
    node_name                                                cluster-01
    process_name                                                    -
    reserved_cid                                               10526

cluster::*> network connections active show -remote-ip x.x.x.x -fields cid,proto,service,remote-
ip,local-address,node
node                 cid        vserver local-address remote-ip     proto service
-------------------- ---------- ------- ------------- ------------- ----- -------
cluster-01           1011516253 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-01           1011516256 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-01           1011516257 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-01           1011516258 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-01           1011516259 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-01           1011516260 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-01           1011516261 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-01           1011516262 DEMO    x.x.x.y       x.x.x.x TCP   nfs
cluster-01           1011516272 DEMO    x.x.x.z       x.x.x.x TCP   nfs
```

The “rootonly” options – nfsrootonly and mountrootonly

The `rootonly` options are added to help prevent untrusted client access. Untrusted clients (those not part of the export rules) can potentially access data by using SSH tunneling to trusted clients. However, those requests come from untrusted ports (ports greater than 1,024). This can provide a back door for clients not intended to have access.

Therefore, the enabling or disabling of the `rootonly` options hinges on need. Does the environment require more ports to allow NFS to function properly? Or is it more important to prevent untrusted clients from accessing mounts?

One potential compromise is to make use of NFSv4.x and/or Kerberos authentication for a higher level of secured access to NFS exports. TR-4616: NFS Kerberos in ONTAP covers how to use NFS Kerberos.

In these scenarios, using the `mount-rootonly` and/or `nfs-rootonly` options can alleviate these issues.

To check port usage on the client:

```
# netstat -na | grep [IP address]
```

To check port usage on the cluster:

```
cluster::> network connections active show -node [nodename] -vserver [vservername] -service nfs*
```

For example, you can specify that the client must use a port outside of the reserved port range with the mount option `noresvport` (`resvport` is the default if not specified and uses source ports between 1-1024). When you do this and `mount-rootonly` is enabled for the NFS SVM, the mount fails:

```
cluster::*> nfs show -vserver DEMO -fields mount-rootonly,nfs-rootonly
vserver mount-rootonly nfs-rootonly
------- -------------- ------------
DEMO    enabled        disabled

# mount -o noresvport,vers=3 demo:/scripts /mnt/client1
mount.nfs: access denied by server while mounting demo:/scripts
```

From a packet trace, you can see that the client source port is outside of the allowed range (36643):

```
129    x.x.x.x        x.x.x.y        MOUNT   138    V3 MNT Call (Reply In 130) /scripts
User Datagram Protocol, Src Port: 36643, Dst Port: 635
```

When you use `resvport`, the mount succeeds:

```
# mount -o resvport,vers=3 demo:/scripts /mnt/client1
#
```

A packet trace shows that the source port is within the 1024 range (703):

```
44      x.x.x.x         x.x.x.y         MOUNT   138     V3 MNT Call (Reply In 45) /scripts
User Datagram Protocol, Src Port: 703, Dst Port: 635
45      x.x.x.y         x.x.x.x         MOUNT   150     V3 MNT Reply (Call In 44)
```

When the `mount-rootonly` option is set to disabled:

```
cluster::*> nfs modify -vserver DEMO -mount-rootonly disabled
```

The mount using `noresvport` succeeds:

```
# mount -o noresvport,vers=3 demo:/scripts /mnt/client1
#
```

The trace shows that the source port is outside of the 1024 range (58323)

```
101     x.x.x.x         x.x.x.y         MOUNT   138     V3 MNT Call (Reply In 102) /scripts
User Datagram Protocol, Src Port: 58323, Dst Port: 635
102     x.x.x.y         x.x.x.x         MOUNT   150     V3 MNT Reply (Call In 101)
```

Note: Because NFSv4.x mounts do not use the ancillary mount protocols for NFS mounts, the `mount-rootonly` port does not factor in to those operations and only affects NFSv3 mounts.

Mount port exhaustion with a large number of NFS clients

In environments with a large number of clients connecting through NFS, it is important to remember that, by default, the number of mount ports are limited to 1,024. In a “mount storm” scenario (that is thousands of clients mounting or unmounting at about the same time), 1,024 ports can be exhausted quickly and create connectivity issues.

- Mount operations (only applicable to NFSv3) are limited through the NFS option `mount-rootonly`, which is set to **enabled** by default. This limits the incoming port range for mounts to 1-1024.
- For NFS operations over port 2049, the default number of incoming ports allowed is 65,534. This is controlled by the `nfs-rootonly` NFS option and is set to **disabled** by default. To limit the number of incoming NFS client source ports to 1,024, set this option to **enabled**.

In some circumstances, the number of ports used to mount or for NFS operations might be exhausted, which then causes subsequent mount and NFS operations to hang or fail until a port is made available.

If an environment has thousands of clients that are mounted through NFS and generating I/O (such as the container example in “Behavior of NFS connections with containers”), it is possible to exhaust all ports on an NFS server. For example, one scenario is with ESX using NFS datastores because some legacy best practices call for a data LIF/IP address per datastore. In environments with many volumes/datastores, this creates a situation where the NFS ports are overrun. The remediation for this situation is to disable the `mount-rootonly` and/or the `nfs-rootonly` options on the NFS server. Performing this action removes the 1 to 1,024 port range limit and allows up to 65,534 ports to be used in an NFS server for mounts and NFS operations. On the client side, you might also need to use the mount option `noresvport` to use non-privileged source ports, as NFS mounts default to the `resvport` mount option when not specified.

Mount port usage with UDP versus TCP

By default, NFSv3 mounts use UDP for the MOUNT protocol. Because UDP has no acknowledgements, ONTAP maintains the connection ID for UDP mounts for 60 seconds to ensure the mount has a chance to succeed or fail before ONTAP removes the connection ID. This also applies to unmount operations.

Because of this, a large number of mount/unmount requests at the same time might use up all of the available source ports in the 1-1024 range until the 60 second expiration is reached, and the existing mount connection IDs are cleared. Even unsuccessful mounts using UDP maintain the connection IDs for 60 seconds.

For example, this mount using a port outside of the allowed range failed:

```
# mount -o noresvport,vers=3 demo:/scripts /mnt/client1
mount.nfs: access denied by server while mounting demo:/scripts
```

From ONTAP, you can see there are three connection IDs generated from the failed mount request that expire after 60 seconds. In a mount storm, 60 seconds can be an eternity:

```
cluster::*> network connections active show -remote-ip x.x.x.x -fields remote-port,cid,service
node               cid        vserver remote-port service
------------------ ---------- ------- ----------- --------
cluster-02         2328843541 DEMO    35470       port-map
cluster-02         2328843542 DEMO    60414       mount
cluster-02         2328843543 DEMO    33649       mount
3 entries were displayed.
```

TCP uses acknowledgements in its conversations so that ONTAP does not need to maintain connection IDs after the mount request is successful. As a result, using `-o tcp` for your NFS mount option means that ports in the 1-1024 range are freed up faster, preventing most of the cases in which port exhaustion for mounts might occur.

Best practices for environments with a large number of NFS clients

The following section covers some of the best practices for environments with a large number of NFS clients, such as EDA compute farms.

Considerations for environments with mount storms

A mount storm is where a large number of NFS clients mount or unmount NFS exports to the same NFS server or cluster in a short period of time. NFS environments that heavily use automounters can experience mount storms without administrators being aware.

In a mount storm scenario, system resources can be used up quickly. The following best practices should be considered when designing an environment where it is possible to encounter mount storms.

- More cluster nodes and data LIFs on each node means that there are more available system resources to use to balance out NFS mounts, connections, and so on. When designing for large scale NFS mount environments, consider using as many nodes as possible for incoming NFS connections. ONTAP supports up to 24 nodes for NAS-only clusters.
- When using multiple data LIFs in a cluster for load balancing of connections, use a DNS load balancer to simplify the environment by masking many IP addresses behind a single host name. This enables easy addition/removal of IP addresses to the DNS FQDN without needing to notify end users of a change in connectivity and helps maintain an even balance of incoming NAS connections from clients whether automounters are used or not.
- Monitor data LIF locations and storage failovers closely. If a data LIF migrates to another node (due to port failure or manual migration), then that data LIF is using the current node's resources for NAS connections and might contribute to faster resource exhaustion. The same is true for storage failovers (planned or unplanned) because the failed node's data LIF will migrate to the surviving partner and that node must maintain all of the NAS connections. If a data LIF migration occurs, resolve the issue that caused the migration and revert the data LIF back to its home node.
- Monitor connection ID usage in the cluster by using `statistics start -object cid` or the equivalent REST API functions. Try to keep connection ID `in_use` values to approximately 50% of

the `max_cid` values for the node in case of storage failover (roughly 50,000 `in_use` connection IDs per node). You can do this by adding more nodes with new data LIFs to the cluster.

- Monitor the EMS for the events `maxCID.limit.exceeded`, `maxCID.threshold.exceeded`, and `nblade.execsOverLimit`, and contact NetApp Support if these events are generated.
- Multiplexing of NFS connections by way of more data LIFs per node or through nconnect can increase performance for some workloads, but it can also use more available resources per storage node (such as connection IDs and exec contexts). If you use multiplexing, be aware of the potential side effects (as covered in “Impact of nconnect on total connections” and “Exec context throttling”).
- If possible, use the `tcp` mount option for NFS mounts (`-o tcp`).This reduces the total number of connection IDs generated per mount/unmount operation and helps to prevent resource exhaustion (connection IDs and NFS mount ports) in mount storm scenarios.
- If you require more than 1024 available incoming NFS mount ports (for instance, if 2000 clients are mounting at the same time), you might need to consider disabling the NFS server option `mount-rootonly`, using the `noresvport` mount option on the NFS clients/automounters, and using the mount option `tcp` to reduce the number of connection IDs generated per mount. (UDP mount connections remain in cache for up to 60 seconds; TCP mount connections are removed on client ACK).

Note: NFSv4.x is not susceptible to issues with UDP and the mount protocol but has its own challenges, as covered in “NFSv4.x considerations.”

NFS client best practices

NFS client best practices are generally dependent on the applications being used. When determining how to configure NFS clients, always involve the application vendor in those discussions. The following best-practice recommendations are not set in stone and can be overridden by application recommendations or by workload testing.

Client and NFS utility versions

For NFS client versions, the best practice is usually to run the latest OS patch version available, as well as updating the NFS utilities to their latest releases to get the newest bug fixes and feature functionality. In environments where OS and NFS utility versions are qualified and standardized, this might not be an option.

As a general rule, ONTAP NFS supports and any all NFS clients that comply with RFC standards. This does not include NFS clients that have custom kernel compilations.

RPC slot tables

RPC slot table refers to the maximum allowed threads on a single TCP connection that is allowed by the NFS client and server. These values are controlled through sunrpc configuration on NFS clients. The latest NFS client versions default to a dynamic slot table value of 65536, which means that the client attempts to use as many slot tables as it can in a single TCP connection. ONTAP, however, supports only 128 slot tables per TCP connection. If a client exceeds that value, ONTAP enacts NAS flow control and pause the client operations until resources are freed up.

As a best practice, set the slot table values on NFS clients to a static value no higher than 128. In environments with many clients, this value might need to be set as low as 16. For more information, including details on the performance impact, see “Network connection concurrency and TCP slots: NFSv3.”

Mount options

NFS mount option recommendations depend solely on the workload and application being used. There is general advice for specific mount options, but the decision on which NFS options to use is dependent on

the client OS administrators and application vendor recommendations. There is no single correct recommendation for mount options. The following sections cover only a subset of NFS mount options. For a complete list of supported NFS mount options, use `man nfs` on your NFS client.

Default mount options

Linux clients set the default mount options out of the box. These default options depend on the client OS version and NFS configuration files found on the clients. The default mount options are set during a mount operation where no options are specified with the -o flag.

In some cases, mount options are negotiated with the NFS server. Specifically, in newer Linux kernels, the rsize/wsize values and the NFS versions are based on what the NFS server is set to.

For example, if NFSv4.1 is enabled and no NFS version is specified in configuration files or with the mount command, then the client uses NFSv4.1 because it is the highest supported NFS version enabled.

The following example shows output from a mount command that issued no specific options. The ONTAP NFS server was set to 1MB TCP max transfer size (`-tcp-max-xfer-size`) and has NFSv4.1 enabled.

```
# mount DEMO:/flexgroup_16 /flexgroup
# mount | grep flexgroup
DEMO:/flexgroup_16 on /flexgroup type nfs4
(rw,relatime,vers=4.1,rsize=1048576,wsize=1048576,namlen=255,hard,proto=tcp,port=0,timeo=600,retr
ans=2,sec=sys,clientaddr=10.x.x.x,local_lock=none,addr=10.x.x.y)
```

Wsize/rsize

The mount options wsize and rsize determine how much data is sent between the NFS client and server for each packet sent. This might help optimize performance for specific applications but should be set as per application vendor best practices, as what is best for one application may not be best for other applications.

Newer NFS clients autonegotiate the wsize and rsize values to what the `-tcp-max-xfer-size` value is set to on the ONTAP NFS server if the mount command does not explicitly set the values. ONTAP defaults `-tcp-max-xfer-size` to 64K and can be set to a maximum of 1MB.

Note: The general recommendation for `-tcp-max-xfer-size` is to increase that value in ONTAP to 262144 (256K) and then specify explicit mount options as the applications require it.

For examples of some performance testing with different workload types and different wsize/rsize values, see “Performance examples for different TCP maximum transfer window sizes.”

For examples of wsize/rsize recommendations for specific applications, see:

- TR-3633: Oracle on NetApp ONTAP
- TR-4435: SAP HANA on NetApp AFF Systems using NFS

NFS Readahead

Readahead in NFS is a way for NFS clients to predictively request blocks of a file to improve performance and throughput for sequential I/O workloads. Until recently, the readahead value for NFS mounts was set to 15 times the rsize value of the mount. For example, if you set rsize to 64KiB, then readahead size would be 960KiB.

In modern NFS clients (such as RHEL 8.3 and later or Ubuntu 18.04 and later), the readahead value is no longer determined by the mount rsize but is globally defaulted to 128KiB. This can cause severe negative performance consequences on reads. In newer Linux client versions that use the default 128KiB readahead value, it is recommended to set the value to a higher limit. Testing read performance with different values is the preferred method, but internal NetApp testing has found that the value can be safely set to as high as 15360KiB for sequential read workloads.

For more information about setting and viewing readahead values, consult with your client OS vendor. For example, this SUSE KB describes readahead for those OS client flavors: Tuning NFS client read ahead on SLE 10 and 11.

For a CentOS/RedHat client, the process is similar.

```
# cat /etc/redhat-release
CentOS Linux release 7.8.2003 (Core)
```

Find the BDEV information for the mount with this command (BDEV format is N:NN):

```
# cat /proc/self/mountinfo | grep /mnt/client1
125 39 0:46 / /mnt/client1 rw,relatime shared:107 - nfs DEMO:/files
rw,vers=3,rsize=1048576,wsize=1048576,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,mount
addr=10.193.67.219,mountvers=3,mountport=635,mountproto=udp,local_lock=none,addr=10.193.67.219
```

Use the BDEV information to find the readahead value (BDEV for that mount point is 0:46):

```
# cat /sys/class/bdi/0:46/read_ahead_kb
15360
```

In the above example, readahead is set to 15360KiB (15x the rsize) for the `/mnt/client1` mount and rsize is set to 1MB on my CentOS 7.8 client.

On CentOS 8.3, this is the value my mount is set to by default:

```
# cat /sys/class/bdi/0:50/read_ahead_kb
128
```

Actimeo and Nocto

NFSv3 manages shared file system consistency and application performance by using cached file/directory data and cached file/directory attributes. It is a loose consistency, because the application doesn't have to go to shared storage and retrieve data every time to use it. That can have tremendous impact to application performance. The cached information has timers that set the period to trust the cache data and at timeout, a light weight, fast getattr/access call to revalidate the data till the next time out.

There are two mechanisms that manage this process:

- **Cto.** Close to open consistency assures getting the latest data for a file, regardless of cache.
- **Actimeo.** Attribute cache timers (default 3s for file, 30s for directory).

If a client has complete ownership of data, for example, it is not shared, then there is guaranteed consistency. You can reduce getattr/access operations to storage and speed up the application by turning off cto consistency (nocto as a mount option) and by turning up the timeouts for the attribute cache management (actimeo=600 as a mount option changes the timer to 10m versus the defaults previously mentioned). In some testing, nocto reduces ~65–70% of the getattr/access calls and actimeo reduces another ~20–25%.

There are other cases that can benefit from a similar set of mount options, even though there is not complete ownership by the clients. For applications that use grids of clients such as EDA, web hosting and movie rendering and have relatively static data sets (such as the tools/libraries in EDA, the web content for the web hosting, or the textures for rendering), the typical behavior is the data set is largely cached on the clients (very few reads; no writes). In those instances, there will be many getattr/access calls coming back to storage. These data sets are typically updated through either another client mounting the file systems and periodically pushing content updates or in some case SnapMirror relationships to multiple file systems for update.

In these instances, there is a known lag in picking up new content and the application still works with potentially out-of-date data. For those scenarios, nocto and actimeo can be used to control the time

period where out-of-data date can be managed. For example, in EDA with tools and libraries and other static content, actimeo=600 works well because this data is typically updated infrequently. For small web hosting where clients need to see their data updates timelier as they are editing their sites, actimeo=10 might be acceptable. For large scale web sites where there is content pushed to multiple file systems, actimeo=60 might be more effective. As always, test in your environments.

Using this mount options reduce the workload to storage significantly in these instances (for example, a recent EDA experience reduced IOPs to the tool volume from >150K to ~6K) and applications can run significantly faster because they can trust the data in memory, rather than needing to query the NFS storage. This also helps reduce overall CPU % and load on the ONTAP nodes.

Actimeo

The actimeo mount option controls the attribute cache timeout on NFS clients. The actimeo option covers the entire range of available attribute caches, including:

```
acregmin=n
The minimum time (in seconds) that the NFS client caches attributes of a regular file before it
requests fresh attribute information from a server. If this option is not specified, the NFS
client uses a 3-second minimum.

acregmax=n
The maximum time (in seconds) that the NFS client caches attributes of a regular file before it
requests fresh attribute information from a server. If this option is not specified, the NFS
client uses a 60-second maximum.

acdirmin=n
The minimum time (in seconds) that the NFS client caches attributes of a directory before it
requests fresh attribute information from a server. If this option is not specified, the NFS
client uses a 30-second minimum.

acdirmax=n
The maximum time (in seconds) that the NFS client caches attributes of a directory before it
requests fresh attribute information from a server. If this option is not specified, the NFS
client uses a 60-second maximum.
```

Attribute caching provides some relief on networks by reducing the number of metadata calls. This also helps reduce latency to some workloads, as these metadata operations can now occur locally on the client. Attribute caching generally has no effect to the number of overall operations, unless all operations to the storage were metadata – specifically ACCESS calls.

For example, in our Customer Proof of Concept (CPOC) labs, actimeo was set to 10 minutes (600 seconds) and observed latency cut in half with an EDA workload generated by vdbench (from ~2.08ms to ~1.05ms). Figure 13 shows the actimeo default latency and Figure 14 shows an actimeo 600 latency.

Figure 13) Default actimeo latency — vdbench.

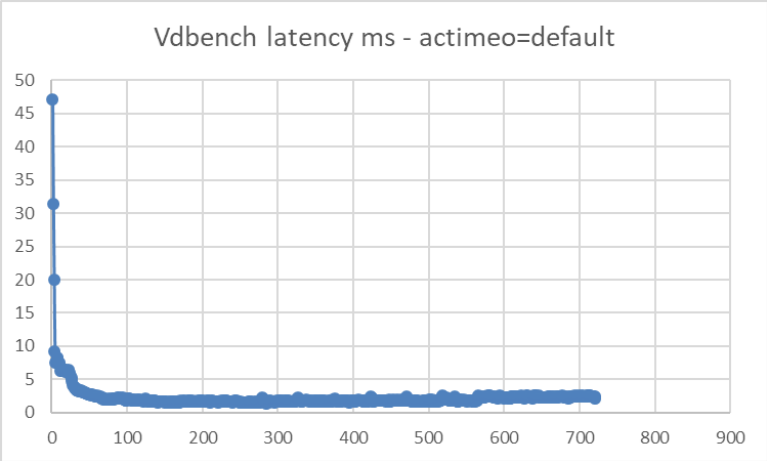
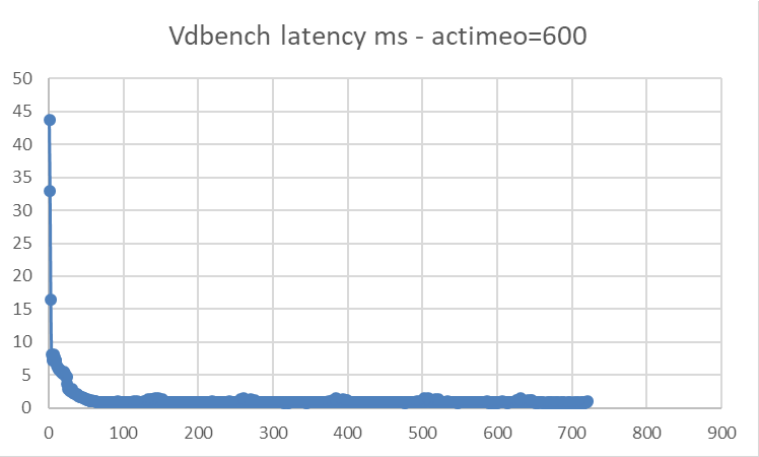


Figure 14) Actimeo=600 latency — vdbench.



The downside of setting the actimeo value too high is that attributes that change might not be reflected properly until the cache timeout occurs, which could result in unpredictable access issues.

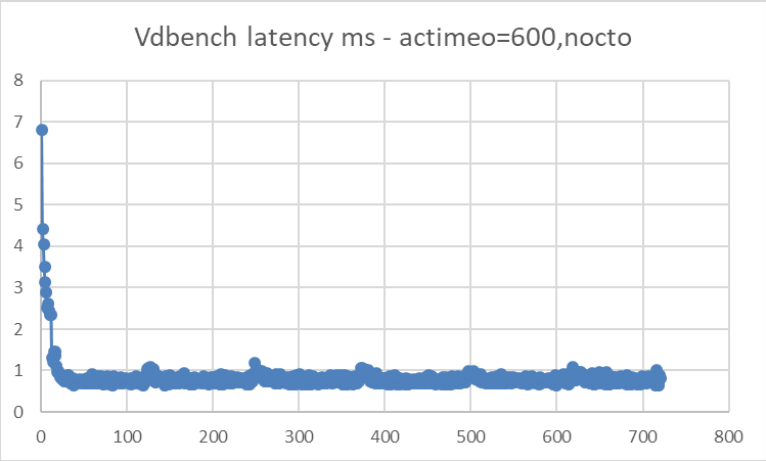
Note: The recommendation for attribute caching is to leave the defaults, unless otherwise directed by the application vendor or if testing shows a vast improvement in performance.

Nocto

Nocto stands for no close-to-open, which means that a file can close before a write has completed in order to save time. What this means in NFS environments is that other clients that have a file open for reading won't get consistent updates to that file. By default, the nocto option is not set on NFS mounts, which means that all files will wait to finish writes before allowing a close.

The nocto option is used primarily to increase raw performance. For example, in the same vdbench tests run in our Customer Proof of Concept Labs, the nocto mount option reduced latency by an additional .35ms to .7ms, as shown in Figure 15.

Figure 15) Actimeo=600, nocto latency — vdbench.



Note: The recommendation for the nocto option is to use only with read-heavy/read-mostly workloads or workloads where data is not shared between multiple systems (such as a single-writer workload).

Clientaddr

By default, the clientaddr mount option is set automatically by using the NFS client IP address. However, in some cases, this option might need to be specified in NFS mounts.

Two scenarios where it might be necessary to specify clientaddr would be:

- **Multi-NIC clients** might need to specify this option to ensure that NFS uses the desired IP address for connectivity.
- **NFSv4.x clients** might need to specify this option if two clients with the same host name (but different IP addresses) try to access NFS exports in the same SVM. NFSv4.x will send a client ID to the NFS server based on the host name. ONTAP responds with `CLID_IN_USE` and prevent the second client from mounting if it uses the same client ID. Specifying clientaddr option can force the client to increment the client ID on subsequent mount attempts.

  Note: In most instances, the clientaddr option does not need to be specified.

Nconnect

A new NFS mount option called nconnect is in its nascent stages for use with NFS mounts. The nconnect option is only available on newer Linux clients. Be sure to verify with the OS vendor documentation to determine whether the option is supported in your kernel.

The purpose of nconnect is to provide multiple transport connections per TCP connection or mount point on a client. This helps increase parallelism and performance for NFS mounts. Details about nconnect and how it can increase performance for NFS in Cloud Volumes ONTAP can be found in the blog post The Real Baseline Performance Story: NetApp Cloud Volumes Service for AWS.

ONTAP 9.8 and later offers official support for the use of nconnect with NFS mounts, provided the NFS client also supports it. To use nconnect, verify whether your client version provides it and use ONTAP 9.8 or later. ONTAP 9.8 and later supports nconnect by default with no option needed.

Note: Nconnect is not recommended for use with NFSv4.0. NFSv3, NFSv4.1, and NFSv4.2 should work fine with nconnect.

Table 15 shows results from a single Ubuntu client using different nconnect thread values.

Table 15) Nconnect performance results.

Nconnect value	Threads per process	Throughput	Difference
1	128	1.45GB/s	-
2	128	2.4GB/s	+66%
4	128	3.9GB/s	+169%
8	256	4.07GB/s	+181%

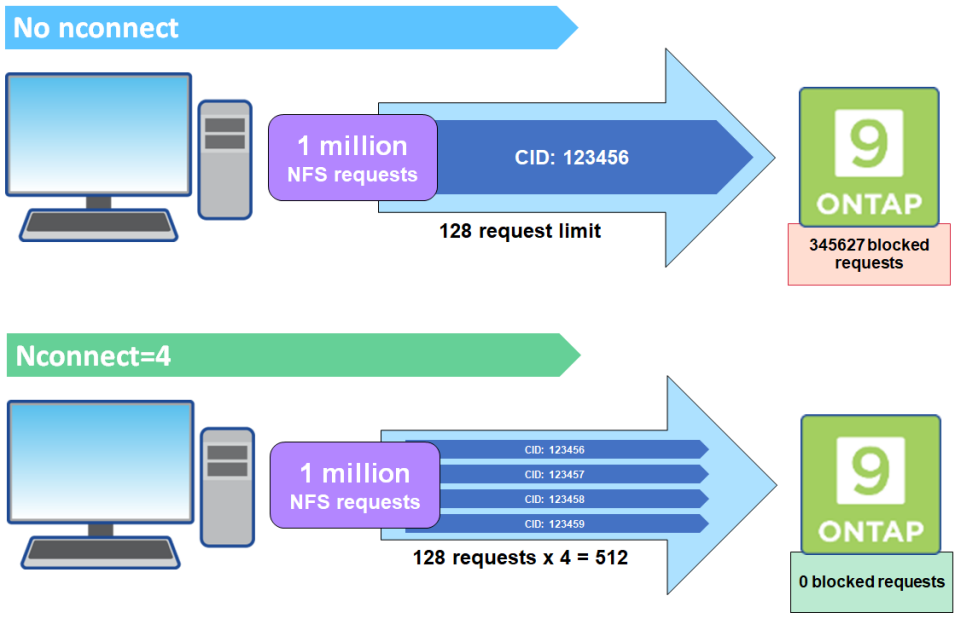
Note: The recommendation for using nconnect depends on client OS and application needs. Testing with this new option is highly recommended before deploying in production.

How can I tell nconnect is working?

Nconnect is designed to allocate more sessions across a single TCP connection. This helps to better distribute NFS workloads and add some parallelism to the connection, which helps the NFS server handle the workloads more efficiently. In ONTAP, when an NFS mount is established, a Connection ID (CID) is created. That CID provides up to 128 concurrent in-flight operations. When that number is exceeded by the client, ONTAP enacts a form of flow control until it can free up some available resources as other operations complete. These pauses usually are only a few microseconds, but over the course of millions of operations, those can add up and create performance issues. Nconnect can take the 128 limit and multiply it by the number of nconnect sessions on the client, which provides more concurrent operations per CID and can potentially add performance benefits, as seen in Table 15.

Figure 16 illustrates how mounts without nconnect handle concurrent operations and how nconnect works to distribute operations to NFS mounts.

Figure 16) NFS mounts with and without nconnect



To determine whether nconnect is indeed working in your environment, you can verify a few things.

When nconnect is not being used, a single CID is established per client mount. You can verify those cids by running the following command:

```
cluster::> network connections active show -node [nodes] -service nfs* -remote-host [hostname]
```

For example, this is the output from an active NFS connection without nconnect:

```
cluster::> network connections active show -node * -service nfs* -remote-host centos83-
perf.ntap.local
Vserver    Interface              Remote
Name       Name:Local Port        Host:Port                    Protocol/Service
---------- ---------------------- ---------------------------- ----------------
Node: node1
DEMO       data1:2049             centos83-perf.ntap.local:1013
                                                               TCP/nfs
```

When nconnect is in use, more cids per mount are present. In this example, we used nconnect=8.

```
cluster::> network connections active show -node * -service nfs* -remote-host centos83-
perf.ntap.local
Vserver    Interface              Remote
Name       Name:Local Port        Host:Port                    Protocol/Service
---------- ---------------------- ---------------------------- ----------------
Node: node1
DEMO       data1:2049             centos83-perf.ntap.local:669 TCP/nfs
DEMO       data1:2049             centos83-perf.ntap.local:875 TCP/nfs
DEMO       data1:2049             centos83-perf.ntap.local:765 TCP/nfs
DEMO       data1:2049             centos83-perf.ntap.local:750 TCP/nfs
DEMO       data1:2049             centos83-perf.ntap.local:779 TCP/nfs
DEMO       data1:2049             centos83-perf.ntap.local:773 TCP/nfs
DEMO       data1:2049             centos83-perf.ntap.local:809 TCP/nfs
DEMO       data1:2049             centos83-perf.ntap.local:897 TCP/nfs
```

Another way to determine whether nconnect is being used is through a statistics capture for the CID object. You can start the statistics for that object by running the following command:

```
cluster::> set diag
cluster::*> statistics start -object cid
```

When that object runs, it tracks the number of total allocated cids (`alloc_total`).

For example, this is the number of `alloc_total` for a mount without nconnect:

```
cluster::*> statistics show -object cid -counter alloc_total

    Counter                                                     Value
    -------------------------------- --------------------------------
    alloc_total                                                    11
```

This is from a mount with nconnect=4:

```
cluster::*> statistics show -object cid -counter alloc_total

    Counter                                                     Value
    -------------------------------- --------------------------------
    alloc_total                                                    16
```

This is the `alloc_total` from nconnect=8:

```
cluster::*> statistics show -object cid -counter alloc_total

    Counter                                                     Value
    -------------------------------- --------------------------------
    alloc_total                                                    24
```

Hard/soft

The hard or soft mount options specify 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.

Note: For business-critical NFS exports, NetApp recommends using hard mounts.

Intr/nointr

The intr option allows NFS processes to be interrupted when a mount is specified as a hard mount. This policy is deprecated in new clients such as RHEL 6.4 and is hardcoded to nointr. Kill -9 is the only way to interrupt a process in newer kernels.

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

Logging, monitoring, and statistics

The following section covers ways to view logs and set up monitoring for NFS-specific environments, as well as managing NFS operations.

Managing NFS locks

ONTAP NFS servers keep track of locks when they are established by a NAS client. If desired, storage administrators can view and break locks using the following command from Advanced Privilege.

```
cluster::*> vserver locks
break show
```

Additionally, to specify NFSv4.x locks, run the following command from Diag Privilege:

```
cluster::*> vserver locks nfsv4 show
```

NFS events: Event messaging system

ONTAP provides a messaging system that keeps track of events within the cluster. These offer summaries of informational, warning and error conditions across all subsystems.

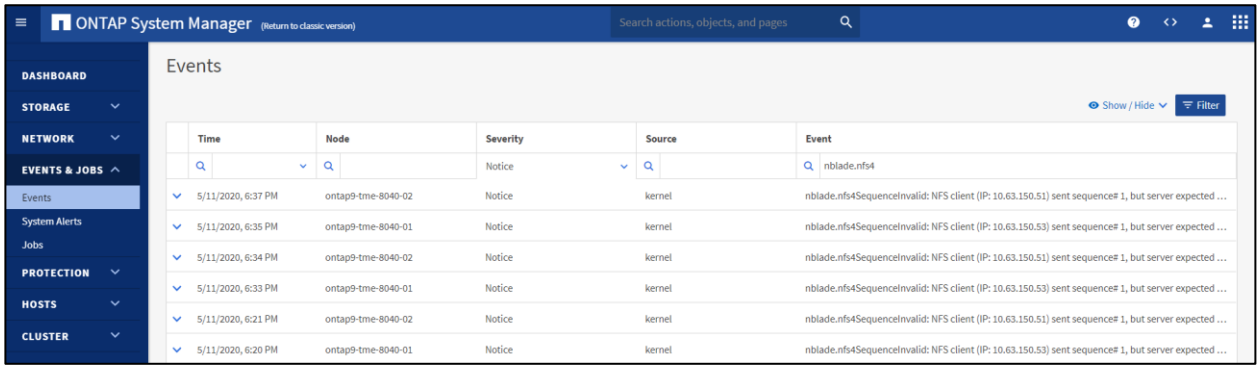
Event messaging system (EMS) logs can be viewed by using ONTAP System Manager (Figure 17), as well as from the CLI by running the following command:

```
cluster::> event log show
```

NFS events cover a wide array of subsystems, from general NFS events (such as exports or other errors) to name service and authentication issues. Event names can be filtered by portions of the event message name in the CLI and in the GUI.

```
cluster::*> event log show -message-name nblade.nfs*
Time                Node             Severity      Event
------------------- ---------------- ------------- ---------------------------
5/11/2020 18:37:50  node2
                                     NOTICE        nblade.nfs4SequenceInvalid: NFS client (IP:
x.x.x.x) sent sequence# 1, but server expected sequence# 2. Server error: OLD_STATEID.
5/11/2020 18:35:52  node1
                                     NOTICE        nblade.nfs4SequenceInvalid: NFS client (IP:
x.x.x.y) sent sequence# 1, but server expected sequence# 2. Server error: OLD_STATEID.
5/11/2020 18:34:23  node2
                                     NOTICE        nblade.nfs4SequenceInvalid: NFS client (IP:
x.x.x.x) sent sequence# 1, but server expected sequence# 2. Server error: OLD_STATEID.
```

Figure 17) Filtering events in ONTAP System Manager UI.



EMS events are triggered with a severity level, which lets you know which messages are important (such as ERROR and EMERGENCY), and which messages are simply informative (such as INFORMATIONAL and NOTICE). In general, DEBUG-level messages can be ignored unless there are other noticeable issues occurring in the environment. If you see messages ERROR, ALERT or EMERGENCY, it might be worth opening a support case.

You can filter based on severity, message name, and other variables. The following event log commands can show NFS/multiprotocol NAS-relevant EMS events:

```
cluster::> event log show -message-name dns*
cluster::> event log show -message-name *export*
cluster::> event log show -message-name ldap*
cluster::> event log show -message-name mgmt.nfs*
cluster::> event log show -message-name nameserv*
cluster::> event log show -message-name nblade*
cluster::> event log show -message-name netgroup*
cluster::> event log show -message-name *nfs*
cluster::> event log show -message-name secd*
```

NFS statistics

Statistics for NFS are collected within the ONTAP Counter Manager archives and can be viewed by using ONTAP System Manager for up to a year. Additionally, general NFS statistic captures can be done from Active IQ Performance Manager, as well as using third-party performance monitoring tools such as Grafana with NetApp Harvest.

If individual performance counters for NFS are desired, use the `statistics start` command to enable captures for a specific counter object or for multiple objects. After the statistics counters are started, they will run until you stop them. These are most useful to run during periods when performance might be suffering, or if you wish to see the workload trends for a specific volume.

Determining what type of VM is hosted on NFS

It is possible to determine what type of VM is accessing the storage by using statistics captured in the counter manager. Run the following commands to show the statistics. These are available in Diagnostic Privilege.

```
cluster::> set diag
cluster::*> statistics start -object wafl -counter wafl_nfs_application_mask
cluster::*> statistics show -object wafl -counter wafl_nfs_application_mask -raw
```

The output of these statistics shows masks for specific VM types (Table 16).

Table 16) VM statistic masks.

VM Type	Mask
None	0
ESX/ESXi	1
Citrix Xen	2
Red Hat KVM	4

If more than one VM application is being used, then the masks are added together to determine which ones are in use. For example, if ESX/ESXi and Red Hat KVM are in use, then the masks would be 1 + 4 = 5.

Determining if Oracle is in use

Additionally, the WAFL counters can determine if Oracle data is hosted on NFS.

```
cluster::> set diag
cluster::*> statistics start -object wafl -counter wafl_nfs_oracle_wcount
cluster::*> statistics show -object wafl -counter wafl_nfs_oracle_wcount -raw
```

Performance monitoring enhancements in ONTAP 9.0 and later

ONTAP 9.0 and later introduced some performance monitoring enhancements that can aid storage administrators in performance monitoring and proactive performance management.

Note: For a complete list of performance counters, objects, and instances, run the `statistics catalog` command, found at Advanced Privilege.

Top clients

ONTAP provides the ability to track the top NAS clients writing data to a cluster by using the new ONTAP 9 feature called top clients. This feature tracks the overall incoming operations and lists the client IP address, NAS protocol being used, total IOPS, node being accessed, and the SVM to which the client is connecting.

You can also watch the top clients in real time by using the command line with the admin privilege command `statistics top client show`. This command allows you to specify a minimum of 30-second intervals, the number of iterations to show, as well as the maximum number of clients that should be displayed. In the following example, a Python file create script was run from two clients to a NetApp FlexGroup volume over NFSv3 to show an example of what to expect from the command's output.

```
cluster::> statistics top client show -interval 30 -iterations 5 -max 10

cluster : 6/26/2017 17:42:27
*Estimated
     Total
      IOPS Protocol               Node Vserver Client
---------- -------- ------------------ ------- -------------
     23010      nfs              node01 DEMO   x.x.x.b
     20006      nfs              node02 DEMO   x.x.x.c
        17     cifs              node02 CIFS
                                         x.x.x.d
```

By default, the CLI orders by IOPS. It's also possible to order by throughput by using the `-sort-key` option.

For example:

```
cluster::> statistics top client show -interval 30 -iterations 5 -max 10 -sort-key write_data

cluster : 6/26/2017 18:04:53
```

```
*Estimated
     Write
Data (Bps) Protocol               Node Vserver Client
---------- -------- ------------------ ------- -------------
    154968      nfs              node01 DEMO   x.x.x.b
    150400      nfs              node02 DEMO   x.x.x.c
```

Hot files/top files

In addition to exposing the top clients accessing a cluster, ONTAP can also show the top files in use on a cluster. This can be particularly useful when dealing with ESX/virtualized environments, where a single ESX server might be hosting hundreds of VMs on a single NFS mounted datastore. In that scenario, the top clients feature would not be as useful as knowing which files are doing the most work. ONTAP can expose that information all the way down to the VMDK level.

In the following example, this ONTAP cluster is hosting several ESXi VMs on an NFS-mounted datastore. Again, we can see this information in ONTAP System Manager, as well as in the CLI.

From the CLI, the command is `statistics top file show` at Admin Privilege. In the following example, we see Active IQ Unified Manager VMDK as the top consumer of write throughput and IOPS:

```
cluster::> statistics top file show -interval 30 -iterations 1

cluster : 6/26/2017 18:01:21
*Estimated
     Total
      IOPS              Node Vserver     Volume File
---------- ----------------- ------- ---------- ----------------------------------------
        48            node03  vmware datastore1 /stme-ocum-01_1/stme-ocum-01_2-flat.vmdk
        31            node03  vmware datastore1 /OCUM722-MP/OCUM722-MP_2-flat.vmdk

cluster::> statistics top file show -interval 30 -iterations 1 -max 10 -sort-key write_data

cluster : 6/26/2017 18:05:04
*Estimated
     Write
Data (Bps)              Node Vserver     Volume File
---------- ----------------- ------- ---------- ------------------------------------------
    685600            node03  vmware datastore1 /Parisi OCUM 7.2/Parisi OCUM 7.2_2-flat.vmdk
    159475            node03  vmware datastore1 /stme-ocum-01_1/stme-ocum-01_2-flat.vmdk
```

Note: The above example is a lab environment. Results vary depending on workload.

Supported counters with top files

The top files statistic has support for the following counters:

```
Object: top_file
    Counter                     Description
    --------------------------- ----------------------------------------------
    other_ops                   Estimated Other Operations
    read_data                   Estimated Bytes From Read Operations
    read_ops                    Estimated Read Operations
    total_data                  Estimated Total Bytes From All Operations
    total_ops                   Estimated Total Operations
    write_data                  Estimated Bytes From Write Operations
    write_ops                   Estimated Write Operations
```

If you attempt to use an unsupported counter (such as latency) with `sort-key`, the following is displayed:

```
cluster::*> statistics top file show -interval 30 -iterations 1 -max 10 -sort-key latency

Error: command failed: Unsupported sort-key counter: only "selector" counters are valid for
statistically tracked objects. For a list of valid sort-key counters, use this diagnostic
command: "statistics catalog counter show -object top_file -properties *selector*"
```

Viewing storepool allocations

In some cases, you might need to view NFSv4.x storepool allocations, usually if you suspect a performance problem is due to running out of storepool resources or if you're unable to access NFSv4 mounts.

See these additional examples of issues with storepool exhaustion:

- 1051413 - Exhaustion of nblade nfsv4 storepool resources
- 1077736 - NFS storepool string exhaustion might occur while running NFSv4 traffic
- 1158309 - NFSv4 resource exhaustion might occur on storePool_StringAlloc
- 1312769 - Nfsv4.1 Mount fails when StoreTypeSessionConnectionHolder storepool exhausts

When storepool exhaustion occurs, an EMS message (Nblade.nfsV4PoolExhaust) is triggered in ONTAP 8.2.3 and later.

For further details on storepool objects, see the following:

- The Difference Between storePool and Lock Manager Objects
- Unable to access NFSv4 files due to storepool exhaustion

To view storepool objects, run the following commands:

```
cluster::> set diag
cluster::*> statistics start -object nfsv4_diag
cluster::*> statistics show -object nfsv4_diag -counter *storePool_* -raw
```

To identify an offending client, use SSH to send this command to ONTAP:

```
"set d -c off; rows 0; vserver locks nfsv4 show -inst; quit" | tee locks.txt | grep -i "client
name" | sort | uniq -c | sort -n"
```

If you suspect an issue with storepool exhaustion, contact NetApp support. In general, run the latest available ONTAP releases when using NFSv4.x.

Viewing NFS usage

You can see how many RPC calls have been issued per NFS version on a local node. The values are persistent and clear only when the node reboots. This command is available only from diagnostic privilege level and must be issued on the local node.

```
cluster::> set diag
cluster::*> system node nfs usage show
       Node: node2
                 v3: 120
                 v4: 2076
```

Viewing active NFS connections in the cluster

In ONTAP, it is possible to view active NFS connections across all SVMs and nodes in the cluster by using the `network connections active show` command. This command allows filtering of IPs, services, and other features to provide more useful and granular information. The command can be used in place of the classic `netstat` commands found in 7-Mode.

For example:

```
cluster::> network connections active show
    show           show-clients   show-lifs      show-protocols show-services

cluster::> network connections active show -node node1 -service nfs*
              Vserver   Interface         Remote
      CID Ctx Name      Name:Local Port   Host:Port            Protocol/Service
```

```
--------- --- --------- ----------------- ------------------- ----------------
Node: node1
286571835   6 vs0          data:2049         x.x.x.z:763    TCP/nfs

cluster::> network connections active show -node node2 -service nfs*
There are no entries matching your query.
```

Additionally, it is possible to view network connections in a LISTEN state with `network connections listening show`.

Mapping NFS clients to specific volumes

ONTAP 9.7 introduced a new command that allows storage administrators to see which clients are mounted through NFS to specific volumes in the cluster.

The use cases vary, but usually fall under one of these scenarios:

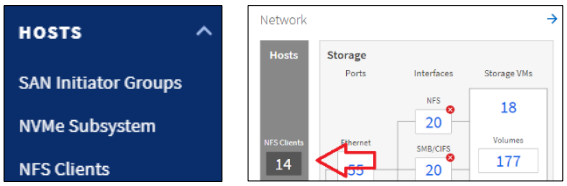
- The need to discover who's using a volume before performing migrations, cutovers, and so on
- Troubleshooting issues
- Load distribution

To view connected NFS clients:

```
cluster::> nfs connected-clients show ?
  [ -instance | -fields <fieldname>, ... ]
  [[-node] <nodename>]                                                  Node Name
  [[-vserver] <vserver>]                                                Vserver
  [[-data-lif-ip] <IP Address>]                                         Data LIF IP Address
  [[-client-ip] <IP Address>]                                           Client IP Address
  [[-volume] <volume name>]                                             Volume Accessed
  [[-protocol] <Client Access Protocol>]                                Protocol Version
  [ -idle-time <[<integer>d][<integer>h][<integer>m][<integer>s]> ]      Idle Time (Sec)
  [ -local-reqs <integer> ]                                             Number of Local Reqs
  [ -remote-reqs <integer> ]                                            Number of Remote Reqs
```

ONTAP 9.8 enables you to see these clients from ONTAP System Manager. Either click the NFS Clients link in the dashboard or navigate to Hosts → NFS Clients in the left menu.

Figure 18) Viewing NFS client to volume mappings in ONTAP System Manager



For sample output, see "NFS connected-clients sample output."

Advanced NFS concepts

This section covers NFS concepts that extend outside of the realm of basic configuration.

Umask

In NFS operations, permissions can be controlled through mode bits, which leverage numerical attributes to determine file and folder access. These mode bits determine read, write, execute, and special attributes. Numerically, these are represented as:

- Execute = 1

- Read = 2
- Write = 4

Total permissions are determined by adding or subtracting a combination of the preceding.

For example:

```
4 + 2 + 1 = 7 (can do everything)
4 + 2 = 6 (rw) and so on…
```

For more information about UNIX permissions, see UNIX Permissions Help.

Umask is a functionality that allows an administrator to restrict the level of permissions allowed to a client. By default, the umask for most clients is set to 0022, which means that files created from that client are assigned that umask. The umask is subtracted from the base permissions of the object. If a volume has 0777 permissions and is mounted using NFS to a client with a umask of 0022, objects written from the client to that volume have 0755 access (0777 – 0022).

```
# umask
0022
# umask -S
u=rwx,g=rx,o=rx
```

However, many operating systems do not allow files to be created with execute permissions, but they do allow folders to have the correct permissions. Thus, files created with a umask of 0022 might end up with permissions of 0644.

The following is an example using RHEL 6.5:

```
# umask
0022
# cd /cdot
# mkdir umask_dir
# ls -la | grep umask_dir
drwxr-xr-x.  2 root     root         4096 Apr 23 14:39 umask_dir

# touch umask_file
# ls -la | grep umask_file
-rw-r--r--.  1 root     root            0 Apr 23 14:39 umask_file
```

NFS user nfsnobody

In some cases, NFS clients might see file owner/group information show up in file listings as `nfsnobody`.

```
# ls -la | grep newfile
-rwxrwxrwx   1 nfsnobody nfsnobody          0 May 19 13:30 newfile.txt
```

When you list the file with numerics, you find that the owner:group is `65534`.

```
# ls -lan | grep newfile
-rwxrwxrwx   1 65534 65534          0 May 19 13:30 newfile.txt
```

The user `65534` on most Linux clients is `nfsnobody`, but in ONTAP, that user is `pcuser`.

```
cluster::*> unix-user show -vserver DEMO -id 65534
               User            User   Group  Full
Vserver        Name            ID     ID     Name
-------------- --------------- ------ ------ --------------------------------
DEMO           pcuser          65534  65534
```

It's also the default `anonymous` user in export-policy rules.

```
cluster::*> export-policy rule show -vserver DEMO -policyname default -fields anon
vserver policyname ruleindex anon
------- ---------- --------- -----
```

```
DEMO    default    1          65534
DEMO    default    2          65534
DEMO    default    3          65534
```

When you look at the file permissions from the ONTAP cluster, you might see that the UNIX owner is indeed `65534`, but that there are also Windows ACLs and owners that are different.

```
cluster::*> vserver security file-directory show -vserver DEMO -path /data/newfile.txt

               Vserver: DEMO
             File Path: /data/newfile.txt
     File Inode Number: 7088
        Security Style: ntfs
       Effective Style: ntfs
        DOS Attributes: 20
 DOS Attributes in Text: ---A----
Expanded Dos Attributes: -
          UNIX User Id: 65534
         UNIX Group Id: 65534
        UNIX Mode Bits: 777
 UNIX Mode Bits in Text: rwxrwxrwx
                   ACLs: NTFS Security Descriptor
                         Control:0x8004
                         Owner:NTAP\ntfs
                         Group:NTAP\DomainUsers
                         DACL - ACEs
                           ALLOW-Everyone-0x1f01ff-(Inherited)
```

When `nfsnobody` or `65534` is displayed in the NFS listings, one of two things is more than likely occurring:

- The volume that is being exported to NFS clients is also used by Windows SMB clients and the Windows users writing to the shares don't map to valid UNIX users and/or groups.
- The volume that is being exported to NFS clients has the anonymous user set to `65534` and something is causing the NFS user to squash to the anonymous user. For more information about squashing, see "The anon user."

You can view the Windows-user-to-UNIX-user mapping by running the following command in Advanced Privilege:

```
cluster::*> access-check name-mapping show -vserver DEMO -direction win-unix -name ntfs
'ntfs' maps to 'pcuser'

cluster::*> access-check name-mapping show -vserver DEMO -direction win-unix -name prof1
'prof1' maps to 'prof1'
```

NFSv4.x: nobody:nobody

One of the most common issues seen with an NFSv4.x configuration is when a file or folder is shown in a listing using `ls` as being owned by the `user:group` combination of `nobody:nobody`.

For example:

```
sh-4.2$ ls -la | grep prof1-file
-rw-r--r-- 1 nobody nobody    0 Apr 24 13:25 prof1-file
```

And the numeric ID is `99`.

```
sh-4.2$ ls -lan | grep prof1-file
-rw-r--r-- 1 99 99    0 Apr 24 13:25 prof1-file
```

On the cluster (and using NFSv3), that file ownership appears to have a proper UID/GID.

```
cluster::*> vserver security file-directory show -vserver DEMO -path /home/prof1/prof1-file
```

```
               Vserver: DEMO
             File Path: /home/prof1/prof1-file
     File Inode Number: 9996
        Security Style: unix
       Effective Style: unix
        DOS Attributes: 20
 DOS Attributes in Text: ---A----
Expanded Dos Attributes: -
          UNIX User Id: 1002
         UNIX Group Id: 10002
         UNIX Mode Bits: 644
 UNIX Mode Bits in Text: rw-r--r--
                   ACLs: -
```

In some instances, the file might show the correct owner, but `nobody` as the group.

```
sh-4.2$ ls -la | grep newfile1
-rw-r--r-- 1 prof1  nobody    0 Oct  9  2019 newfile1
```

Who is nobody anyway?

The `nobody` user in NFSv4.x is different from the `nfsnobody` user mentioned in “NFS user nfsnobody.” You can view how an NFS client sees each user by running the `id` command.

```
# id nobody
uid=99(nobody) gid=99(nobody) groups=99(nobody)
# id nfsnobody
uid=65534(nfsnobody) gid=65534(nfsnobody) groups=65534(nfsnobody)
```

With NFSv4.x, the `nobody` user is the default user defined by the `idmapd.conf` file and can be defined as any user you want to use.

```
# cat /etc/idmapd.conf | grep nobody
#Nobody-User = nobody
#Nobody-Group = nobody
```

Why does this happen?

Because security through name string mapping is a key tenet of NFSv4.x operations, the default behavior when a name string does not match properly is to squash that user to one that won’t normally have any access to files and folders owned by users and groups.

When you see `nobody` for the user and/or group in file listings, that generally means something in NFSv4.x is misconfigured. Case sensitivity can come into play here.

For example, if user1@NTAP.LOCAL (uid 1234, gid 1234) is accessing an export, then ONTAP must be able to find user1@NTAP.LOCAL (uid 1234, gid 1234). If the user in ONTAP is USER1@NTAP.LOCAL, then it won’t match. In many cases, you can see the following in the messages file on the client:

```
May 19 13:14:29 centos7 nfsidmap[17481]: nss_getpwnam: name 'root@defaultv4iddomain.com' does not
map into domain 'NTAP.LOCAL'
May 19 13:15:05 centos7 nfsidmap[17534]: nss_getpwnam: name 'nobody' does not map into domain
'NTAP.LOCAL'
```

The client and server must both agree that a user is indeed who they are claiming to be, so the following should be checked to ensure that the user that the client sees has the same information as the user that ONTAP sees.

- NFSv4.x ID domain (Client: `idmapd.conf` file; ONTAP: `-v4-id-domain` option)
- User name and numeric IDs (name service switch configuration – client: `nsswitch.conf` and/or local passwd and group files; ONTAP: `ns-switch` commands)

- Group name and numeric IDs (Name service switch configuration – client: `nsswitch.conf` and/or local passwd and group files; ONTAP: `ns-switch` commands)

In almost all cases, if you see `nobody` in user and group listings from clients but ONTAP reports the correct user and group information (through vserver security file-directory show), the issue is user or group name domain ID translation.

You can also make use of the ONTAP option `-v4-numeric-ids`, covered in "Bypassing the name string — Numeric IDs."

Viewing name ID strings for NFSv4.x on clients

If you are using NFSv4.x, there is a name-string mapping that takes place during NFS operations. If the name string doesn't match, then you see issues as described in the section "NFSv4.x: nobody:nobody."

In addition to using `/var/log/messages` to find an issue with NFSv4 IDs, you can use the `nfsidmap -l` command on the NFS client to view which user names have properly mapped to the NFSv4 domain.

For example, this is output of the command after a user that exists in both the client and ONTAP SVM accesses an NFSv4.x mount:

```
# nfsidmap -l
4 .id_resolver keys found:
  gid:daemon@CENTOS-LDAP.LOCAL
  uid:nfs4@CENTOS-LDAP.LOCAL
  gid:root@CENTOS-LDAP.LOCAL
  uid:root@CENTOS-LDAP.LOCAL
```

When a user that does not map properly into the NFSv4 ID domain (in this case, `netapp-user`) tries to access the same mount and touches a file, they get assigned `nobody:nobody`, as expected.

```
# su netapp-user
sh-4.2$ id
uid=482600012(netapp-user), 2000(secondary)
sh-4.2$ cd /mnt/nfs4/
sh-4.2$ touch newfile
sh-4.2$ ls -la
total 16
drwxrwxrwx  5 root   root   4096 Jan 14 17:13 .
drwxr-xr-x. 8 root   root     81 Jan 14 10:02 ..
-rw-r--r--  1 nobody nobody    0 Jan 14 17:13 newfile
drwxrwxrwx  2 root   root   4096 Jan 13 13:20 qtree1
drwxrwxrwx  2 root   root   4096 Jan 13 13:13 qtree2
drwxr-xr-x  2 nfs4   daemon 4096 Jan 11 14:30 testdir
```

The `nfsidmap -l` output shows the user `pcuser` in the display, but not `netapp-user`; this is the anonymous user in our export-policy rule (`65534`).

```
# nfsidmap -l
6 .id_resolver keys found:
  gid:pcuser@CENTOS-LDAP.LOCAL
  uid:pcuser@CENTOS-LDAP.LOCAL
  gid:daemon@CENTOS-LDAP.LOCAL
  uid:nfs4@CENTOS-LDAP.LOCAL
  gid:root@CENTOS-LDAP.LOCAL
  uid:root@CENTOS-LDAP.LOCAL
```

Hiding Snapshot copies

A NetApp Snapshot copy is a read-only, point-in-time (PiT) image of a volume. The image consumes minimal storage space and incurs negligible performance overhead because it records only changes to files since the last Snapshot copy was made.

Users can access Snapshot copies to restore individual files from NFS clients. In some workloads however, applications might list the contents of the NFS mount. In some cases, the `.snapshot` directory gets included, which can add considerable time to these scans – especially in high file count environments. In those cases, the Snapshot directory can be hidden from NFS mounts using the NFS server option `-v3-hide-snapshot`.

As evidenced by the command, this only impacts NFSv3 clients, as the `.snapshot` directory is already hidden for NFSv4.x.

Even when the `.snapshot` directory is hidden, it is still accessible. To remove access to the `.snapshot` directory to clients, use the volume option `-snapdir-access`.

Note: These options don't take effect until the export is remounted.

Viewing and managing NFS credentials

In ONTAP 9.3, a global cache for name services was implemented to offer better performance, reliability, resilience, and supportability for NAS credentials and name service server management.

Part of those changes was the implementation of the NFS credential cache, which stores NFS user and group information in ONTAP when NFS exports are accessed.

These caches can be viewed and managed through the Advanced Privilege `nfs credentials` commands.

```
cluster::*> nfs credentials ?
  count                       *Count credentials cached by NFS
  flush                       *Flush credentials cached by NFS
  show                        *Show credentials cached by NFS
```

Cache entries will populate the node where the TCP connection for the NFS mount exists. This information can be seen through the following command on the cluster:

```
cluster::*> nfs connected-clients show -vserver DEMO -client-ip x.x.x.x -fields data-lif-ip -
volume scripts
node               vserver data-lif-ip   client-ip     volume  protocol
------------------ ------- ------------- ------------- ------- --------
Node1              DEMO    x.x.x.y       x.x.x.x       scripts nfs3
```

From the command above, we know the client IP `x.x.x.x` is connected to a data LIF on node1. That helps us narrow down which node to focus on for cache entries.

The `nfs credentials count` command allows you to see how many credentials are currently stored in the NFS credential cache. This can be useful in understanding the impact of clearing the cache.

```
cluster::*> nfs credentials count -node node1
Number of credentials cached by NFS on node "node1": 4
```

If a user traverses into an ONTAP NFS export, user IDs, group IDs, and so on are all added to the NFS credential cache. For example, we have a user named `prof1`.

```
# id prof1
uid=1102(prof1) gid=10002(ProfGroup) groups=10002(ProfGroup),10000(Domain
Users),1202(group2),1101(group1),1220(sharedgroup),1203(group3)
```

That user has eight different entries – a numeric UID and seven group memberships. Then, the user `prof1` accesses an NFS export. Our credential cache increases by eight.

```
cluster::*> nfs credentials count -node node1
Number of credentials cached by NFS on node "node1": 12
```

That count is for the entire node – not just per SVM. If you have multiple SVMs in your environment, the count might not be useful for troubleshooting.

Viewing the NFS credential cache

In addition to showing how many credentials are in the NFS credential cache, we can also view individual cache entries for users and/or groups. If a user in your environment complains about having access issues, you can look for that user in the cache.

Note: You cannot view the contents of entire credential cache.

In this example, `prof1` accessed the mount. You can see that cache entry, as well as the flags that tell us more about the cache entry.

```
cluster::*> nfs credentials show -node node1 -vserver DEMO -unix-user-name prof1

Credentials
-----------
                    Node: node1
                 Vserver: DEMO
               Client IP: -
                   Flags: unix-extended-creds-present, id-name-mapping-present
  Time since Last Refresh: 52s
   Time since Last Access: 44s
               Hit Count: 4

  UNIX Credentials:
                   Flags: 1
               Domain ID: 0
                     UID: 1102
             Primary GID: 10002
         Additional GIDs: 10002
                          10000
                          1101
                          1202
                          1203
                          1220

  Windows Credentials:
                   Flags: -
                User SID: -
       Primary Group SID: -
             Domain SIDs: -

  ID-Name Information:
                    Type: user
                      ID: 1102
                    Name: prof1
```

You can also see an entry for the user's primary group.

```
cluster::*> nfs credentials show -node node1 -vserver DEMO -unix-group-name ProfGroup

Credentials
-----------
                    Node: node1
                 Vserver: DEMO
               Client IP: -
                   Flags: id-name-mapping-present
  Time since Last Refresh: 64s
   Time since Last Access: 6s
               Hit Count: 2

  UNIX Credentials:
                   Flags: -
               Domain ID: -
                     UID: -
             Primary GID: -
         Additional GIDs: -

  Windows Credentials:
                   Flags: -
```

```
                User SID: -
        Primary Group SID: -
              Domain SIDs: -

  ID-Name Information:
                     Type: group
                       ID: 10002
                     Name: ProfGroup
```

You can also view credential cache entries for users and groups down to the client IP that tried the access.

```
cluster::*> nfs credentials show -node node1 -vserver DEMO -client-ip x.x.x.x -unix-user-id 1102

Credentials
-----------
                         Node: node1
                      Vserver: DEMO
                    Client IP: x.x.x.x
                        Flags: unix-extended-creds-present, id-name-mapping-present
       Time since Last Refresh: 35s
        Time since Last Access: 34s
                    Hit Count: 2
               Reference Count: 4
 Result of Last Update Attempt: no error

  UNIX Credentials:
                     Flags: 1
                 Domain ID: 0
                       UID: 1102
               Primary GID: 10002
          Additional GIDs: 10002
                           10000
                           1101
                           1202
                           1203
                           1220

  Windows Credentials:
                    Flags: -
                 User SID: -
        Primary Group SID: -
              Domain SIDs: -

  ID-Name Information:
                     Type: user
                       ID: 1102
                     Name: prof1
```

The credential cache also keeps negative entries (entries that could not be resolved) in cache. Negative entries occur when ONTAP can't resolve the numeric UID to a valid user. In this case, the UID 1236 cannot be resolved by ONTAP, but attempted access to the NFS export.

```
# su cifsuser
bash-4.2$ cd /scripts/
bash: cd: /scripts/: Permission denied
bash-4.2$ id
uid=1236(cifsuser) gid=1236(cifsuser) groups=1236(cifsuser)

cluster::*> nfs credentials show -node node1 -vserver DEMO -unix-user-id 1236

Credentials
-----------
                     Node: node1
                  Vserver: DEMO
                Client IP: -
                    Flags: no-unix-extended-creds, no-id-name-mapping
  Time since Last Refresh: 33s
   Time since Last Access: 7s
```

```
               Hit Count: 15

  UNIX Credentials:
                     Flags: -
                 Domain ID: -
                       UID: -
              Primary GID: -
           Additional GIDs: -

  Windows Credentials:
                     Flags: -
                  User SID: -
        Primary Group SID: -
              Domain SIDs: -

  ID-Name Information:
                     Type: -
                       ID: -
                     Name: -
```

NFS credential cache with NFSv4.x and multiprotocol NAS

The NFS credential cache entries also store Windows credentials and NFSv4 ID mapping credentials.

When a user traverses NFSv4.x exports and maps into the ID domain correctly, we'd see the `ID-Name Information` field populated:

```
Credentials
-----------
                          Node: node
                       Vserver: DEMO
                     Client IP: x.x.x.x
                         Flags: unix-extended-creds-present, id-name-mapping-present
       Time since Last Refresh: 12s
        Time since Last Access: 9s
                     Hit Count: 2
                Reference Count: 4
 Result of Last Update Attempt: no error

  UNIX Credentials:
                     Flags: 1
                 Domain ID: 0
                       UID: 1102
              Primary GID: 10002
           Additional GIDs: 10002
                            10000
                            1101
                            1202
                            1203
                            1220

  Windows Credentials:
                     Flags: -
                  User SID: -
        Primary Group SID: -
              Domain SIDs: -

  ID-Name Information:
                     Type: user
                       ID: 1102
                     Name: prof1
```

If the user accesses an export that has NTFS permissions/security style, we'd see the flag `cifs-creds-present`, as well as the domain SID information under `Windows Credentials`:

```
Credentials
-----------
                          Node: node1
                       Vserver: DEMO
```

```
                    Client IP: x.x.x.x
                        Flags: ip-qualifier-configured, unix-extended-creds-present, cifs-creds-
present
       Time since Last Refresh: 19s
        Time since Last Access: 1s
                    Hit Count: 9
                Reference Count: 2
 Result of Last Update Attempt: no error

  UNIX Credentials:
                       Flags: 0
                   Domain ID: 0
                         UID: 1102
                Primary GID: 10002
           Additional GIDs: 10002
                            10000
                            1101
                            1202
                            1203
                            1220

  Windows Credentials:
                       Flags: 8320
                  User SID: S-1-5-21-3552729481-4032800560-2279794651-1214
         Primary Group SID: S-1-5-21-3552729481-4032800560-2279794651-513
                Domain SIDs: S-1-5-21-3552729481-4032800560-2279794651
                            S-1-18
                            S-1-1
                            S-1-5
                            S-1-5-32

  ID-Name Information:
                       Type: -
                         ID: -
                       Name: -
```

NFS credential cache settings

The timeout value of the NFS credential cache is controlled by the NFS server options listed in Table 17.

Table 17) NFS credential cache settings.

Option	What it does	Default value (ms)
`-cached-cred-negative-ttl`	This optional parameter specifies the age of the negative cached credentials after which they will be cleared from the cache. The value specified must be between 60000 and 604800000.	7200000 (2 hours)
`-cached-cred-positive-ttl`	This optional parameter specifies the age of the positive cached credentials after which they will be cleared from the cache. The value specified must be between 60000 and 604800000.	86400000 (24 hours)
`-cached-cred-harvest-timeout`	This optional parameter specifies the harvest timeout for cached credentials. The value specified must be between 60000 and 604800000.	86400000 (24 hours)

Cache entries maintain the time since last access/refresh (as seen in the `show` command). If an entry stays idle for a period of time, it is eventually removed from the cache. If the entry is active, it is refreshed and stays in cache.

These values can be modified to longer or shorter timeout values, depending on the desired effects:

- **Longer cache timeout values** reduce network load and provide faster lookups of users but can produce more false positives/false negatives as the cache entries are not always in sync with name services.
- **Shorter cache timeout values** increase load on the network and name servers and can add some latency to name lookups (depending on name service source) but offer more accurate and up-to-date entries.

The best practice is to leave the values as is. If you need to change the values, be sure to monitor the results and adjust as needed.

Flushing the NFS credential cache

In cases where a user has been added or removed from a group and does not have the desired access, the credential cache entry can be flushed manually, rather than waiting for the cache entry to timeout.

The command can be run for a UNIX user or numeric ID or a UNIX group or numeric ID. Additionally, the command can be run as granularly as down to the client IP address having the issue.

```
cluster::*> nfs credentials flush -node node1 -vserver DEMO -client-ip x.x.x.x -unix-user-id 1102

Number of matching credentials flushed: 2
```

Note: You can only flush one NFS credential cache entry at a time.

The NFS credential cache is separate from the name service cache. For information about managing the name service caches, see TR-4835: How to Configure LDAP in ONTAP.

Using NFSv4.x ACLs with NFSv3 mounts

NFSv3, by default, has a fairly limited way to manage permissions. However, it is possible in ONTAP to set NFSv4.x ACLs on files and folders and have them apply to NFSv3 exports.

The method for doing this is straightforward:

1. Configure and enable NFSv4.x in ONTAP and on your client.
2. Enable NFSv4.x ACL support in ONTAP.
3. Mount the export via NFSv4.x.
4. Apply the NFSv4.x ACLs.
5. Unmount and then remount the export using NFSv3 and test it out.

In the example environment, we mounted a homedir volume and then set up an ACL on a file owned by root for a user called `prof1` by using `nfs4_setfacl -e` (which allows you to edit a file rather than have to type in a long command).

The file lives in the root user's homedir. The root homedir is set to 755, which means anyone can read them, but no one but the owner (root) can write to them.

```
drwxr-xr-x 2 root root 4096 Jul 13 10:42 root
```

That is, unless we set NFSv4.x ACLs to allow a user full control.

```
[root@centos7 mnt]# nfs4_getfacl /mnt/root/file
A::prof1@ntap.local:rwaxtTnNcCy
A::OWNER@:rwaxtTnNcCy
A:g:GROUP@:rxtncy
A::EVERYONE@:rxtncy
```

We can also see those permissions from the ONTAP CLI.

```
cluster::*> vserver security file-directory show -vserver DEMO -path /home/root/file
```

```
Vserver: DEMO
 File Path: /home/root/file
 File Inode Number: 8644
 Security Style: unix
 Effective Style: unix
 DOS Attributes: 20
 DOS Attributes in Text: ---A----
Expanded Dos Attributes: -
 UNIX User Id: 0
 UNIX Group Id: 1
 UNIX Mode Bits: 755
 UNIX Mode Bits in Text: rwxr-xr-x
 ACLs: NFSV4 Security Descriptor
 Control:0x8014
 DACL - ACEs
 ALLOW-user-prof1-0x1601bf
 ALLOW-OWNER@-0x1601bf
 ALLOW-GROUP@-0x1200a9-IG
 ALLOW-EVERYONE@-0x1200a9
```

In the above example, we provided `prof1` full control over the file. We then mounted through NFSv3. When you become a user that isn't on the NFSv4.x ACL, you can't write to the file or remove the file (the expected behavior).

```
[root@centos7 /]# su student1
sh-4.2$ cd /mnt/root
sh-4.2$ ls -la
total 8
drwxr-xr-x 2 root root 4096 Jul 13 10:42 .
drwxrwxrwx 11 root root 4096 Jul 10 10:04 ..
-rwxr-xr-x 1 root bin 0 Jul 13 10:23 file
-rwxr-xr-x 1 root root 0 Mar 29 11:37 test.txt

sh-4.2$ touch file
touch: cannot touch 'file': Permission denied
sh-4.2$ rm file
rm: remove write-protected regular empty file 'file'? y
rm: cannot remove 'file': Permission denied
```

When you change to the `prof1` user, you have access to do whatever you want, even though the mode bit permissions in v3 say you should not be able to. That is because the NFSv4.x ACL is working:

```
[root@centos7 /]# su prof1
sh-4.2$ cd /mnt/root
sh-4.2$ ls -la
total 8
drwxr-xr-x 2 root root 4096 Jul 13 10:42 .
drwxrwxrwx 11 root root 4096 Jul 10 10:04 ..
-rwxr-xr-x 1 root bin 0 Jul 13 10:23 file
-rwxr-xr-x 1 root root 0 Mar 29 11:37 test.txt

sh-4.2$ vi file
sh-4.2$ cat file
NFSv4ACLS!
```

When you do a `chmod`, however, nothing seems to change from the NFSv4 ACL for the user. We set 700 on the file, which showed up in NFSv3 mode bits.

```
sh-4.2$ chmod 700 file
sh-4.2$ ls -la
total 8
drwxr-xr-x 2 root root 4096 Jul 13 10:42 .
drwxrwxrwx 11 root root 4096 Jul 10 10:04 ..
-rwx------ 1 root bin 11 Aug 11 09:58 file
-rwxr-xr-x 1 root root 0 Mar 29 11:37 test.txt
```

But notice how the `prof1` user still has full control.

```
cluster::*> vserver security file-directory show -vserver DEMO -path /home/root/file

Vserver: DEMO
 File Path: /home/root/file
 File Inode Number: 8644
 Security Style: unix
 Effective Style: unix
 DOS Attributes: 20
 DOS Attributes in Text: ---A----
Expanded Dos Attributes: -
 UNIX User Id: 0
 UNIX Group Id: 1
 UNIX Mode Bits: 700
 UNIX Mode Bits in Text: rwx------
 ACLs: NFSV4 Security Descriptor
 Control:0x8014
 DACL - ACEs
 ALLOW-user-prof1-0x1601bf
 ALLOW-OWNER@-0x1601bf
 ALLOW-GROUP@-0x120088-IG
 ALLOW-EVERYONE@-0x120088
```

That is because the NFSv4.x ACL preserve is enabled. If that option is disabled, `chmod` wipes the ACL.

Commands to troubleshoot permission issues

In most cases, NFS permission issues are fairly straightforward; NFSv3 uses basic RWX mode bits. However, things can get complicated when NFSv4 ACLs and/or multiprotocol NAS access and different security styles are involved. This section intends to show some useful commands for troubleshooting permissions issues in NAS environments. For name service cache information, see the section “Viewing and managing NFS credentials.” For more detailed information, see TR-4835: LDAP in NetApp ONTAP.

Verifying UNIX UIDs and group memberships

For NFSv3 operations, UNIX user names and group names aren’t hugely important, as the numeric can be passed to verify identity. However, with NFSv4 and NTFS security style objects, numeric IDs need to translate to valid UNIX user and group names for proper name resolution. For NFSv4, this is needed to avoid squashing a user to nobody. In NTFS security styles, the UNIX user name is needed to map to a valid Windows user name.

In ONTAP, there are a few commands you can use to see a UNIX user’s ID and group memberships.

For local UNIX users and groups, run the following commands:

```
cluster::> unix-user show
cluster::> unix-group show
```

For basic UID/GID information for all UNIX users (local and name services; Advanced Privilege), run the following command:

```
cluster::*> access-check authentication show-ontap-admin-unix-creds
```

Or:

```
cluster::*> getxxbyyy getpwbyname -node node1 -vserver DEMO -user name prof1 -show-source true
  (vserver services name-service getxxbyyy getpwbyname)
Source used for lookup: LDAP
pw_name: prof1
pw_passwd:
pw_uid: 1102
pw_gid: 10002
pw_gecos:
pw_dir:
pw_shell:
```

```
cluster::*> getxxbyyy getpwbyname -node node1 -vserver DEMO -user name host -show-source true
  (vserver services name-service getxxbyyy getpwbyname)
Source used for lookup: Files
pw_name: host
pw_passwd: *
pw_uid: 598
pw_gid: 0
pw_gecos:
pw_dir:
pw_shell:
```

To view user information and group memberships (local and name services; Advanced Privilege), run the following commands:

```
cluster::*> getxxbyyy getgrlist -node node1 -vserver DEMO -user name prof1
  (vserver services name-service getxxbyyy getgrlist)
pw_name: prof1
Groups: 10002 10002 10000 1101 1202 1203 48
```

Viewing user and group information for multiprotocol users

If you have both CIFS/SMB and NFS configured in your environment, you can get a full list of user names, name mapping, IDs, group names, privileges, and group memberships from a single command in Advanced Privilege. This is the preferred command to use in multiprotocol environments. The command does not work if there are no SMB/CIFS servers configured.

```
cluster::*> access-check authentication show-creds -node node1 -vserver DEMO -unix-user-name
prof1 -list-name true -list-id true
  (vserver services access-check authentication show-creds)

 UNIX UID: 1102 (prof1) <> Windows User: S-1-5-21-3552729481-4032800560-2279794651-1110
(NTAP\prof1 (Windows Domain User))

 GID: 10002 (ProfGroup)
 Supplementary GIDs:
  10002  (ProfGroup)
  10000  (Domain Users)
  1101  (group1)
  1202  (group2)
  1203  (group3)
  48  (apache-group)

 Primary Group SID: S-1-5-21-3552729481-4032800560-2279794651-1111   NTAP\ProfGroup (Windows
Domain group)

 Windows Membership:
  S-1-5-21-3552729481-4032800560-2279794651-1301   NTAP\apache-group (Windows Domain group)
  S-1-5-21-3552729481-4032800560-2279794651-1106   NTAP\group2 (Windows Domain group)
  S-1-5-21-3552729481-4032800560-2279794651-513   NTAP\DomainUsers (Windows Domain group)
  S-1-5-21-3552729481-4032800560-2279794651-1105   NTAP\group1 (Windows Domain group)
  S-1-5-21-3552729481-4032800560-2279794651-1107   NTAP\group3 (Windows Domain group)
  S-1-5-21-3552729481-4032800560-2279794651-1111   NTAP\ProfGroup (Windows Domain group)
  S-1-5-21-3552729481-4032800560-2279794651-1231   NTAP\local-group.ntap (Windows Alias)
  S-1-18-2   Service asserted identity (Windows Well known group)
  S-1-5-32-551   BUILTIN\Backup Operators (Windows Alias)
  S-1-5-32-544   BUILTIN\Administrators (Windows Alias)
  S-1-5-32-545   BUILTIN\Users (Windows Alias)
 User is also a member of Everyone, Authenticated Users, and Network Users

 Privileges (0x22b7):
  SeBackupPrivilege
  SeRestorePrivilege
  SeTakeOwnershipPrivilege
  SeSecurityPrivilege
  SeChangeNotifyPrivilege
```

Showing file permissions as seen by ONTAP

When troubleshooting permissions issues, you might not have access to view permissions from a NAS client. Or you might want to verify what the NAS client is seeing for permissions with what ONTAP is seeing. To do that, you can run the following command:

```
cluster::> file-directory show -vserver DEMO -path /home/prof1
  (vserver security file-directory show)

               Vserver: DEMO
             File Path: /home/prof1
     File Inode Number: 8638
        Security Style: ntfs
       Effective Style: ntfs
        DOS Attributes: 10
 DOS Attributes in Text: ----D---
Expanded Dos Attributes: -
          UNIX User Id: 0
         UNIX Group Id: 0
        UNIX Mode Bits: 777
 UNIX Mode Bits in Text: rwxrwxrwx
                   ACLs: NTFS Security Descriptor
                         Control:0x8504
                         Owner:NTAP\prof1
                         Group:BUILTIN\Administrators
                         DACL - ACEs
                           ALLOW-Everyone-0x1f01ff-OI|CI
                           ALLOW-NTAP\prof1-0x1f01ff-OI|CI
                           ALLOW-NTAP\sharedgroup-0x1200a9-OI|CI
                           ALLOW-NTAP\Administrator-0x1f01ff-OI|CI
```

You can also verify which effective permissions a specific user has to a specific file or directory.

```
cluster::> file-directory show-effective-permissions -vserver DEMO -unix-user-name prof1 -path
/home/prof1
  (vserver security file-directory show-effective-permissions)

                       Vserver: DEMO
             Windows User Name: NTAP\prof1
                Unix User Name: prof1
                     File Path: /home/prof1
               CIFS Share Path: -
         Effective Permissions:
                                Effective File or Directory Permission: 0x1f01ff
                                       Read
                                       Write
                                       Append
                                       Read EA
                                       Write EA
                                       Execute
                                       Delete Child
                                       Read Attributes
                                       Write Attributes
                                       Delete
                                       Read Control
                                       Write DAC
                                       Write Owner
                                       Synchronize
```

Checking export policy access

In some cases, permissions issues might be due to the export policy settings. For example, if your policy is set to allow only reads, then that might override any user permissions set on the mount.

ONTAP provides a way to check the export policy access for the client by running the following command:

```
cluster::> export-policy check-access
```

Using security tracing

If you want to trace permissions issues as they occur, you can use the security trace filter functionality to trace both NFS and SMB/CIFS permissions.

To create a trace filter, run the following command:

```
cluster::> vserver security trace filter create ?
    -vserver <vserver name>         Vserver
   [-index] <integer>               Filter Index
  [[-protocols] {cifs|nfs}, ...]    Protocols (default: cifs)
  [ -client-ip <IP Address> ]       Client IP Address to Match
  [ -path <TextNoCase> ]            Path
  { [ -windows-name <TextNoCase> ]  Windows User Name
  | [ -unix-name <TextNoCase> ] }   UNIX User Name or User ID
  [ -trace-allow {yes|no} ]         Trace Allow Events (default: no)
  [ -enabled {enabled|disabled} ]   Filter Enabled (default: enabled)
  [ -time-enabled {1..720} ]        Minutes Filter is Enabled (default: 60)
```

If you desire, you can narrow the trace down to specific user names or IP addresses.

```
cluster::*> vserver security trace filter modify -vserver DEMO -index 1 -protocols nfs -client-ip
x.x.x.x -trace-allow yes -enabled enabled
```

After the trace is created, you can see results in real time. When you view the results, you can filter by successes, failures, user IDs, protocol, and more.

```
cluster::> vserver security trace trace-result show ?
  [ -instance | -fields <fieldname>, ... ]
  [[-node] <nodename>]                  Node
  [ -vserver <vserver name> ]           Vserver
  [[-seqnum] <integer>]                 Sequence Number
  [ -keytime <Date> ]                   Time
  [ -index <integer> ]                  Index of the Filter
  [ -client-ip <IP Address> ]           Client IP Address
  [ -path <TextNoCase> ]                Path of the File Being Accessed
  [ -win-user <TextNoCase> ]            Windows User Name
  [ -security-style <security style> ]  Effective Security Style On File
  [ -result <TextNoCase> ]              Result of Security Checks
  [ -unix-user <TextNoCase> ]           UNIX User Name
  [ -session-id <integer> ]             CIFS Session ID
  [ -share-name <TextNoCase> ]          Accessed CIFS Share Name
  [ -protocol {cifs|nfs} ]              Protocol
  [ -volume-name <TextNoCase> ]         Accessed Volume Name
```

Here is an example of what a permission/access failure looks like for a specific user:

```
cluster::> vserver security trace trace-result show -node * -vserver DEMO -unix-user 1102 -result
*denied*

Vserver: DEMO


Node            Index Filter Details             Reason
--------------- ----- -------------------------- ------------------------------
Node2           1     Security Style: UNIX and   Access is denied. The
                      NFSv4 ACL                  requested permissions are not
                                                 granted by the ACE while
                                                 setting attributes. Access is
                                                 not granted for: "Write DAC"
                      Protocol: nfs
                      Volume: home
                      Share: -
                      Path: /dir
                      Win-User: -
```

```
                    UNIX-User: 1102
                    Session-ID: -
```

Dynamic NAS TCP autotuning

ONTAP uses dynamic NAS TCP autotuning, which enables the NAS protocol stack to adjust buffer sizes on the fly to the most optimal setting. This capability is needed because static methods to set TCP buffer sizes do not consider the dynamic nature of networks nor the range of different types of connections made to a system at one time. Autotuning is used to optimize the throughput of NAS TCP connections by computing the application data read rate and the rate of the data being received by the system to compute optimal buffer size. The feature is not configurable and only increases buffer sizes; buffers never decrease. The starting value for this is 32K. Autotuning applies to individual TCP connections, rather than on a global scale.

This is not the same as the maximum transfer size value (`-tcp-max-xfer-size`) included under the NFS server options.

Maximum transfer size settings in ONTAP 9 and later

In NetApp ONTAP 9 and later, `-v3-tcp-max-read-size` and `-v3-tcp-max-write-size` have been deprecated. The NetApp recommendation is to leverage the option `-tcp-max-xfer-size` instead. This change also allows TCP transfer sizes of 1MB for both reads and writes. ONTAP versions prior to ONTAP 9 only allowed 1MB for reads.

Note: If these values are adjusted, they affect only new mounts. Existing mounts maintain the block size that was set at the time of the mount. If the sizes are changed, existing mounts can experience rejections of write operations or smaller responses for reads than requested. Whenever you change block size options, make sure that clients are unmounted and remounted to reflect those changes. For more information, see bug 962596.

Why dynamic Window size?

Most environments do not benefit from static TCP window sizes, because window sizes are generally considered in the context of a single host or connection. On a server, such as NFS running on ONTAP, there are multiple connections to multiple hosts. Each connection has its own uniqueness and requires varying degrees of throughput. With a static window, a server becomes extremely limited in how it can handle the diversity of inbound connections. Participants in network infrastructures often change and rarely are static; thus the TCP stack needs to be able to handle those participants in an efficient and effective manner. Dynamic window sizes help prevent the caveats seen in static window environments, such as overutilizing a network and creating a throughput collapse or underutilizing a network and experiencing less operating efficiency over time.

Exec context throttling

When an NFS operation is sent from a client, ONTAP reserves a resource on the node as a placeholder. This operation called an execution context (exec context). When the operation is finished, the exec context is freed back to the system for use.

These resources are finite within a node and depend on factors such as the ONTAP version and platform/memory. You can see the per-node limit of these values by running the following command:

```
cluster::> set diag
cluster::*> systemshell -node node1 -command "sysctl -a | grep preallocated"
```

Table 18 shows some examples of per-node values for available exec contexts.

Table 18) Exec contexts per node.

Node type	ONTAP version	Total preallocated exec contexts
AFF8040	9.8	1500
AFF A300	9.9.1	3000
AFF A800	9.8	10000
FAS9000	9.9.1	10000

In addition to per-node limits, there is also a limit of 128 concurrent operations (assigned exec contexts) per TCP CID. If a client sends more than 128 concurrent operations, ONTAP blocks those operations until a new resource is freed up. By default, Linux clients are configured to send up to 65,536 of these operations concurrently, so the limits can start to be reached relatively quickly.

The implications of this process are covered in more detail in “Identifying potential issues with RPC slot tables.”

In some cases, a single client can be overwhelming a node’s WAFL layer with requests to the same volume, which then increases latency in WAFL that increases the amount of time it takes to free up exec contexts and release them back to the system. Thus, reducing the total number of exec contexts available to other workloads connecting to the same node. This is commonly seen in grid computing applications, where many clients are sending NFS operations to the same volumes at once. For example, if every NFS client is sending 128 concurrent operations at a time, then it would only take 24 clients to overwhelm the limits.

ONTAP 9.9.1 introduces a new feature for platforms with >256GB of RAM that helps limit the impact that bully workloads have on other workloads in the system when they send more concurrent operations than supported by ONTAP. This new feature works by throttling back the number of available exec contexts for all connections to prevent one workload from overrunning others. This throttling is based on the total utilization of exec contexts on the node and helps scale back the operations to ensure the node totals don’t get exceeded.

Table 19) Exec context throttle scale.

Node utilization of exec contexts	Scale-back factor	Per-connection exec limit
60%	1	128
70%	8	16
80%	16	8

After a node hits 70% of the total available exec contexts, each connection is only able to perform 16 concurrent operations until the node’s total utilization drops back to 60%. The exec limit then increases back to 128.

Some considerations:

- This feature is only available in ONTAP 9.9.1 and later, and only on platforms with more than 256GB of memory (such as the AFF A700, AFF A800, and FAS9000).
- These platforms also increase the total number of available exec contexts per node to 10,000. Because throttling does not occur until over 6,000 execs are allocated (previous total exec limit was 3,000), existing workloads should not notice any negative difference in performance.
- This throttling does not help reduce the number of blocked exec contexts due to per-client overruns. For tuning Linux clients and/or using nconnect, you should still follow the guidance in the “Network connection concurrency and TCP slots: NFSv3” section.

How do I know if connections are being throttled?

If you are running ONTAP 9.9.1 on platforms with 256GB or greater memory, then exec throttling is enabled by default. For all other platforms, this feature does not apply.

To view whether exec throttling is occurring on your cluster, start the `exec_ctx` statistics object.

```
cluster::> set diag
cluster::*> statistics start -object exec_ctx
```

Then, view the statistics. The following counters are displayed if exec throttling is in use:

- **throttle_scale.** The scale-back factor which is currently in effect (1, 8, or 16).
- **throttle_increases.** The number of times the scale-back factor has been increased.
- **throttle_decreases.** The number of time the scale-back factor has been decreased.
- **throttle_hist.** A histogram of allocations at each scale factor (incrementing counters in 1, 8, or 16).

Network connection concurrency and TCP slots: NFSv3

When an NFS mount is established from a client to an ONTAP NFS server, a CID is also established. Each incoming NFS operation gets assigned a placeholder resource in ONTAP called an executive context (`exec_ctx` or `exec`). As operations complete, the reserved `execs` are freed to the system for use with a new incoming operation.

For each CID, ONTAP allows 128 `execs` to be used at any given moment. If a client sends more than 128 operations at a time, then ONTAP pushes back on that client until a new resource is freed. These pushbacks are only microseconds per operation in most cases, but over the course of millions of requests across hundreds of clients, the pushback can manifest into performance issues that don't have the usual signatures of performance issues on storage systems, such as protocol, disk or node latency. As a result, isolating these issues can be difficult and time consuming.

Older Linux kernels (pre-RHEL 6.x days) had a static setting for RPC slot tables of 16. In newer Linux clients, that setting was changed to a maximum of 65,536 and the NFS client would dynamically increase the number of slot tables needed. As a result, newer NFS clients could potentially flood NFS servers with more requests than they can handle at one time.

NFSv4.x operations are sent as compound requests (such as three or four NFS operations per packet) and NFSv4.x sessions slots are used to parallelize requests instead of RPC slot tables. For more information, see "NFSv4.x concurrency — session slots."

Here are a few ways to address performance issue caused by slot tables:

- Add more NFSv3 mount points per client (use different locations in the volume to be effective)
- Throttle the number of NFSv3 requests a single client can send per TCP connection/session
- Use the nconnect mount option to get more TCP connections/sessions per mount (ensure both the client and ONTAP versions support nconnect)

However, before you decide to address RPC slots in your environment it is important to remember that lowering the RPC slot tables on an NFS client is effectively a form of throttling and can negatively affect performance depending on the workload. An NFS client that needs to send one million NFS requests will send those requests regardless of RPC slot table settings. Setting RPC slot tables is essentially telling the NFS client to limit the number of requests it can send at any given time. The decision of whether to throttle the NFS client or let the storage system enact a form of flow control depends on your workload and use case.

Before adjusting these values, it's important to test and identify if too many slot tables will cause performance issues/impact to applications.

Identifying potential issues with RPC slot tables

As previously mentioned, modern NFSv3 clients use dynamic values for RPC slot tables, which means that the client will send as many concurrent operations on a single TCP session as possible—up to 65,336. However, ONTAP allows only 128 concurrent operations per TCP connection, so if a client sends more than 128, ONTAP will enact a form of flow control on NFSv3 operations to prevent rogue clients from overrunning storage systems by blocking the NFS operation (exec contexts in ONTAP) until resources free up. This flow control can manifest as performance issues that cause extra latency and slower job completion times that might not have a readily apparent reason from the general storage system statistics. These issues can appear to be network related, which can send storage administrators down the wrong troubleshooting path.

To investigate whether RPC slot tables might be involved, use the ONTAP performance counter. You can verify whether the number of exec contexts blocked by the connection being overrun is incrementing.

To gather those statistics, run the following command:

```
statistics start -object cid -instance cid
```

Then, review the statistics over a period of time to see if they are incrementing.

```
statistics show -object cid -instance cid -counter execs_blocked_on_cid
```

On NFS clients, you can leverage the `nfsiostat` command to show active in-flight slot tables.

```
# nfsiostat [/mount/path] [interval seconds]
```

When you set lower RPC slot table values on a client, the RPC slot table queue shifts from the storage to the client, so the `rpc bklog` values will be higher.

With the slot tables set to 128, the `rpc bklog` got as high as 360 when creating 5,000,000 files from two clients and sent around 26,000 ops/s.

```
# nfsiostat /mnt/FGNFS 1 | grep "bklog" -A 1
           ops/s       rpc bklog
       25319.000         354.091
--
           ops/s       rpc bklog
       24945.000         351.105
--
           ops/s       rpc bklog
       26022.000         360.763
```

But ONTAP didn’t have to block any of the incoming operations.

```
cluster::*> statistics show -object cid -counter execs_blocked_on_cid -sample-id
All_Multi1_bs65536

    Counter                                                     Value
    -------------------------------- --------------------------------
    execs_blocked_on_cid                                            0

    Counter                                                     Value
    -------------------------------- --------------------------------
    execs_blocked_on_cid                                            0
```

If the RPC slot table values are set to higher values, then the RPC queue (`rpc bklog`) will be lower on the client. In this case, the slot tables were left as the default 65,536 value. The client backlog was 0 and the ops/s were higher.

```
# nfsiostat /mnt/FGNFS 1 | grep "bklog" -A 1
           ops/s       rpc bklog
       22308.303           0.000
--
```

```
          ops/s       rpc bklog
       30684.000           0.000
```

That means the storage would need to absorb more of those RPC calls because the client isn't holding back as many of those operations. We can see that in the ONTAP statistics.

```
cluster::*> statistics show -object cid -counter execs_blocked_on_cid -sample-id
All_Multi1_bs65536

    Counter                                                     Value
    -------------------------------- --------------------------------
    execs_blocked_on_cid                                       145324

    Counter                                                     Value
    -------------------------------- --------------------------------
    execs_blocked_on_cid                                       124982
```

When we exceed a certain number of blocked execs, we'll log an EMS. The following EMS was generated:

```
cluster::*> event log show -node tme-a300-efs01-0* -message-name nblade.execsOverLimit
Time                Node             Severity      Event
------------------- ---------------- ------------- ---------------------------
4/8/2021 17:01:30   node1
                                     ERROR         nblade.execsOverLimit: The number of in-flight
requests from client with source IP x.x.x.x to destination LIF x.x.x.x (Vserver 20) is greater
than the maximum number of in-flight requests allowed (128). The client might see degraded
performance due to request throttling.
```

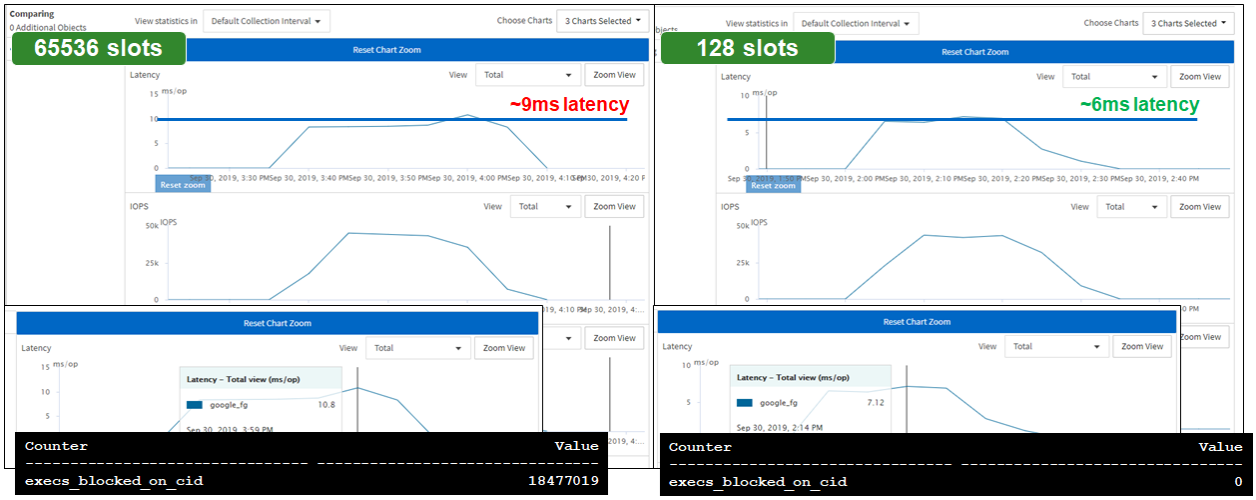
In general, if you aren't seeing `nblade.execOverLimit` EMS events in ONTAP 9.8 and later, RPC slot tables aren't likely causing problems for your workloads. In ONTAP 9.7 and ealrlier, these events do not exist, so you would want to monitor the CID stats and watch for large increments of `exec_blocked_on_cid`. If you're unsure if your environment is having an issue, contact NetApp support.

Example #1: RPC slot table impact on performance — high file count workload

In the example shown in Figure 19, a script was run to create 18 million files across 180,000 subdirectories. This load generation was done from three clients to the same NFS mount. The goal was to generate enough NFS operations with clients that had the default RPC slot table settings to cause ONTAP to enter a flow-control scenario. Then the same scripts were run again on the same clients—but with the RPC slot tables set to 128.

The result was that the default slot tables (65,536) generated 18 million `execs_blocked_on_cid` events and added 3ms of latency to the workload versus the run with the lower RPC slot table setting (128).

Figure 19) Impact of RPC slot tables on NFSv3 performance.



Although 3ms might not seem like a lot of latency, it can add up over millions of operations, considerably slowing down job completion.

Example #2: RPC slot table impact on performance — sequential I/O workload

In the section called “Performance examples for different TCP maximum transfer window sizes,” we show a number of tests that illustrate NFSv3 vs. NFSv4.1 performance differences, along with different wsize/rsize mount option values. While running these tests, we also saw the negative effects of RPC slot tables increasing the number of execs blocked on CIDs causing performance bottlenecks that added 14.4ms of write latency to some of the performance runs, which in turn added 5.5 minutes to the overall job completion times.

The tests were run across two clients on a 10GB network, using a script that runs multiple dd operations in parallel. Overall, eight folders with two 16GB files each were created and then read and deleted.

- When the RPC slots were set to the maximum dynamic value of 65,536, the dd operations took 20 minutes, 53 seconds.
- When the RPC slots were lowered to 128, the same script took just 15 minutes, 23 seconds.

With the 1MB wsize/rsize mount options and 65,536 RPC slots, the `execs_blocked_on_cid` incremented to ~1,000 per node.

```
cluster::*> statistics show -object cid -counter execs_blocked_on_cid

Scope: node1

    Counter                                                     Value
    -------------------------------- ----------------------------------
    execs_blocked_on_cid                                         1001

Scope: node2

    Counter                                                     Value
    -------------------------------- ----------------------------------
    execs_blocked_on_cid                                         1063
```

Figure 20 shows the side-by-side comparison of latency, IOPS, and throughput for the jobs using a 1MB wsize/rsize mount value.

Figure 20) Parallel dd performance — NFSv3 and RPC slot tables; 1MB rsize/wsize.

Figure 21 shows the side-by-side comparison of latency, IOPS, and throughput for the jobs using a 256K wsize/rsize mount value.

Figure 21) Parallel dd performance — NFSv3 and RPC slot tables; 256K rsize/wsize.

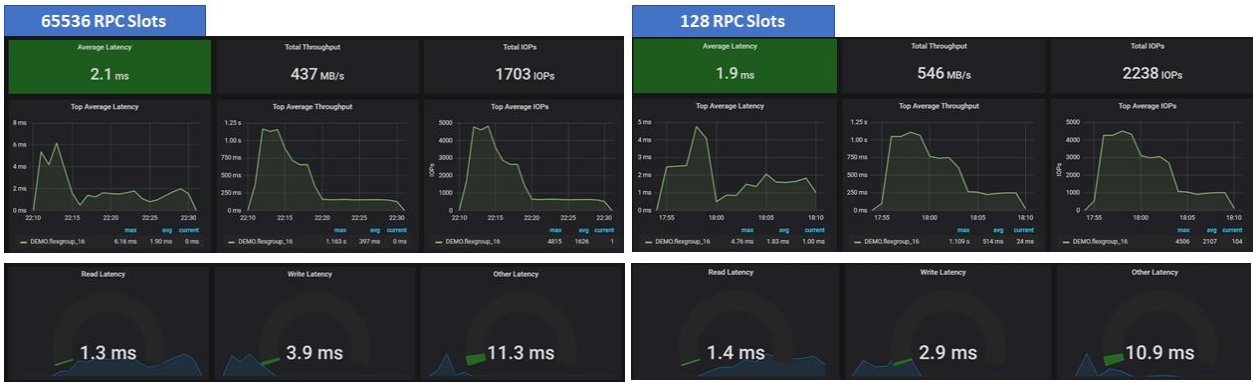


Table 20 shows the comparison of job times and latency with 65,536 and 128 RPC slots.

Table 20) Job comparisons — parallel dd with 65,536 and 128 RPC slots.

Test	Average read latency (ms)	Average write latency (ms)	Completion time
NFSv3 – 1MB wsize/rsize; 65,536 slot tables	9.2 (+3.2ms)	42.3 (+14.4ms)	20m53s (+5m30s)
NFSv3 – 1MB wsize/rsize; 128 slot tables	6	27.9	15m23s
NFSv3 – 256K wsize/rsize; 65,536 slot tables	1.3 (-.1ms)	3.9 (+1ms)	19m12s (+4m55s)
NFSv3 – 256K wsize/rsize; 128 slot tables	1.4	2.9	14m17s
NFSv3 – 64K wsize/rsize; 65,536 slot tables	.2 (0ms)	3.4 (+1.2ms)	17m2s (+2m14s)
NFSv3 – 64K wsize/rsize; 128 slot tables	.2	2.2	14m48s

Resolving issues with RPC slot tables

In some cases, slot tables on NFS clients don't need to be adjusted at all, as the incrementing counters might not be creating a performance issue. Keep in mind that if a client wants to send one million requests to an ONTAP NFS mount, it will send those requests regardless of what the client is tuned to send. The consideration is, where will the pushback come from: the client or the server?

Setting the RPC slot tables to 128 means that the client will only send 128 concurrent requests at a time to the storage system and will hold any excess requests in its own queues, which can be viewed with the

`nfsiostat` command (for an example of `nfsiostat`, see "Identifying potential issues with RPC slot tables").

In some cases, that value might stop the incrementing exec counters in ONTAP (and suppress the EMS messages), but performance might actually suffer depending on the client's configuration and available resources.

ONTAP cannot control the number of slot tables that a client sends per TCP session for NFSv3 operations. Therefore, if there is a performance issue caused by the client slot table settings, clients must be configured to limit the maximum slot tables sent through NFS to no more than 128. The configuration of this setting varies depending on the client OS version, the number of clients, and a remount required is required for this to take effect. For more information on how to set the values for your client OS, contact the client vendor.

As previously mentioned, the value to set the RPC slot tables to can vary depending on workload and number of clients. For example, in the tests mentioned above, 128 RPC slots using a 1MB wsize/rsize achieved a job completion time of 15 minutes, 23 seconds. Setting the RPC slots to 64 lowered that job completion time to 14 minutes, 2 seconds. However, setting the value too low can have the opposite effect, where the clients will be throttled unnecessarily. Test different values in your environment for the best possible results.

Table 21) Job comparisons — parallel dd with 65,536, 128, and 64 RPC slots.

Test	Average read latency (ms)	Average write latency (ms)	Completion time
NFSv3 – 1MB wsize/rsize; 65,536 slot tables	9.2 (+3.2ms)	42.3 (+19.4ms)	20m53s (+6m51s)
NFSv3 – 1MB wsize/rsize; 128 slot tables	6 (-.3ms)	27.9 (+5ms)	15m23s (+1m21s)
NFSv3 – 1MB wsize/rsize; 64 slot tables	6.3	22.9	14m2s

It is possible to get more performance out of a client's NFS connectivity by connecting more mount points to different IP addresses in the cluster on the same client, but that approach can create complexity. For example, rather than mounting a volume at `SVM:/volumename`, multiple mount points on the same client across different folders and IP addresses in the volume could be created.

For example:

```
LIF1:/volumename/folder1
LIF2:/volumename/folder2
LIF3:/volumename/folder3
```

Using nconnect to avoid slot table exhaustion issues

Another possible option is to use the nconnect mount option available for some Linux distributions that can perform multiplexing of NFSv3 over the same TCP connection. This option provides more available concurrent sessions and better overall performance. For example, if you use nconnect=8, you get 128 rpc slots * 8 sessions, for concurrency up to 1,024 slots. This also helps reduce the need to adjust client-side slot table settings.

For example, a test run using NFSv3 and no nconnect applied to the NFS mount with the default 65,536 slot tables on the client showed the following:

- One million 4K files and 2,000 directories created by two clients in ~81 seconds
- A total of 835,324 execs blocked across two A300 AFF nodes

When the same test used nconnect=2, these were the results:

- One million 4K files and 2,000 directories created by two clients in ~82 seconds
- A total of 827,275 execs blocked across two A300 AFF nodes

The, the test was run with nconnect=8:

- One million 4K files and 2,000 directories created by two clients in ~85 seconds
- A total of zero execs blocked across two A300 AFF nodes

The file creation script used can be found here:
https://github.com/whyistheinternetbroken/NetAppFlexGroup/blob/master/file-create-hfc.py

Table 22 shows a side-by-side comparison of this particular test using different nconnect settings and their results. It also illustrates the previously mentioned caveats – blocked `exec contexts` in ONTAP do not always equate to worse performance and that nconnect can add performance based on the number of sessions you specify and the workload in use. This set of tests were all run against ONTAP 9.9.1, which includes the new exec context throttle feature. Ten runs of each were used and averaged out and the default client settings were used. Mount wsize and rsize used was 64K.

This was the workload makeup:

```
create_percent                                  33%
lookup_percent                                  33%
write_percent                                   33%
```

Table 22) High file count creation (one million files) — NFSv3 — with and without nconnect — default slot tables.

Test	Average completion time	Average total execs blocked in ONTAP
NFSv3 – no nconnect	~69.5 seconds	214770
NFSv3 – nconnect=2	~70.14 seconds	88038
NFSv3 – nconnect=4	~70.1 seconds	11658
NFSv3 – nconnect=8	~71.8 seconds	0
NFSv3 – nconnect=16	~71.7 seconds	0

In these tests, we can make the following observations:

- Nconnect does not help much with this type of workload.
- Exec contexts being blocked can create performance issues (as described in “Example #1: RPC slot table impact on performance — high file count workload”) but doesn’t always create performance issues.

The same tests were run again using the 128 maximum RPC slot table setting and only without nconnect and with nconnect=8. In this instance, throttling the RPC slot tables on the clients not only had a slightly positive effect on performance for this workload’s average completion time, but it also had more predictability. For instance, when the slot tables were set to 65,536, completion times out of 10 runs had a wide variance - from 55.7 seconds to 81.7 seconds for this workload—whereas 128 slot tables kept performance at a more consistent range of 68-71 seconds. This is because the storage never had to push back on the NFS operations. With more clients added to the mix, the predictability becomes more impactful, as one or two clients can potentially bully other clients into poor performance results – especially in ONTAP releases prior to 9.9.1 (where exec context throttling was added to help mitigate bully workloads).

When nconnect was added to these tests, the performance suffered a bit, because each client wasn’t able to push as many NFS operations across the multiple TCP sessions because the client was throttled to 128. If you plan on using nconnect for your workloads, you should consider not setting the RPC slot table values on the clients at all and instead let nconnect spread the workload across TCP sessions.

Table 23) High file count creation (one million files) — NFSv3 — with and without nconnect — 128 slot tables.

Test	Average completion time	Average total execs blocked in ONTAP
NFSv3 – no nconnect	~69.4 seconds	0
NFSv3 – nconnect=8	~71.2 seconds	0

When the slot tables were scaled back even further (to 16), the performance suffered significantly, because the client is now sending just 16 requests at a time, so the script takes ~28.1 seconds more to complete without nconnect. With nconnect, we were able to keep around the same average completion time as the other tests with more slot tables (~70.6 seconds), as we get 16 slot tables per session (16 * 8 = 128), rather than 16 slot tables on a single session. Because we can send 128 operations per session with nconnect and only 16 per session without, the performance is greatly improved when using nconnect with this configuration. However, in environments with hundreds of clients, setting the slot tables to 16 might be the only way to avoid performance problems caused by slot table overruns.

Table 24) High file count creation (one million files) — NFSv3 — with and without nconnect — 16 slot tables.

Test	Average completion time	Average total execs blocked in ONTAP
NFSv3 – no nconnect	~99.3 seconds	0
NFSv3 – nconnect=8	~70.6 seconds	0

From these results, nconnect can provide a way to better distribute the RPC slot requests across TCP connections without costing much in the way of performance and removing the need to adjust RPC slot tables in environments that require high performance and a large number of NFS operations. For more information about nconnect, see the section called “Nconnect.”

Note: Because of the variant nature of workloads and impacts to performance, you should always test various client settings, mount options, and so on in your environment, because there is no single correct NFS configuration.

Setting RPC slot tables for environments with large client counts

Although the ONTAP session limit for slot tables per TCP connection is 128, there are also node-level limits for exec contexts that can be exceeded.

For example, if a single node can handle up to 3,000 exec contexts, and each TCP session can handle up to 128 exec contexts, then you can have up to 24 concurrent TCP sessions maxing out the RPC slot tables at any given time before the resources are exhausted and ONTAP has to pause to let the resources release back for use (3,000/128 = 23.47). For examples of how to see the available exec contexts per node, see “Exec context throttling.”

That does not mean that ONTAP can only support up to 24 clients per node; ONTAP supports up to 100,000 TCP connections per node (platform dependent). Rather, what this means is that if 24 or more clients are sending the maximum allowed slot entries per connection (128) all at the same time to a single node, then there will be some latency build up as ONTAP works to free up resources.

Table 25 shows examples of how many clients can send the maximum concurrent operations per connection to a single node.

Table 25) Total clients at maximum concurrent operations (128) before node exec context exhaustion.

Node type	Total available exec contexts per node	Total clients sending maximum operations per connection (128) per node
AFF8040	1,500	1,500/128 = ~11.7
AFF A300	3,000	3,000/128 = ~23.4
AFF A800	10,000	10,000/128 = ~78.1*

Note: *This number varies based on the exec context throttling.

If RPC slot table maximum entries are higher on the client, fewer clients are needed to hit the maximum value. If you have hundreds of clients in an environment that are all working at the same time to the same nodes in the cluster (such as with EDA workloads), then you should consider setting the RPC slot tables to much lower values based on the number of clients and the number of cluster nodes participating in the workload.

Table 26) Total clients using 16 slot tables before node exec context exhaustion.

Node type	Total available exec contexts per node	Total clients sending max operations per connection (128) per node
AFF8040	1,500	1,500/16 = ~93.7
A300	3,000	3,000/16 = ~187.5
A800	10,000	10,000/16 = ~625*

In addition, the following approaches can help improve overall performance for workloads with many clients:

- Clusters with more nodes and data LIFs on each node.
- DNS round robin to spread network connections across more nodes on initial mount.
- FlexGroup volumes to leverage more hardware in the cluster.
- FlexCache volumes to spread read-heavy workloads across more nodes and mount points.
- Consider setting RPC slot table values lower on NFS clients to reduce the number of concurrent operations; values are platform/ONTAP version-dependent and client count-dependent. For example, see “Table 25.”
- Nconnect to increase the single client performance.
- ONTAP 9.9.1 or later when using platforms with 256GB RAM or greater for benefits of “Exec context throttling” to help mitigate bully workload impact.

The slot table recommendations will adjust based on ONTAP hardware, ONTAP version, NFS mount options, and so on. Testing different values in your environment is highly recommended. Does the RPC slot table limit affect other NAS protocols?

RPC slot table limits affect only NFSv3 traffic:

- SMB clients use different connection methodologies for concurrency, such as SMB multichannel, SMB multiplex, and SMB credits. The SMB connection methodology depends on client/server configuration and protocol version. For example, SMB 1.0 uses SMB multiplex (mpx), whereas SMB2.x uses SMB credits.
- NFSv4.x clients do not use RPC slot tables—instead, they use state IDs and session slots to control the flow of concurrent traffic from clients.

Table 27 shows test run results from NFSv3 and NFSv4.1 using the same 65536 slot table values.

Table 27) Job comparisons - parallel dd — NFSv3 and NFSv4.1 with 65536 RPC slots.

Test	Average read latency (ms)	Average write latency (ms)	Completion time
NFSv3 – 1MB wsize/rsize; 65,536 slot tables	9.2 (+2.7ms)	42.3 (+5.5ms)	20m53s (+5m47s)
NFSv4.1 – 1MB wsize/rsize; 65,536 slot tables	6.5	36.8	15m6s
NFSv3 – 256K wsize/rsize; 65,536 slot tables	1.3 (-.1ms)	3.9 (+.7ms)	19m12s (+7m2s)
NFSv4.1 – 256K wsize/rsize; 65,536 slot tables	1.4	3.2	12m10s
NFSv3 – 64K wsize/rsize; 65,536 slot tables	.2 (+.1ms)	3.4 (+2.2ms)	17m2s (+1m54s)

Test	Average read latency (ms)	Average write latency (ms)	Completion time
NFSv4.1 – 64K wsize/rsize; 65,536 slot tables	.1	1.2	15m8s

Virtual memory tuning

Another way to tune NFS clients to improve performance is to modify the virtual memory settings and the dirty buffer values (`vm.dirty`). The virtual memory cache is important to how a client performs file caching and modifying the default values can control how well reads or writes perform on a client. The client writes file operations directly to RAM before flushing them to disk, which means there are fewer back and forth conversations between the NFS client and server for workloads. For more information, see https://www.kernel.org/doc/Documentation/sysctl/vm.txt.

There are three triggers for file caches being flushed:

- **Time-based.** After a buffer reaches the age defined by this tunable, it must be marked for cleaning (such as flushing, also known as writing to storage).
- **Memory pressure.** After the allocated memory has been filled, the file cache is flushed.
- **Close.** When a file handle is closed, all dirty buffers are asynchronously flushed to storage.

The default cache settings on clients provide a percentage of RAM dedicated for file caching that is usually adequate for most workloads. The following are the default values on a CentOS/RHEL 8.3 client:

```
vm.dirty_background_bytes = 0 ## modifying this sets vm.dirty_background_ratio to 0
vm.dirty_background_ratio = 10  ## modifying this sets vm.dirty_background_bytes to 0
vm.dirty_bytes = 0 ## modifying this sets vm.dirty_ratio to 0
vm.dirty_ratio = 30 ## modifying this sets vm.dirty_bytes to 0
vm.dirty_expire_centisecs = 3000
vm.dirty_writeback_centisecs = 500
vm.dirtytime_expire_seconds = 43200
```

The following describes the attribute for the values we will focus on in this section. These values can be set without needing to remount or reboot the client to have them take effect.

vm.dirty_ratio | vm.dirty_ bytes

These tunables define the amount of RAM made available for data modified but not yet written to stable storage. Whichever tunable is set, automatically sets the other tunable zero; Red Hat advises against manually setting either of the two tunables to zero. Setting the `vm.dirty_bytes` to a static value can provide more consistent performance across clients regardless of the amount of RAM present, but that also can artificially limit performance on systems with large amounts of available RAM for workloads.

vm.dirty_background_ratio | vm.dirty_background_bytes

These tunables define the starting point at which the Linux writeback mechanism begins flushing dirty blocks to stable storage.

vm.dirty_expire_centisecs

This tunable defines how old a dirty buffer can be before it must be tagged for asynchronously writing out. Some workloads might not close file handles immediately after writing data. Without a file close, there is no flush until either memory pressure or 30 seconds passes (by default). Waiting for these values can prove suboptimal for application performance, so reducing the wait time can help performance in some use cases.

vm.dirty_writeback_centisecs

The kernel flusher thread is responsible for asynchronously flushing dirty buffers between each flush thread sleep. This tunable defines the amount of time spent sleeping between buffer flushes. Lowering this value in conjunction with `vm.dirty_expire_centisecs` can also improve performance for some workloads.

Other impacts of untuned filesystem caches

Because the default virtual memory tunables in modern systems might not do justice to the amount of available RAM, writeback can potentially slow down other storage operations that have file system caches that have not been tuned to use more available virtual memory.

Some of these potential impacts include:

- Slow or hung directory listings (ls)
- Read throughput significantly lower than write throughput
- High latency (seconds or higher) from nfsiostat

These issues would only be seen from the client performing the mixed read/write workloads and would be seen across all mounted NFS volumes from the storage endpoint on that client during the impact period.

What workloads benefit the most from virtual memory tuning?

Workload types can be highly variant and the type of workload in use and how files are written controls how impactful virtual memory tuning will be.

In Table 28, one million files were created by using a Python script with the f.write function. The following changes were made to the virtual memory settings between tests:

- `vm.dirty_ratio` changed from 30% to 40%
- `vm.dirty_background_ratio` from 10% to 20%
- `vm.dirty_expire_centisecs` from 3,000 to 300
- `vm.dirty_writeback_centisecs` from 500 to 100

The results were underwhelming because this type of operation closes the file much faster than the cache can expire, so there is little for the client to cache. However, when the client is configured to keep more file cache in memory, the total execs blocked by ONTAP are reduced, because the client isn't sending many requests to the storage as frequently.

Table 28) One million files using f.write — NFSv3, 65536 slots — VM dirty bytes defaults versus tuned.

Test	Average completion time	Average total execs blocked in ONTAP	Time delta
NFSv3 – no nconnect – default vm.dirty settings	~69.1 seconds	214,770	-
NFSv3 – no nconnect – vm.dirty settings tuned	~69.5 seconds	144,790	+.4 seconds
NFSv3 – nconnect=8 – default vm.dirty settings	~71.8 seconds	0	-
NFSv3 – nconnect=8 – vm.dirty settings tuned	~69.5 seconds	0	-2.3 seconds

When the file size was increased, the overall completion times drastically improved with the virtual memory settings, while nconnect didn't make a lot of difference because our bottleneck here was not due to TCP session limitations.

In this test, dd was used to create 50 x 500MB (2 clients, 1 file per folder, 25 folders, 25 simultaneous processes per client) files, with a 256K wsize/rsize mount value.

Table 29) 50x 500MB files using dd — NFSv3, 65536 slots — VM dirty bytes defaults versus tuned.

Test	Average completion time	Average total execs blocked in ONTAP	Time delta (vm.dirty defaults vs. vm.dirty set)
NFSv3 – no nconnect – default vm.dirty settings	~134.4 seconds	0	-
NFSv3 – no nconnect – vm.dirty settings tuned	~112.3 seconds	0	-22.1 seconds
NFSv3 – nconnect=8 – default vm.dirty settings	~132.8 seconds	0	-
NFSv3 – nconnect=8 – vm.dirty settings tuned	~112.9 seconds	0	-19.9 seconds

As seen in the high file count/small files example, these settings do not make a drastic difference for every workload, so it's important to test different values until you find the correct combination.

After you find the right values, you can use `/etc/sysctl.conf` to retain the values on reboot.

NFSv4.x concurrency — session slots

NFSv3 can have performance limitations with TCP slot tables, but NFSv4.x has its own limits when dealing with connection concurrency and TCP slot table settings on NFS clients do not apply to NFSv4.x operations. Instead, NFSv4.x uses session slots for concurrency.

NFSv4.x session slots operate in a similar fashion, in that the number of requests sent by a client over a single TCP connection are limited to a set value, as specified by this Advanced Privilege option:

```
[-v4.x-session-num-slots <integer>] - Number of Slots in the NFSv4.x Session slot tables
(privilege: advanced)
This optional parameter specifies the number of entries in the NFSv4.x session slot table. By
default, the number of slots is 180. The maximum value is 2000.
```

Note: The stated maximum is 2,000, but it is not recommended to exceed 1,024, because you might experience NFSv4.x session hangs as per bug 1392470.

When an NFSv4.x session is set up, the client and server negotiate the maximum requests allowed for the session, with the lower value (client and server settings) being applied. Most Linux clients default to 64 session slots. This value is tunable in Linux clients through the modprobe configuration.

```
$ echo options nfs max_session_slots=180 > /etc/modprobe.d/nfsclient.conf
$ reboot
```

You can see the current value for an NFSv4.x mount with the `systool` command (found in the sysfsutils package):

```
# systool -v -m nfs | grep session
    max_session_slots   = "64"
```

After modifying the client option:

```
# systool -v -m nfs | grep session
    max_session_slots   = "180"
```

Despite the value default of 180 for the NFS server, ONTAP still has a per-connection limit of 128 exec contexts per CID, so it is still possible for an NFSv4.x client to overrun the available exec contexts on a single TCP connection to ONTAP and enter a pause state while resources are freed to the system if the

value is set too high. If session slots are overrun, this also increaseS the `execs_blocked_by_cid` counter mentioned in the “Identifying potential issues with RPC slot tables” section.

In most cases, the default 180 value does not encounter this issue, and setting the session slots to a higher value can create conditions where you might run out of system resources. To get more performance out of your clients, consider using the nconnect option for more parallel handling of these operations rather than adjusting the session slot values.

Note: Be sure to use the latest patched release of ONTAP for NFSv4.x to avoid issues such as this: High NFS Latency with multiple connections to node LIFs from common client TCP socket

Can increasing session slots increase overall performance?

In short, yes—increasing session slots for the client and server can increase overall performance, because you’re providing more concurrency to the workload. However, the performance gains will vary based on workload type, number of slots set, robustness of the clients, as well as how busy the storage system is. It can also impact other workloads on the client, as available resources can be starved out by a single workload. Testing different session slot values is important to ensure that the changes don’t negatively impact your environment.

In Table 30, we measured the completion time, average IOPS, and throughput for a high file count workload (one million x 4KB files) and a low count sequential write workload (32x 2GB files).

The following parameters were used:

- FlexGroup volume across two AFF A300 nodes
- Two CentOS 8.3 clients
- NFSv4.1 mounts (AUTH_SYS, 256K wsize, no pNFS, nconnect=8)
- Virtual memory tuning as per Virtual memory tuning

Table 30) NFSv4.x session slot performance comparison.

Test	Completion time (seconds)	Average IOPS	Average MBps
NFSv4.1 – 1 million x 4KB files (180 session slots)	~253.9	~11688	~15.2
NFSv4.1 – 1 million x 4KB files (256 session slots)	~240.6	~12685	~16.7
NFSv4.1 – 1 million x 4KB files (512 session slots)	~246.5	~12342	~16.1
NFSv4.1 – 1 million x 4KB files (1,024 session slots)	~245.5	~12378	~16.3
NFSv4.1 – 32 x 2GB files (180 session slots)	~148.3	~902	~224.5
NFSv4.1 – 32 x 2GB files (256 session slots)	~149.6	~889	~221.5
NFSv4.1 – 32 x 2GB files (512 session slots)	~148.5	~891	~222
NFSv4.1 – 32 x 2GB files (1024 session slots)	~148.8	~898	~223.7

Table 31) NFSv4.x session slot performance — Percent change versus 180 slots.

Test	Completion time	IOPS	MBps
NFSv4.1 – 1 million x 4KB files (256 session slots)	-5.2%	+8.5%	+9.8%
NFSv4.1 – 1 million x 4KB files (512 session slots)	-2.9%	+5.6%	+5.9%
NFSv4.1 – 1 million x 4KB files (1,024 session slots)	-3.3%	+5.9%	+7.2%
NFSv4.1 – 32 x 2GB files (256 session slots)	+0.9%	-1.4%	-1.3%
NFSv4.1 – 32 x 2GB files (512 session slots)	+0.1%	-1.2%	-1.1%
NFSv4.1 – 32 x 2GB files (1,024 session slots)	+0.3%	-0.4%	-0.4%

Observations

In the high metadata/high file count creation test, more session slots improved completion time, IOPS, and throughput overall, but there seemed to be a sweet spot at 256 session slots against 1,024 session slots.

In the sequential write workload, there was a slight performance degradation when using higher session slots. This shows that increasing session slots can help, but not in all workload types.

NFSv4.x client IDs/NFS4ERR_CLID_INUSE

When NFSv4.x clients mount NFS exports in ONTAP, each client sends its host name as part of the RPC packet in the NFS mount packet. The NFS server responds with a client ID to keep track of the NFSv4.x session.

If another NFSv4.x client with the same host name (but a different IP address), then the NFS server responds with an error:

```
50 11.856227 10.x.x.x 10.x.x.y NFS V4 Call (Reply In 51) EXCHANGE_ID
51 11.856407 10.x.x.y 10.x.x.x NFS V4 Reply (Call In 50) EXCHANGE_ID Status: NFS4ERR_CLID_INUSE
```

As per the NFSv4.1 RFC standard, this is a security mechanism: https://tools.ietf.org/html/rfc5661#section-2.10.8.3.

```
2.10.8.3.  Protection from Unauthorized State Changes

As described to this point in the specification, the state model of NFSv4.1 is vulnerable to an
attacker that sends a SEQUENCE operation with a forged session ID and with a slot ID that it
expects the legitimate client to use next.  When the legitimate client uses the slot ID with the
same sequence number, the server returns the attacker's result from the reply cache, which
disrupts the legitimate client and thus denies service to it.  Similarly, an attacker could send
a CREATE_SESSION with a forged client ID to create a new session associated with the client
ID.  The attacker could send requests using the new session that change locking state, such as
LOCKU operations to release locks the legitimate client has acquired. Setting a security policy
on the file that requires RPCSEC_GSS credentials when manipulating the file's state is one
potential work around, but has the disadvantage of preventing a legitimate client from releasing
state when RPCSEC_GSS is required to do so, but a GSS context cannot be obtained (possibly
because the user has logged off the client).

…

The SP4_MACH_CRED state protection option uses a machine credential where the principal that
creates the client ID MUST also be the principal that performs client ID and session maintenance
operations. The security of the machine credential state protection approach depends entirely on
safe guarding the per-machine credential. Assuming a proper safeguard using the per-machine
credential for operations like CREATE_SESSION, BIND_CONN_TO_SESSION, DESTROY_SESSION, and
DESTROY_CLIENTID will prevent an attacker from associating a rogue connection with a session, or
associating a rogue session with a client ID.
```

From RFC-5661, this is what that error means: https://tools.ietf.org/html/rfc5661#section-15.1.13.2.

```
 15.1.13.2.  NFS4ERR_CLID_INUSE (Error Code 10017)
While processing an EXCHANGE_ID operation, the server was presented with a co_ownerid field that
matches an existing client with valid leased state, but the principal sending the EXCHANGE_ID
operation differs from the principal that established the existing client. This indicates a
collision (most likely due to chance) between clients.  The client should recover by changing the
co_ownerid and re-sending EXCHANGE_ID (but not with the same slot ID and sequence ID; one or both
MUST be different on the re-send).
```

In these instances, the NFSv4.x client should retry the mount with a new client ID. In some cases, use of the NFS mount option `-clientaddr` might be needed. For more information, see RedHat Bugzilla #1821903.

Workarounds

Although the clients are behaving as expected per the RFC standards, there are a few ways to work around this issue.

Workaround #1: Different data LIF in the same SVM

The NFSv4.x client ID issue occurs when you attempt to mount clients with the same host name to the same data LIF in the SVM. However, if you can create additional data LIFs and use those in the mounts, this issue does not occur. In some cases, creating additional data LIFs is not possible, such as when using managed cloud services like Azure NetApp Files or NetApp Cloud Volumes Service.

Workaround #2: Different NFSv4.x versions

Another way to work around the NFSv4.x client ID issue is if one client mounts a volume with a different NFSv4.x version than the other client with the same host name. For example, if client1 (client.domain.com) mounts using NFSv4.1 and client2 (client.domain.com) mounts using NFSv4.2, then this issue does not occur. The caveat is that the ONTAP version you are using must support the NFS versions being attempted. ONTAP 9.8 and later supports NFSv4.2, but managed cloud services such as Azure NetApp Files or NetApp Cloud Volumes Service do not currently support NFSv4.2.

Workaround #3: Change the client name used with NFSv4.x mounts

By default, NFSv4.x uses the client's host name for the client ID value when mounting to the NFS server. However, there are client-side NFS options you can leverage to change that default behavior and override the client ID used for NFSv4.x mounts.

To do this, set the NFS option nfs4-unique-id on the client to a static value for all clients that will use the same host names (see RedHat Bugzilla 1582186). If you add this value to the `/etc/modprobe.d/nfsclient.conf` file, it retains across reboots.

You can see the setting on the client as:

```
 # systool -v -m nfs | grep -i nfs4_unique
    nfs4_unique_id      = ""
```

To set it, run the following command:

```
echo options nfs nfs4_unique_id=[string] > /etc/modprobe.d/nfsclient.conf
reboot
```

For example:

```
# echo options nfs nfs4_unique_id=uniquenfs4-1 > /etc/modprobe.d/nfsclient.conf

# systool -v -m nfs | grep -i nfs4_unique
    nfs4_unique_id      = "uniquenfs4-1"
```

For more information about this option, see NFS Client.

This can be used for any ONTAP version because it is a client-side setting. This includes managed cloud services such as Azure NetApp Files or NetApp Cloud Volumes Service.

NFS silly renames

With NFS, when a file is locked by a client, if the file is deleted, then NFS performs what is called a silly rename, where the file is renamed to an `.nfs*` name. In these scenarios, if an application needs to recursively delete directories, then these deletes will fail because the directory won't technically be empty.

The NFSv4.1 RFC offers a new result flag called `OPEN4_RESULT_PRESERVE_UNLINKED` that addresses silly renames:

```
The use of the OPEN4_RESULT_PRESERVE_UNLINKED result flag allows a client to avoid the common
implementation practice of renaming an open file to ".nfs<unique value>" after it removes the
file.  After the server returns OPEN4_RESULT_PRESERVE_UNLINKED, if a client sends a REMOVE
operation that would reduce the file's link count to zero, the server SHOULD report a value of
zero for the numlinks attribute on the file.
```

This flag is currently unsupported in ONTAP's NFSv4.1 implementation.

How ONTAP NFS handles atime updates

Access time (or atime) refers to the last time a file was accessed through an NFS client. This can be seen through the `stat` command in Linux. In the following example, a stat of the file `vmware-1.log` on an NFS mount from this client shows an access timestamp of January 9, 2021.

```
# stat vmware-1.log
  File: 'vmware-1.log'
  Size: 281796          Blocks: 560        IO Block: 65536  regular file
Device: 27h/39d Inode: 791804619   Links: 1
Access: (0600/-rw-------)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2021-01-09 15:39:30.202594000 -0500
Modify: 2021-01-09 15:39:30.232596000 -0500
Change: 2021-01-14 15:40:29.946918292 -0500
```

A command such as `cat` should update the atime for that file, and it does.

Now the atime shows Jan 14, 2021.

```
# cat vmware-1.log
# stat vmware-1.log
  File: 'vmware-1.log'
  Size: 281796          Blocks: 552        IO Block: 4096   regular file
Device: fd00h/64768d    Inode: 442679      Links: 1
Access: (0600/-rw-------)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2021-01-14 15:41:11.603404196 -0500
Modify: 2021-01-09 15:39:30.232596000 -0500
Change: 2021-01-14 15:40:29.946918292 -0500
```

In some instances, Linux clients might not update the atime properly due to client-side caching. If atimes are not updating properly on commands such as `cat`, try remounting the NFS mount or dropping the caches for the client (such as by running `echo 1 > /proc/sys/vm/drop_caches`; this drops all caches on the client). From the CentOS 7.8 man pages:

```
The Linux client handles atime updates more loosely, however.  NFS clients maintain good
performance by caching data, but that means that application reads, which normally update atime,
are not reflected to the server where a file's atime is actually maintained.
```

ONTAP updates atime whenever the client notifies the storage system of the update. If the client doesn't notify ONTAP, then there's no way for the atime to update. You can disable atime updates with the advanced privilege volume level option `-atime-update`.

```
[-atime-update {true|false}] - Access Time Update Enabled (privilege: advanced)
This optionally specifies whether the access time on inodes is updated when a file is read. The
default setting is true.
```

NAS flow control

ONTAP also provides a NAS-level flow control. This flow control mechanism is separate from the TCP flow control enabled on the NICs and switches of the data network. It is always on and implemented at the NAS protocol stack to prevent rogue clients from overloading a node in the cluster and creating a denial of service (DoS) scenario. This flow control affects all NAS traffic (CIFS and NFS).

How it works

If a client sends too many packets to a node, the flow control adjusts the window size to zero and tells the client to wait on sending any new NAS packets until the other packets are processed. If the client continues to send packets during this zero window, then the NAS protocol stack flow control mechanism sends a TCP reset to that client.

Contact NetApp Technical Support if you suspect a performance problem related to NAS flow control.

Using FlexClone to quickly change all UIDs/GIDs in a volume

In high file count workloads, such as software development environments, occasionally, a set of files might be owned by, and access is limited to, a specific user or group. If another developer needs access to these files, the dataset will need to have the owner/permissions changed. If there are millions of files and folders, that operation would take a long time. Additionally, we don't want to break access for other users in production, so we'd need a way to solve this issue quickly and without disruption.

NetApp FlexClone volumes provide a way to create an instant Snapshot-backed copy of a volume without taking up space in ONTAP, but they also provide a way to quickly change file and folder ownership for everything in that clone. If you want those changes to be permanent, split the clone to its own volume (which would then use up capacity in ONTAP). When we ran the following command to clone a volume with over one million files and folders and changed the UID and GID on all of them, the command took approximately 10 seconds to complete:

```
cluster::*> vol clone create -vserver DEMO -flexclone clone -parent-vserver DEMO -parent-volume
flexvol -junction-path /clone -uid 1100 -gid 1101
[Job 12606] Job succeeded: Successful
```

Auxiliary GIDs — addressing the 16 GID limitation for NFS

NFS has a specific limitation for the maximum number of auxiliary GIDs (secondary groups) that can be honored in a single NFS request. The maximum for AUTH_SYS/AUTH_UNIX is 16 and for AUTH_GSS (Kerberos) it is 32. This is a known protocol limitation of NFS.

ONTAP provides the ability to increase the maximum number of auxiliary groups to 1,024 by avoiding truncation of the group list in the NFS packet by prefetching the requesting user's group from a name service.

```
auth-sys-extended-groups
extended-groups-limit
```

How it works

The options to extend the group limitation work just the way that the manage-gids option for other NFS servers works. Basically, rather than dumping the entire list of auxiliary GIDs a user belongs to, the option does a lookup for the GID on the file or folder and returns that value instead.

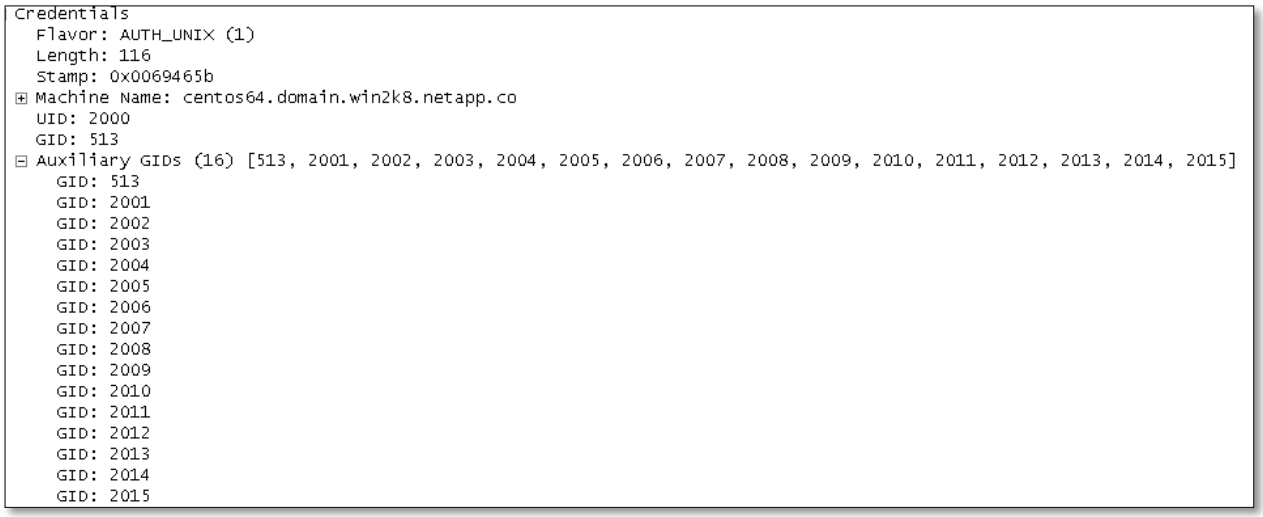
From the man page for mountd:

```
-g  or  --manage-gids

Accept requests from the kernel to  map  user  id  numbers  into lists  of group  id  numbers for
use in access control.  An NFS request will normally except when using Kerberos or other
cryptographic  authentication)  contains  a  user-id  and  a list of group-ids.  Due to a
limitation in the NFS protocol, at most  16 groups ids can be listed.  If you use the -g flag,
then the list of group ids received from the client will be replaced by a list of  group ids
determined by an appropriate lookup on the server.
```

When an access request is made, only 16 GIDs are passed in the RPC portion of the packet (Figure 22).

Figure 22) RPC packet with 16 GIDs.



Any GID beyond the limit of 16 is dropped by the protocol. Extended GIDs can be used with external name services, or locally on the cluster if the users and groups are configured properly. To make sure that a local UNIX user is a member of multiple groups, use the `unix-group adduser(s)` command:

```
COMMANDS
     adduser - Add a user to a local UNIX group

     addusers - Add a list of users to a local UNIX group
```

Performance impact of extended GIDs

Extended groups have a minimal performance penalty, generally in the low single digit percentages. Higher metadata NFS workloads would likely have more effect, particularly on the system's caches. Performance can also be affected by the speed and workload of the name service servers. Overloaded name service servers are slower to respond, causing delays in prefetching the GID.

For more information regarding name services, see TR-4668: Name Service Best Practices.

Considerations for numeric ID authentication (NFSv3 and NFSv4.x)

NFSv3 using AUTH_SYS sends numeric ID information for users and groups to perform user authentication to NFS mounts for permission resolution.

NFSv4.x with ONTAP has a feature that allows NFSv4.x mounts to leverage numeric ID strings instead of name strings, which allows NFSv4.x operations without needing centralized name services, matching names/numeric IDs on client/server, matching ID domains, etc. (`-v4-numeric-ids`).

Enabling the `-auth-sys-extended-groups` option causes numeric ID authentication to fail if the UNIX user numeric ID can't be translated into a valid UNIX user name in name services. This counteracts the `-v4-numeric-ids` option because ONTAP needs to query the incoming numeric user ID to search for any auxiliary groups for authentication. If the incoming numeric ID cannot be resolved to a valid UNIX user or the client's UNIX numeric UID is different than the numeric UID ONTAP knows about, then the lookup fails with `secd.authsys.lookup.failed` in the event log and ONTAP responds to the client with the AUTH_ERROR `client must begin a new session`, which appears as "Permission denied."

To use both options, use the following guidance:

- If you require users and groups that either cannot be queried from both NFS client and server or have mismatched numeric IDs, you can leverage NFS Kerberos and NFSv4.x ACLs to provide proper authentication with NFSv4.x, as clients will send name strings instead of numeric IDs.
- If you are using `-auth-sys-extended-groups` with AUTH_SYS and without NFSv4.x ACLs, any user that requires access through NFS requires a valid UNIX user in the name service database specified in ns-switch (can also be a local user).

For more information about the `-v4-numeric-ids` option, see "Bypassing the name string — Numeric IDs."

Using local files

Using the `-auth-sys-extended-groups` option allows support for >16 auxiliary groups and is best applied with external name services, such as LDAP. However, if an external name service server is unavailable, you can leverage local files to create UNIX users and groups for support with `-auth-sys-extended-groups`. Simply create the users and groups you want to use and run the `unix-group adduser` command to populate the auxiliary group list.

For more information on LDAP configuration, see TR-4835: How to Configure LDAP in ONTAP.

Considerations for Active Directory LDAP

By default, in Microsoft Active Directory LDAP servers, the MaxPageSize attribute is set to a default of 1,000. This means that groups beyond 1,000 are truncated in LDAP queries. To enable full support with the 1,024 value for extended groups, the MaxPageSize attribute must be modified to reflect the 1,024 value. For information about how to change that value, see the following Microsoft TechNet article: How to view and set LDAP policy in Active Directory by using Ntdsutil.exe

Contact Microsoft support for concerns with modifying this value and to review the TechNet library article MaxPageSize is set too high.

High file count considerations

For more detailed information about high file count environments, see the corresponding section in TR-4571: NetApp FlexGroup Volume Best Practices and Implementation Guide.

Volume style recommendation for high file count environments

High file count refers to a workload that creates and stores hundreds of thousands to upwards of billions of files in a single namespace. Datasets such as these generate many metadata operations, which can impact performance on a FlexVol volume. As a result, in high file count environments, a FlexGroup volume is highly recommended. For more information about FlexGroup volumes, see TR-4571: NetApp FlexGroup Volume Best Practices and Implementation Guide.

NFS on nontraditional operating systems

The following section covers NFS use on nontraditional NFS platforms, such as Windows and Apple operating systems. Windows NFS support was first added to ONTAP 8.2.3 and 8.3.1.

NFS on Windows

To use NFS with ONTAP systems earlier than ONTAP 8.2.3 and 8.3.1 on Windows operating systems, server administrators can install third-party tools, such as the Hummingbird/OpenText NFS Client. Red Hat's Cygwin emulates NFS but leverages the SMB protocol rather than NFS, which requires a CIFS license. True Windows NFS is available natively only through Services for Network File System or third-party applications such as Hummingbird/OpenText.

Native Windows NFS in ONTAP

In RFC 1813, the following section covers MS-DOS as a supported client for NFS:

```
   nlm4_share

      struct nlm4_share {
           string      caller_name<LM_MAXSTRLEN>;
           netobj      fh;
           netobj      oh;
           fsh4_mode   mode;
           fsh4_access access;
      };

   This structure is used to support DOS file sharing. The
   caller_name field identifies the host making the request.
   The fh field identifies the file to be operated on. The oh
   field is an opaque object that identifies the host or process
   that is making the request. The mode and access fields
   specify the file-sharing and access modes. The encoding of fh
   is a byte count, followed by the file handle byte array. See
   the description of nlm4_lock for more details.
```

The way that Windows uses NLM is with nonmonitored lock calls. The following nonmonitored lock calls are required for Windows NFS support:

```
NLM_SHARE
NLM_UNSHARE
NLM_NM_LOCK
```

These lock calls were not supported in versions of clustered ONTAP 8.3.1 and earlier or in versions of clustered ONTAP 8.2.3 and earlier. Bug 296324 covers this point.

Note: PCNFS, WebNFS, and HCLNFS (legacy Hummingbird NFS client) are not supported with ONTAP storage systems and there are no plans to include support for these clients.

Considerations for using Windows NFS in ONTAP

Keep the following considerations in mind when using Windows NFS with ONTAP:

- NSM is not supported in Windows NFS. Therefore, volume moves and storage failovers can cause disruptions that might not be seen on NFS clients that do support NSM.
- If you are using Windows NFS on an SVM, the following options must be set to Disabled. An example of one issue that can occur if EJUKEBOX is not disabled is bug 918755.

```
enable-ejukebox
v3-connection-drop
```

Note: These options are enabled by default. Disabling them does not harm NFS data, but might cause some unexpected behavior, such as client crashes. If you want to prevent unexpected issues, consider using a separate SVM for Windows NFS clients.

- Always mount Windows NFS by using the mount option `mtype=hard`.
- When using Windows NFS, the showmount option should be enabled. Otherwise, renames of files and folders on Windows NFS clients might fail.

```
cluster::> nfs server modify -vserver SVM -showmount enabled
```

- Windows NFS clients are not able to properly see the used space and space available through the `df` commands.

Example of mounting NFS in Windows:

```
C:\>mount -o mtype=hard \\x.x.x.e\unix  Z:
Z: is now successfully connected to \\x.x.x.e\unix
```

```
The command completed successfully.
```

Enabling Windows NFS support

In ONTAP, there is a specific NFS server option that must be toggled to Enabled to allow Windows NFS clients. This option is disabled by default. If reverting from a ONTAP version that supports Windows NFS, this option must be disabled prior to attempting the revert.

```
cluster::> nfs server show -vserver nfs_svm -fields v3-ms-dos-client
vserver v3-ms-dos-client
------- ----------------
nfs_svm disabled
```

Windows NFS use cases

Windows NFS can be used to provide access to NetApp storage devices on Windows operating systems in lieu of a CIFS license. Additional use cases include:

- Applications that run on Windows and require NFS connectivity and/or Linux-style commands and functions (such as GETATTR, SETATTR, and so on)
- When a user wants to leverage the NFS protocol rather than CIFS
- Where a user wants to avoid multiprotocol connectivity

Although Windows can be used to leverage NFS connectivity, it might make more sense to use CIFS and the newer features of the latest SMB version that Windows provides for performance and NDO functionality. Additionally, using Windows NFS with ONTAP requires some considerations, covered later.

NFS using Apple OS

NFS mounts are also possible by using Apple OS using the Finder or terminal windows. For the complete mount options in the Apple OS, use the `man mount_nfs` command in a terminal window. When using Apple clients for NFS, there are some things to keep in mind.

Dynamic versus static UIDs

When using a Mac with Active Directory, the default behavior of the Mac is to dynamically create a UID/GID based on the Windows SID of the user. In many cases, this is sufficient, but if control over the UIDs and GIDs is needed (such as integration with an existing LDAP server), then static UIDs can be leveraged. For information about best practices for using Apple OS with Active Directory, see the white paper called Integrate Active Directory using Directory Utility on Mac**.**

Apple OS disables root by default

Apple disables the root user (UID 0) by default in its OS. Users are instead required to log in with a user name other than root and use sudo if performing root-level tasks. It is possible to reenable the root user.

Apple UIDs start at 501

The Apple UID structure starts at UID 501. This UID is not a default UNIX user in ONTAP, nor does it exist in most name service servers. This situation is the same for every Apple OS client in an environment, so it is possible that multiple users exist with the same UID. The options to handle this are as follows:

- Create a user on the cluster or in a name service server with UID 501 to authenticate all Apple users.
- Change the UID on the Apple OS for each user who intends to use NFS on Apple.

Use of Apple NFS with NTFS security-style volumes

Apple NFS handles NTFS security-style volumes differently than Linux NFS clients. Therefore, copies/writes to an NFS mount using Finder applications fail by default when NTFS security style is used. This issue occurs when the Apple client attempts an EXCLUSIVE CREATE operation on the file, which is only allowed by SMB clients in ONTAP.

As a workaround, the NFS server option `-ntfs-unix-security-ops` can be set to ignore to allow NTFS security–style volumes to work properly with NFS mounts on Apple. For more information, see bug 723115.

NFS rootonly operations do not work as expected with Apple OS

In the section called “The rootonly options – nfsrootonly and mountrootonly,” rootonly operations are described. By default, `-mount-rootonly` is enabled, and `-nfs-rootonly` is disabled. Apple OS behavior, when mounting using NFS defaults, is to always use reserved ports for the MOUNT protocol and nonreserved ports for the NFS protocol. The Linux NFS mount option of resvport/noresvport applies in the Apple OS, but noresvport does not control the client’s MOUNT port sent to the cluster. Therefore, Apple NFS clients always use ports in range <1024 for MOUNT.

At present, there is not a known method to change this behavior, so Apple technical support would need to be engaged to use nonreserved ports for NFS MOUNT calls. For NFSv4.x mounts, this does not matter, because NFSv4.x does not leverage the MOUNT protocol. NFS client ports for NFS operations (port 2049) can be controlled using the resvport/noresvport mount options, but the NFS server option on the cluster would need to be toggled to honor the client behavior. Doing so would affect all versions of NFS.

Additionally, when attempting to mount with the resvport option specified in the Apple OS, the `sudo` command would need to be used, because root is disabled and the `-users` option is not specified by default.

Note: When using the Finder to mount NFS, mount options cannot be specified.

Appendix A

The following section contains command examples, configuration output, and other information that would have cluttered the main sections of this document.

Examples

The following section shows examples of commands and of NFS feature functionality.

Examples of controlling the root user

In some cases, storage administrators might want to control how the root user is handled from specific clients. These examples show how the approaches work.

Squashing root

The following examples show how to squash root to anon in various configuration scenarios.

Example 1: Root is squashed to the anon user for all clients.

This approach uses superuser for all NFS clients using sec=sys; other sec types are denied access.

```
cluster::> vserver export-policy rule show -policyname root_squash -instance
  (vserver export-policy rule show)
```

```
                                    Vserver: vs0
                                Policy Name: root_squash
                                 Rule Index: 1
                            Access Protocol: nfs     ← only NFS is allowed (NFSv3 and v4)
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0   ← all clients
                             RO Access Rule: sys     ← only AUTH_SYS is allowed
                             RW Access Rule: sys     ← only AUTH_SYS is allowed
User ID To Which Anonymous Users Are Mapped: 65534   ← mapped to 65534
                   Superuser Security Types: none    ← superuser (root) squashed to anon user
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true

cluster::> volume show -vserver vs0 -volume nfsvol -fields policy
vserver volume policy
------- ------ ------------
vs0     nfsvol root_squash

[root@nfsclient /]# mount -o nfsvers=3 cluster:/nfsvol /mnt
[root@nfsclient /]# cd /mnt

[root@nfsclient mnt]# touch root_squash

[root@nfsclient mnt]# ls -la
total 116
drwxrwxrwx.  3 root   root   106496 Apr 24  2013 .
dr-xr-xr-x. 26 root   root     4096 Apr 24 11:24 ..
drwxr-xr-x.  2 root   daemon   4096 Apr 18 12:54 junction
-rw-r--r--.  1 nobody nobody      0 Apr 24 11:33 root_squash

[root@nfsclient mnt]# ls -lan
drwxrwxrwx.  3     0     0 106496 Apr 24  2013 .
dr-xr-xr-x. 26     0     0   4096 Apr 24 11:24 ..
drwxrwxrwx. 12     0     0   4096 Apr 24 11:05 .snapshot
drwxr-xr-x.  2     0     1   4096 Apr 18 12:54 junction
-rw-r--r--.  1 65534 65534      0 Apr 24  2013 root_squash

[root@nfsclient /]# mount -o sec=krb5 cluster:/nfsvol /mnt
mount.nfs: access denied by server while mounting cluster:/nfsvol
```

Example 2: Root is squashed to the anon user using superuser for a specific client.

In this example, sec=sys and sec=none are allowed.

```
cluster::> vserver export-policy rule show -policyname root_squash_client -instance
  (vserver export-policy rule show)

                                    Vserver: vs0
                                Policy Name: root_squash_client
                                 Rule Index: 1
                            Access Protocol: nfs     ← only NFS is allowed (NFSv3 and v4)
Client Match Hostname, IP Address, Netgroup, or Domain: x.x.x.x     ← just this client
                             RO Access Rule: sys,none       ← AUTH_SYS and AUTH_NONE are allowed
                             RW Access Rule: sys,none       ← AUTH_SYS and AUTH_NONE are allowed
User ID To Which Anonymous Users Are Mapped: 65534   ← mapped to 65534
                   Superuser Security Types: none    ← superuser (root) squashed to anon user
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true

cluster::> volume show -vserver vs0 -volume nfsvol -fields policy
vserver volume policy
------- ------ ------------
vs0     nfsvol root_squash_client

[root@nfsclient /]# mount -o nfsvers=3 cluster:/nfsvol /mnt
[root@nfsclient /]# cd /mnt

[root@nfsclient mnt]# touch root_squash_client

[root@nfsclient mnt]# ls -la
```

```
drwxrwxrwx.  3 root   root   106496 Apr 24  2013 .
dr-xr-xr-x. 26 root   root     4096 Apr 24 11:24 ..
drwxr-xr-x.  2 root   daemon   4096 Apr 18 12:54 junction
-rw-r--r--.  1 nfsnobody nfsnobody      0 Apr 24  2013 root_squash_client

[root@nfsclient mnt]# ls -lan
drwxrwxrwx.  3     0     0 106496 Apr 24  2013 .
dr-xr-xr-x. 26     0     0   4096 Apr 24 11:24 ..
drwxrwxrwx. 12     0     0   4096 Apr 24 11:05 .snapshot
drwxr-xr-x.  2     0     1   4096 Apr 18 12:54 junction
-rw-r--r--.  1 65534 65534      0 Apr 24  2013 root_squash_client
```

Example 3: Root is squashed to the anon user using superuser for a specific set of clients.

This approach uses sec=krb5 (Kerberos) and only NFSv4 and CIFS are allowed.

```
cluster::> vserver export-policy rule show -policyname root_squash_krb5 -instance
  (vserver export-policy rule show)

                                Vserver: vs0
                            Policy Name: root_squash_krb5
                             Rule Index: 1
                          Access Protocol: nfs4,cifs      ← only NFSv4 and CIFS are allowed
Client Match Hostname, IP Address, Netgroup, or Domain: x.x.x.0/24  ← just clients with an IP
address of 10.10.100.X
                           RO Access Rule: krb5    ← only AUTH_RPCGSSD is allowed
                           RW Access Rule: krb5    ← only AUTH_RPCGSSD is allowed
User ID To Which Anonymous Users Are Mapped: 65534   ← mapped to 65534
                  Superuser Security Types: none    ← superuser (root) squashed to anon user
              Honor SetUID Bits in SETATTR: true
                 Allow Creation of Devices: true

cluster::> volume show -vserver vs0 -volume nfsvol -fields policy
vserver volume policy
------- ------ -----------
vs0     nfsvol root_squash

[root@nfsclient /]# mount -o nfsvers=3 cluster:/nfsvol /mnt
mount.nfs: access denied by server while mounting cluster:/nfsvol

[root@nfsclient /]# mount -t nfs4 cluster:/nfsvol /mnt
mount.nfs4: Operation not permitted

[root@nfsclient /]# mount -t nfs4 -o sec=krb5 krbsn:/nfsvol /mnt
[root@nfsclient /]# cd /mnt

[root@nfsclient mnt]# touch root_squash_krb5

[root@nfsclient mnt]# ls -la
drwxrwxrwx.  3 root   root   106496 Apr 24  2013 .
dr-xr-xr-x. 26 root   root     4096 Apr 24 11:24 ..
drwxr-xr-x.  2 root   daemon   4096 Apr 18 12:54 junction
-rw-r--r--.  1 nobody nobody      0 Apr 24 11:50 root_squash_krb5

[root@nfsclient mnt]# ls -lan
drwxrwxrwx.  3  0  0 106496 Apr 24  2013 .
dr-xr-xr-x. 26  0  0   4096 Apr 24 11:24 ..
drwxr-xr-x.  2  0  1   4096 Apr 18 12:54 junction
-rw-r--r--.  1 99 99      0 Apr 24 11:50 root_squash_krb5
```

Note: The UID of 99 in this example occurs in NFSv4 when the user name cannot map into the NFSv4 domain. A look at `/var/log/messages` confirms this:

```
Apr 23 10:54:23 nfsclient nfsidmap[1810]: nss_getpwnam: name 'pcuser' not found in domain
nfsv4domain.netapp.com'
```

In the preceding examples, when the root user requests access to a mount, it maps to the anon UID. In this case, the UID is 65534. This mapping prevents unwanted root access from specified clients to the

NFS share. Because “sys” is specified as the rw and ro access rules in the first two examples, only clients using `sec=sys` gain access. The third example shows a possible configuration using Kerberized NFS authentication. Setting the access protocol to NFS allows only NFS access to the share (including NFSv3 and NFSv4). If multiprotocol access is desired, then the access protocol must be set to allow NFS and CIFS. NFS access can be limited to only NFSv3 or NFSv4 here as well.

Root is root (no_root_squash)

The following examples show how to enable the root user to come into an NFS share as the root user. This is also known as “no_root_squash.”

Example 1: Root is allowed access as root using superuser for all clients only for sec=sys.

In this example, sec=none and sec=sys are allowed rw and ro access; all other anon access is mapped to 65534.

```
cluster::> vserver export-policy rule show -policyname root_allow_anon_squash -instance
  (vserver export-policy rule show)

                                Vserver: vs0
                            Policy Name: root_allow_anon_squash
                             Rule Index: 1
                        Access Protocol: nfs     ← only NFS is allowed (NFSv3 and v4)
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0   ← all clients
                         RO Access Rule: sys,none       ← AUTH_SYS and AUTH_NONE allowed
                         RW Access Rule: sys,none       ← AUTH_SYS and AUTH_NONE allowed
User ID To Which Anonymous Users Are Mapped: 65534   ← mapped to 65534
                  Superuser Security Types: sys     ← superuser for AUTH_SYS only
              Honor SetUID Bits in SETATTR: true

cluster::> volume show -vserver vs0 -volume nfsvol -fields policy
vserver volume policy
------- ------ -----------
vs0     nfsvol root_allow_anon_squash

[root@nfsclient /]# mount -o nfsvers=3 cluster:/nfsvol /mnt
[root@nfsclient /]# cd /mnt

[root@nfsclient mnt]# touch root_allow_anon_squash_nfsv3

[root@nfsclient mnt]# ls -la
drwxrwxrwx.  3 root      root      106496 Apr 24  2013 .
dr-xr-xr-x. 26 root      root        4096 Apr 24 11:24 ..
drwxrwxrwx. 12 root      root        4096 Apr 24 11:05 .snapshot
drwxr-xr-x.  2 root      bin         4096 Apr 18 12:54 junction
-rw-r--r--.  1 root      root           0 Apr 24  2013 root_allow_anon_squash_nfsv3

[root@nfsclient mnt]# ls -lan
drwxrwxrwx.  3  0  0 106496 Apr 24  2013 .
dr-xr-xr-x. 26  0  0   4096 Apr 24 11:24 ..
drwxr-xr-x.  2  0  1   4096 Apr 18 12:54 junction
-rw-r--r--.  1  0  0      0 Apr 24 11:56 root_allow_anon_squash_nfsv3
```

Example 2: Root is allowed access as root using superuser for sec=krb5 only.

In this example, anon access is mapped to 65534; sec=sys and sec=krb5 are allowed, but only using NFSv4.

```
cluster::> vserver export-policy rule show -policyname root_allow_krb5_only -instance
  (vserver export-policy rule show)

                                Vserver: vs0
                            Policy Name: root_allow_krb5_only
                             Rule Index: 1
                        Access Protocol: nfs4    ← only NFSv4 is allowed
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0   ← all clients
```

```
                            RO Access Rule: sys,krb5       ← AUTH_SYS and AUTH_RPCGSS allowed
                            RW Access Rule: sys,krb5       ← AUTH_SYS and AUTH_RPCGSS allowed
User ID To Which Anonymous Users Are Mapped: 65534   ← mapped to 65534
                  Superuser Security Types: krb5    ← superuser via AUTH_RPCGSS only
              Honor SetUID Bits in SETATTR: true

cluster::> volume show -vserver vs0 -volume nfsvol -fields policy
vserver volume policy
------- ------ -----------
vs0     nfsvol root_allow_krb5_only

[root@nfsclient /]# mount -o nfsvers=3 cluster:/nfsvol /mnt
mount.nfs: access denied by server while mounting cluster:/nfsvol

[root@nfsclient /]# mount -t nfs4 cluster:/nfsvol /mnt
[root@nfsclient /]# cd /mnt

[root@nfsclient mnt]# touch root_allow_krb5_only_notkrb5

[root@nfsclient mnt]# ls -la
drwxrwxrwx.  3 root   root   106496 Apr 24  2013 .
dr-xr-xr-x. 26 root   root     4096 Apr 24 11:24 ..
drwxr-xr-x.  2 root   daemon   4096 Apr 18 12:54 junction
-rw-r--r--.  1 nobody nobody      0 Apr 24  2013 root_allow_krb5_only_notkrb5

[root@nfsclient mnt]# ls -lan
drwxrwxrwx.  3  0  0 106496 Apr 24  2013 .
dr-xr-xr-x. 26  0  0   4096 Apr 24 11:24 ..
drwxr-xr-x.  2  0  1   4096 Apr 18 12:54 junction
-rw-r--r--.  1 99 99      0 Apr 24  2013 root_allow_krb5_only_notkrb5


NOTE: Again, the UID of an unmapped user in NFSv4 is 99. This is controlled via /etc/idmapd.conf
in Linux and /etc/default/nfs in Solaris.

[root@nfsclient /]# mount -t nfs4 -o sec=krb5 cluster:/nfsvol /mnt
[root@nfsclient /]# kinit
Password for root@KRB5DOMAIN.NETAPP.COM:
[root@nfsclient /]# cd /mnt

[root@nfsclient mnt]# touch root_allow_krb5_only_krb5mount

[root@nfsclient mnt]# ls -la
drwxrwxrwx.  3 root   root   106496 Apr 24  2013 .
dr-xr-xr-x. 26 root   root     4096 Apr 24 11:24 ..
drwxr-xr-x.  2 root   daemon   4096 Apr 18 12:54 junction
-rw-r--r--.  1 root   daemon      0 Apr 24  2013 root_allow_krb5_only_krb5mount

[root@nfsclient mnt]# ls -lan
drwxrwxrwx.  3  0  0 106496 Apr 24  2013 .
dr-xr-xr-x. 26  0  0   4096 Apr 24 11:24 ..
drwxr-xr-x.  2  0  1   4096 Apr 18 12:54 junction
-rw-r--r--.  1  0  1      0 Apr 24  2013 root_allow_krb5_only_krb5mount
```

Example 3: Root and all anonymous users are allowed access as root using anon=0.

This example allows only for sec=sys and sec=krb5 over NFSv4.

```
cluster::> vserver export-policy rule show -policyname root_allow_anon0 -instance
  (vserver export-policy rule show)

                                Vserver: vs0
                            Policy Name: root_allow_anon0
                             Rule Index: 1
                        Access Protocol: nfs4    ← only NFSv4 is allowed
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0   ← all clients
                            RO Access Rule: krb5, sys      ← AUTH_SYS and AUTH_RPCGSS allowed
                            RW Access Rule: krb5, sys      ← AUTH_SYS and AUTH_RPCGSS allowed
User ID To Which Anonymous Users Are Mapped: 0       ← mapped to 0
                  Superuser Security Types: none    ← superuser (root) squashed to anon user
```

```
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true


cluster::> volume show -vserver vs0 -volume nfsvol -fields policy
vserver volume policy
------- ------ ------------
vs0     nfsvol root_allow_anon0

[root@nfsclient /]# mount -o nfsvers=3 cluster:/nfsvol /mnt
mount.nfs: access denied by server while mounting cluster:/nfsvol

[root@nfsclient /]# mount -t nfs4 cluster:/nfsvol /mnt
[root@nfsclient /]# cd /mnt

[root@nfsclient mnt]# touch root_allow_anon0

[root@nfsclient mnt]# ls -la
drwxrwxrwx.  3 root   root   106496 Apr 24  2013 .
dr-xr-xr-x. 26 root   root     4096 Apr 24 11:24 ..
drwxr-xr-x.  2 root   daemon   4096 Apr 18 12:54 junction
-rw-r--r--.  1 root   daemon      0 Apr 24  2013 root_allow_anon0

[root@nfsclient mnt]# ls -lan
drwxrwxrwx.  3  0  0 106496 Apr 24  2013 .
dr-xr-xr-x. 26  0  0   4096 Apr 24 11:24 ..
drwxr-xr-x.  2  0  1   4096 Apr 18 12:54 junction
-rw-r--r--.  1  0  1      0 Apr 24  2013 root_allow_anon0


[root@nfsclient /]# mount -t nfs4 -o sec=krb5 cluster:/nfsvol /mnt
[root@nfsclient /]# cd /mnt

[root@nfsclient mnt]# touch root_allow_anon0_krb5

[root@nfsclient mnt]# ls -la
drwxrwxrwx.  3 root   root   106496 Apr 24  2013 .
dr-xr-xr-x. 26 root   root     4096 Apr 24 11:24 ..
drwxr-xr-x.  2 root   daemon   4096 Apr 18 12:54 junction
-rw-r--r--.  1 root   daemon      0 Apr 24  2013 root_allow_anon0_krb5

[root@nfsclient mnt]# ls -lan
drwxrwxrwx.  3  0  0 106496 Apr 24  2013 .
dr-xr-xr-x. 26  0  0   4096 Apr 24 11:24 ..
drwxr-xr-x.  2  0  1   4096 Apr 18 12:54 junction
-rw-r--r--.  1  0  1      0 Apr 24  2013 root_allow_anon0_krb5
```

Example of creating load sharing mirrors for vsroot

This example shows how to create load-sharing mirrors for the vsroot volume. Use of load-sharing mirrors applies to NFSv3 and SMB only.

```
cluster::*> vol show -vserver NFS -vsroot true -fields size
vserver volume size
------- ------ ----
NFS     vsroot 1GB

cluster::*> vol create -vserver NFS -volume vsroot_mirror1 -aggregate aggr1_node1 -size 1g -type
DP
[Job 26705] Job succeeded: Successful

cluster::*> vol create -vserver NFS -volume vsroot_mirror2 -aggregate aggr1_node2 -size 1g -type
DP
[Job 26707] Job succeeded: Successful

cluster::*> snapmirror create -source-path NFS:vsroot -destination-path NFS:vsroot_mirror1 -
vserver NFS -type LS -schedule daily
[Job 26709] Job succeeded: SnapMirror: done
```

```
cluster::*> snapmirror create -source-path NFS:vsroot -destination-path NFS:vsroot_mirror2 -
vserver NFS -type LS -schedule daily
[Job 26711] Job succeeded: SnapMirror: done

cluster::*> snapmirror show -source-path NFS:vsroot -fields schedule,state
source-path                 destination-path                    schedule state
--------------------------- ----------------------------------- -------- ------------
cluster://NFS/vsroot        cluster://NFS/vsroot_mirror1 daily    -       -
cluster://NFS/vsroot        cluster://NFS/vsroot_mirror2 daily    -       -
2 entries were displayed.

cluster::*> snapmirror initialize-ls-set -source-path NFS:vsroot
[Job 26714] Job is queued: snapmirror initialize-ls-set for source " cluster://NFS/vsroot".

cluster::*> snapmirror show -source-path NFS:vsroot -fields schedule,state
source-path                 destination-path                    schedule state
--------------------------- ----------------------------------- -------- ------------
cluster://NFS/vsroot        cluster://NFS/vsroot_mirror1        daily    Snapmirrored
cluster://NFS/vsroot        cluster://NFS/vsroot_mirror2        daily    Snapmirrored
2 entries were displayed.
```

Export policy rule inheritance example

In the following example, the SVM root volume has limited superuser access only to the client `x.x.x.x`. When the root user attempts to access a mount from a client other than `x.x.x.x`, it squashes to the anon user, which is 65534:

```
cluster::> vol show -vserver nfs_svm -volume rootvol -fields policy
  (volume show)
vserver volume  policy
------- ------- -------
nfs_svm rootvol default

cluster::> export-policy rule show -vserver nfs_svm -policyname default -instance
  (vserver export-policy rule show)

                                Vserver: nfs_svm
                            Policy Name: default
                             Rule Index: 1
                          Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: x.x.x.x
                           RO Access Rule: any
                           RW Access Rule: any
User ID To Which Anonymous Users Are Mapped: 65534
                 Superuser Security Types: any
             Honor SetUID Bits in SETATTR: true
                Allow Creation of Devices: true

                                Vserver: nfs_svm
                            Policy Name: default
                             Rule Index: 2
                          Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0
                           RO Access Rule: any
                           RW Access Rule: any
User ID To Which Anonymous Users Are Mapped: 65534
                 Superuser Security Types: none
             Honor SetUID Bits in SETATTR: true
                Allow Creation of Devices: true
2 entries were displayed.
```

As per the example in the section “Limiting Access to the SVM Root Volume,” root would not be able to list the contents of the SVM root based on the volume permissions (711) and the existing export policy rules on any hosts other than x.x.x.x.

```
# ifconfig | grep "inet addr"
          inet addr:x.x.x.y  Bcast:x.x.225.255  Mask:255.255.255.0
          inet addr:127.0.0.1  Mask:255.0.0.0
```

```
# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023
# mount | grep mnt
x.x.x.e:/ on /mnt type nfs (rw,nfsvers=3,addr=x.x.x.e)
# cd /mnt
# ls
ls: cannot open directory .: Permission denied
```

If the data volumes in the SVM also are set to this export policy, they use the same rules, and only the client set to have root access is able to log in as root.

If root access is desired to the data volumes, then a new export policy can be created and root access can be specified for all hosts or a subset of hosts through subnet, netgroup, or multiple rules with individual client IP addresses or host names.

The same concept applies to the other export policy rule attributes, such as RW.

For example, if the default export policy rule is changed to disallow write access to all clients except `x.x.x.x` and to allow superuser, then even root is disallowed write access to volumes with that export policy applied:

```
cluster::> export-policy rule modify -vserver nfs_svm -policyname default -ruleindex 2 -rwrule
never -superuser any

cluster::> export-policy rule show -vserver nfs_svm -policyname default -instance
  (vserver export-policy rule show)

                                Vserver: nfs_svm
                            Policy Name: default
                             Rule Index: 1
                        Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: x.x.x.x
                          RO Access Rule: any
                          RW Access Rule: any
User ID To Which Anonymous Users Are Mapped: 65534
                   Superuser Security Types: any
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true

                                Vserver: nfs_svm
                            Policy Name: default
                             Rule Index: 2
                        Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0
                          RO Access Rule: any
                          RW Access Rule: never
User ID To Which Anonymous Users Are Mapped: 65534
                   Superuser Security Types: any
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true
2 entries were displayed.

# ifconfig | grep "inet addr"
          inet addr:x.x.x.y  Bcast:x.x.225.255  Mask:255.255.255.0
          inet addr:127.0.0.1  Mask:255.0.0.0
# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023
# mount | grep mnt
x.x.x.e:/ on /mnt type nfs (rw,nfsvers=3,addr=x.x.x.e)
# cd /mnt
# touch rootfile
touch: cannot touch `rootfile': Read-only file system
```

When a new policy and rule are created and applied to the data volume, the same user is allowed to write to the data volume mounted below the SVM root volume. This is the case despite the export policy rule at the SVM root volume disallowing write access.

Example:

```
cluster::> export-policy create -vserver nfs_svm -policyname volume
cluster::> export-policy rule create -vserver nfs_svm -policyname volume -clientmatch 0.0.0.0/0 -
rorule any -rwrule any -allow-suid true -allow-dev true -ntfs-unix-security-ops fail -chown-mode
restricted -superuser any -protocol any -ruleindex 1 -anon 65534

cluster::> export-policy rule show -vserver nfs_svm -policyname volume -instance
  (vserver export-policy rule show)

                                Vserver: nfs_svm
                            Policy Name: volume
                             Rule Index: 1
                        Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0
                          RO Access Rule: any
                          RW Access Rule: any
User ID To Which Anonymous Users Are Mapped: 65534
                  Superuser Security Types: any
              Honor SetUID Bits in SETATTR: true
                 Allow Creation of Devices: true

::> volume modify -vserver flexvol -volume unix -policy volume
```

From the client:

```
# ifconfig | grep "inet addr"
          inet addr:x.x.x.y  Bcast:x.x.225.255  Mask:255.255.255.0
          inet addr:127.0.0.1  Mask:255.0.0.0
# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023
# cd /mnt/unix
[root@linux-client unix]# ls
file
[root@linux-client unix]# touch rootfile
[root@linux-client unix]# ls -la | grep rootfile
-rw-r--r--. 1 root root     0 Apr  1  2014 rootfile
# cd ..
# ls
nfs4  ntfs  unix
# touch rootdir
touch: cannot touch `rootdir': Read-only file system
```

However, the read-only attribute for the export policy rules needs to allow read access from the parent to allow mounts to occur. Setting `rorule` to `never` or not setting an export policy rule in the parent volume's export policy (empty policy) disallows mounts to volumes underneath that parent.

In the following example, the vsroot volume has an export policy that has `rorule` and `rwrule` set to `never`, while the data volume has an export policy with a rule that is wide open:

```
cluster::> export-policy rule show -vserver nfs -policyname wideopen -instance
  (vserver export-policy rule show)

                                Vserver: nfs
                            Policy Name: wideopen
                             Rule Index: 1
                        Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0
                          RO Access Rule: any
                          RW Access Rule: any
User ID To Which Anonymous Users Are Mapped: 0
                  Superuser Security Types: any
              Honor SetUID Bits in SETATTR: true
                 Allow Creation of Devices: true

cluster::> export-policy rule show -vserver nfs -policyname deny -instance
  (vserver export-policy rule show)
```

```
                                    Vserver: nfs
                                Policy Name: deny
                                 Rule Index: 1
                            Access Protocol: any
Client Match Hostname, IP Address, Netgroup, or Domain: 0.0.0.0/0
                             RO Access Rule: never
                             RW Access Rule: never
User ID To Which Anonymous Users Are Mapped: 65534
                   Superuser Security Types: sys
               Honor SetUID Bits in SETATTR: true
                  Allow Creation of Devices: true

cluster::> volume show -vserver nfs -volume rootvol -fields policy,unix-permissions
vserver volume  policy unix-permissions
------- ------- ------ ----------------
nfs     rootvol deny   ---rwx--x-x

cluster::> volume show -vserver nfs -volume unix -fields policy,unix-permissions
vserver volume policy   unix-permissions
------- ------ -------- ----------------
nfs     unix   wideopen ---rwxrwxrwx
```

When a mount of the volume UNIX is attempted, access is denied.

```
# mount -o nfsvers=3 x.x.x.e:/unix /cdot
mount.nfs: access denied by server while mounting x.x.x.e:/unix
```

When the deny policy is changed to allow read-only access, mounting is allowed.

```
cluster::> export-policy rule modify -vserver nfs -policyname deny -rorule any -ruleindex 1

# mount -o nfsvers=3 x.x.x.e:/unix /cdot
# mount | grep unix
x.x.x.e:/unix on /cdot type nfs (rw,nfsvers=3,addr=x.x.x.e)
```

As a result, storage administrators can have complete and granular control over what users see and access file systems using export policies, rules, and volume permissions.

Export policy rule index example

In the following example, a client with the IP address of `x.x.x.x` (host name of `centos64`) has been denied access to read a volume while all other clients are allowed access. However, the client rule is below the All Access rule, so mount and read are allowed.

Example:

```
cluster::> export-policy rule show -vserver NAS -policyname allow_all
             Policy          Rule    Access   Client                RO
Vserver      Name            Index   Protocol Match                 Rule
------------ --------------- ------  -------- --------------------- ---------
NAS          allow_all       1       any      0.0.0.0/0             any
NAS          allow_all       99      any      x.x.x.x        never
2 entries were displayed.

cluster::> vol show -vserver NAS -volume unix -fields policy
vserver volume policy
------- ------ ---------
NAS     unix   allow_all

[root@centos64 ~]# mount -o nfsvers=3 x.x.x.a:/vol/nfs /7mode
[root@centos64 ~]# mount x.x.x.a:/unix /mnt
[root@centos64 ~]# cd /mnt
[root@centos64 mnt]# ls -la
total 12
drwxrwxrwx.  3 root root    4096 Dec 10 14:49 .
dr-xr-xr-x. 46 root root    4096 Dec 10 14:57 ..
drwxrwx---.  2 root root 4096 Dec 10 15:00 file
```

If those rules are flipped, the client is denied access despite the rule allowing access to everyone being in the policy. Rule index numbers can be modified with the `export-policy rule setindex` command. In the following example, rule #1 was changed to rule #99. Rule #99 gets moved to #98 by default.

```
cluster::> export-policy rule setindex -vserver NAS -policyname allow_all -ruleindex 1 -
newruleindex 99

cluster::> export-policy rule show -vserver NAS -policyname allow_all
             Policy          Rule    Access   Client                RO
Vserver      Name            Index   Protocol Match                 Rule
------------ --------------- ------  -------- --------------------- ---------
NAS          allow_all       98      any      x.x.x.x        never
NAS          allow_all       99      any      0.0.0.0/0             any
2 entries were displayed.

cluster::> export-policy cache flush -vserver NAS -cache all

Warning: You are about to flush the "all (but showmount)" cache for Vserver "NAS" on node
"node2", which will result in increased traffic to the name servers. Do you want to proceed with
         flushing the cache? {y|n}: y


[root@centos64 /]# mount x.x.x.a:/unix /mnt
mount.nfs: access denied by server while mounting x.x.x.a:/unix
```

NFSv4.x ACL Explicit DENY example

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@nfsclient /]# nfs4_getfacl /mixed
A::OWNER@:rwaDxtTnNcCy
D::OWNER@:
A:g:GROUP@:rwaDxtTnNcy
D:g:GROUP@:C
A::EVERYONE@:rxtncy
D::EVERYONE@:waDTC

[root@nfsclient /]# 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:

```
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 it just had access to write to:

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

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

[root@nfsclient /]# 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
```

NFSv4.x ACL preservation example

This is a newly created UNIX-style volume:

```
cluster::> volume show -vserver vs0 -volume unix -fields security-style,
unix-permissions,user,group
vserver volume user group security-style unix-permissions
------- ------ ---- ----- -------------- ----------------
vs0     unix   0    1     unix           ---rwxr-xr-x

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

                Vserver: vs0
              File Path: /unix
         Security Style: unix
        Effective Style: unix
         DOS Attributes: 10
 DOS Attributes in Text: ----D---
Expanded Dos Attributes: -
           Unix User Id: 0
          Unix Group Id: 1
```

```
        Unix Mode Bits: 755
 Unix Mode Bits in Text: rwxr-xr-x
                  ACLs: -
```

In the preceding example, the volume (/unix) has 755 permissions. That means that 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@nfsclient /]# mount -t nfs4 krbsn:/unix /unix
[root@nfsclient /]# ls -la | grep unix
drwxr-xr-x.   2 root     daemon         4096 Apr 30 11:24 unix
[root@nfsclient /]# nfs4_getfacl /unix
A::OWNER@:rwaDxtTnNcCy
A:g:GROUP@:rxtncy
A::EVERYONE@:rxtncy
```

The NFSv4 ACLs earlier 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@nfsclient /]# chmod 775 /unix
[root@nfsclient /]# ls -la | grep unix
drwxrwxr-x.   2 root     daemon         4096 Apr 30 11:24 unix
[root@nfsclient /]# nfs4_getfacl /unix
A::OWNER@:rwaDxtTnNcCy
A:g:GROUP@:rwaDxtTnNcy
A::EVERYONE@:rxtncy
```

When a user ACE is added to the ACL, the entry is reflected in the ACL on the appliance. In addition, the entire ACL is now populated. Note that the ACL is in SID format.

```
[root@nfsclient /]# nfs4_setfacl -a A::ldapuser@nfsv4domain.netapp.com:ratTnNcCy /unix
[root@nfsclient /]# nfs4_getfacl /unix
A::ldapuser@nfsv4domain.netapp.com:ratTnNcCy
A::OWNER@:rwaDxtTnNcCy
A:g:GROUP@:rwaDxtTnNcy
A::EVERYONE@:rxtncy

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

               Vserver: vs0
             File Path: /unix
        Security Style: unix
       Effective Style: unix
        DOS Attributes: 10
 DOS Attributes in Text: ----D---
Expanded Dos Attributes: -
          Unix User Id: 0
         Unix Group Id: 1
        Unix Mode Bits: 775
 Unix Mode Bits in Text: rwxrwxr-x
                  ACLs: NFSV4 Security Descriptor
                        Control:0x8014
                        DACL - ACEs
                          ALLOW-S-1-8-55-0x16019d
                          ALLOW-S-1-520-0-0x1601ff
                          ALLOW-S-1-520-1-0x1201ff-IG
                          ALLOW-S-1-520-2-0x1200a9
```

To see the translated ACLs, use fsecurity from the node shell on the node that owns the volume:

```
cluster::> node run -node node2 fsecurity show /vol/unix
```

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

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

  Unix security:
    uid: 0
    gid: 1
    mode: 0775 (rwxrwxr-x)

  NFSv4 security descriptor:
    DACL:
      Allow - uid: 55 - 0x0016019d
      Allow - OWNER@ - 0x001601ff
      Allow - GROUP@ - 0x001201ff
      Allow - EVERYONE@ - 0x001200a9 (Read and Execute)
    SACL:
      No entries.
```

When a change is made to the mode bit when NFSv4 ACLs are present, the NFSv4 ACL that was just set is wiped by default:

```
[root@nfsclient /]# chmod 755 /unix
[root@nfsclient /]# ls -la | grep unix
drwxr-xr-x.   2 root     daemon         4096 Apr 30 11:24 unix
[root@nfsclient /]# nfs4_getfacl /unix
A::OWNER@:rwaDxtTnNcCy
A:g:GROUP@:rxtncy
A::EVERYONE@:rxtncy

cluster::> node run -node node2 fsecurity show /vol/unix

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

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

  Unix security:
    uid: 0
    gid: 1
    mode: 0755 (rwxr-xr-x)

  No security descriptor available.
```

To control this behavior in ONTAP, use the following diag-level option:

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

After the option is enabled, the ACL stays intact when mode bits are set.

```
[root@nfsclient /]# nfs4_setfacl -a A::ldapuser@nfsv4domain.netapp.com:ratTnNcCy /unix
[root@nfsclient /]# ls -la | grep unix
drwxr-xr-x.   2 root     daemon         4096 Apr 30 11:24 unix
[root@nfsclient /]# nfs4_getfacl /unix
A::ldapuser@nfsv4domain.netapp.com:ratTnNcCy
A::OWNER@:rwaDxtTnNcCy
A:g:GROUP@:rxtncy
A::EVERYONE@:rxtncy

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

               Vserver: vs0
             File Path: /unix
         Security Style: unix
        Effective Style: unix
```

```
         DOS Attributes: 10
 DOS Attributes in Text: ----D---
Expanded Dos Attributes: -
           Unix User Id: 0
          Unix Group Id: 1
         Unix Mode Bits: 755
 Unix Mode Bits in Text: rwxr-xr-x
                   ACLs: NFSV4 Security Descriptor
                         Control:0x8014
                         DACL - ACEs
                           ALLOW-S-1-8-55-0x16019d
                           ALLOW-S-1-520-0-0x1601ff
                           ALLOW-S-1-520-1-0x1200a9-IG
                           ALLOW-S-1-520-2-0x1200a9

cluster::> node run -node node2 fsecurity show /vol/unix

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

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

  Unix security:
    uid: 0
    gid: 1
    mode: 0755 (rwxr-xr-x)

  NFSv4 security descriptor:
    DACL:
      Allow - uid: 55 - 0x0016019d
      Allow - OWNER@ - 0x001601ff
      Allow - GROUP@ - 0x001200a9 (Read and Execute)
      Allow - EVERYONE@ - 0x001200a9 (Read and Execute)
    SACL:
      No entries.
```

Note that the ACL is still intact after mode bits are set.

```
[root@nfsclient /]# chmod 777 /unix
[root@nfsclient /]# ls -la | grep unix
drwxrwxrwx.   2 root     daemon         4096 Apr 30 11:24 unix
[root@nfsclient /]# nfs4_getfacl /unix
A::ldapuser@win2k8.ngslabs.netapp.com:ratTnNcCy
A::OWNER@:rwaDxtTnNcCy
A:g:GROUP@:rwaDxtTnNcy
A::EVERYONE@:rwaDxtTnNcy

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

               Vserver: vs0
             File Path: /unix
         Security Style: unix
        Effective Style: unix
         DOS Attributes: 10
 DOS Attributes in Text: ----D---
Expanded Dos Attributes: -
           Unix User Id: 0
          Unix Group Id: 1
         Unix Mode Bits: 777
 Unix Mode Bits in Text: rwxrwxrwx
                   ACLs: NFSV4 Security Descriptor
                         Control:0x8014
                         DACL - ACEs
                           ALLOW-S-1-8-55-0x16019d
                           ALLOW-S-1-520-0-0x1601ff
                           ALLOW-S-1-520-1-0x1201ff-IG
                           ALLOW-S-1-520-2-0x1201ff

cluster::*> node run -node node2 fsecurity show /vol/unix
```

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

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

  Unix security:
    uid: 0
    gid: 1
    mode: 0777 (rwxrwxrwx)

  NFSv4 security descriptor:
    DACL:
      Allow - uid: 55 - 0x0016019d
      Allow - OWNER@ - 0x001601ff
      Allow - GROUP@ - 0x001201ff
      Allow - EVERYONE@ - 0x001201ff
    SACL:
      No entries.
```

NFS audit event example

The following example is from an audit event using NFSv3:

```
- <Event>
- <System>
  <Provider Name="Netapp-Security-Auditing" />
  <EventID>4663</EventID>
  <EventName>Get Object Attributes</EventName>
  <Version>1</Version>
  <Source>NFSv3</Source>
  <Level>0</Level>
  <Opcode>0</Opcode>
  <Keywords>0x8020000000000000</Keywords>
  <Result>Audit Success</Result>
  <TimeCreated SystemTime="2013-08-08T20:36:05.011243000Z" />
  <Correlation />
  <Channel>Security</Channel>
  <Computer>e284de25-3edc-11e2-92d0-123478563412/525c9a2c-dce2-11e2-b94f-123478563412</Computer>
  <Security />
  </System>
- <EventData>
  <Data Name="SubjectIP" IPVersion="4">x.x.x.x</Data>
  <Data Name="SubjectUnix" Uid="10000" Gid="503" Local="false" />
  <Data Name="ObjectServer">Security</Data>
  <Data Name="ObjectType">Directory</Data>
  <Data Name="HandleID">00000000000453;00;00000040;3a2cada4</Data>
  <Data Name="ObjectName">/</Data>
  <Data Name="InformationRequested">File Type; File Size; Last Accessed Time; Last Metadata
Modfied Time; Last Modified Time; Unix Mode; Unix Owner; Unix Group;</Data>
  </EventData>
  </Event>
```

NFSv4.x referral example

The data volume lives on node1.

```
cluster::> volume show -vserver vs0 -volume nfsvol -fields node
vserver volume node
------- ------ -------------
vs0     nfsvol node1
```

The data LIF lives on node2.

```
cluster::> net int show -vserver vs0 -lif data2 -fields curr-node,home-node
  (network interface show)
vserver lif   home-node     curr-node     address
```

```
------- ----- ------------ ------------- -----------
vs0     data2 node2          node2         x.x.x.a
```

There is also a data LIF on node1.

```
cluster::> net int show -vserver vs0 -curr-node node1 -role data
  (network interface show)
            Logical    Status     Network            Current       Current Is
Vserver     Interface  Admin/Oper Address/Mask       Node          Port    Home
----------- ---------- ---------- ------------------ ------------- ------- ----
vs0
            data1         up/up    x.x.x.a/24     node1        e0a     true
```

The client makes a mount request to the data LIF on node2, at the IP address `x.x.x.a`.

```
[root@nfsclient /]# mount -t nfs4 x.x.x.a:/nfsvol /mnt
```

The mount location looks to be at the IP address specified by the client.

```
[root@nfsclient /]# mount | grep /mnt
x.x.x.a:/nfsvol on /mnt type nfs4 (rw,addr=x.x.x.a,clientaddr=x.x.x.z)
```

But the cluster shows that the connection was actually established to node1, where the data volume lives. No connection was made to node2.

```
cluster::> network connections active show -node node1 -service nfs*
              Vserver   Interface         Remote
      CID Ctx Name      Name:Local Port   Host:Port            Protocol/Service
--------- --- --------- ----------------- -------------------- ----------------
Node: node1
286571835   6 vs0          data:2049         x.x.x.z:763    TCP/nfs

cluster::> network connections active show -node node2 -service nfs*
There are no entries matching your query.
```

NFS connected-clients sample output

The following shows sample output of the NFS connected-clients command:

```
cluster::*> nfs connected-clients show -vserver DEMO

     Node: node1
  Vserver: DEMO
  Data-Ip: 10.x.x.a
Client-Ip       Volume-Name      Protocol Idle-Time     Local-Reqs Remote-Reqs
--------------- ---------------- -------- ------------- ---------- -----------
10.x.x.b        scripts          nfs3     1h 3m 46s     391        0
10.x.x.c        XCP_catalog      nfs3     17s           0          372
10.x.x.d        scripts          nfs3     17s           372        0
10.x.x.c        home             nfs4     12h 50m 28s   256        0
10.x.x.b        home             nfs4.1   9h 51m 48s    47         0
```

Performance comparison: NFSv3 and NFSv4 using nconnect and pNFS

In our CPOC labs, real-world workloads are run in a controlled lab environment to illustrate the potential performance and feature benefits of NetApp products.

In one such CPOC lab, a sequential read workload was tested by using the following components:

- Single NVIDIA DGX-2 client
- Ubuntu 20.04.2
- NFSv4.1 with pNFS and nconnect
- AFF A400 cluster

- NetApp FlexGroup volume
- 256K wsize/rsize
- 100GbE connections
- 32x 1GB files

In these tests, the following results were seen. Latency for both were sub 1ms.

Table 32) NFSv4.1/pNFS/nconnect vs. NFSv3 — sequential reads.

Test	Bandwidth
NFSv3	10.2GBps
NFSv4.1/pNFS	21.9GBps

Note: Both NFSv3 and NFSv4.1 used nconnect=16.

In these tests, NFSv4.1 with pNFS doubled the performance for the sequential read workload at 250us latency. Because the files were 1GB in size, the reads were almost entirely from the controller RAM.

Some examples of sequential workloads include:

- Hadoop
- VMware
- Database workloads
- High-performance computing workloads
- Media workloads

If you have a sequential workload, consider NFSv4.1, pNFS and nconnect for the best performance experience. Remember that workloads with large amounts of getattrs/high metadata (such as EDA/software build) will likely not see the same performance gains as these types of workloads. If you want to find out more about the CPOC labs, contact your NetApp sales team.

Note: Oracle workloads can also benefit from NFSv4.1 and pNFS, but instead of using nconnect, use dNFS for the best overall results. See TR-3633: Oracle Databases on ONTAP for details and listen to the Tech ONTAP Podcast Episode 279: NetApp and Oracle – What's New.

NFSv3 vs. NFSv4.x — Performance comparisons

To accurately understand the impact on performance between NFSv3 and NFSv4.x in your environment, testing is essential. Not all workloads perform the same with each protocol and NFSv4.x is substantially different from NFSv3 in a variety of ways including locking, state fullness, metadata handling, and compound operations, to name a few. A workload in NFSv3 might have a very different profile than the same workload using NFSv4.x. For example, a high file create workload (1000 directories, 1 million files across those directories) using ONTAP 9.9.1 NFSv3 produced statistics that looked like this — roughly an even split of creates, lookups, and writes.

```
Object: nfsv3
Instance: DEMO
    Counter                                                     Value
    -------------------------------- --------------------------------
    access_total                                                 1014
    create_percent                                                33%
    create_total                                              1000000
    fsinfo_total                                                    4
    getattr_total                                                   3
    lookup_percent                                                33%
    lookup_total                                              1000003
    mkdir_total                                                  1003
    null_total                                                      4
    pathconf_total                                                  2
    total_ops                                                   31936
```

```
    write_percent                                          33%
    write_total                                        1000000
```

The same exact workload using NFSv4.1 produces a very different statistical profile. Because NFSv4.x operates very differently than NFSv3 for file creates, you see the following:

- Higher metadata (15%)
- Fewer CREATE operations (1000 versus. 1000000 with NFSv3)
- OPEN/CLOSE operations for each file creation
- Several other metadata operation types (COMPOUND, SEQUENCE, GETFH, PUTFH)
- Write totals stay the same but percentage wise, are much lower (7%)

```
Object: nfsv4_1
Instance: DEMO
    Counter                                                Value
    -------------------------------- --------------------------------
    access_percent                                            7%
    access_total                                         1001014
    close_percent                                             7%
    close_total                                          1000000
    compound_percent                                         23%
    compound_total                                       3002078
    create_session_total                                       2
    create_total                                            1003
    exchange_id_total                                          8
    getattr_percent                                          15%
    getattr_total                                        2002064
    getfh_percent                                             7%
    getfh_total                                          1001012
    lookup_total                                               7
    open_percent                                              7%
    open_total                                           1000000
    putfh_percent                                            23%
    putfh_total                                          3002064
    putrootfh_total                                            2
    reclaim_complete_total                                     2
    sequence_percent                                         23%
    sequence_total                                       3002068
    total_ops                                               6417
    write_percent                                             7%
    write_total                                          1000000
```

As a result, just by using NFSv4.1 for this workload, there are more overall metadata operations, which NFSv4.1 tends to perform poorly with. This is illustrated in a side-by-side comparison of various performance metrics, as shown in Table 33.

Table 33) NFSv3 versus NFSv4.1 performance — High file creation workload

Test	Average IOPS	Average MBps	Average latency	Completion time	Average CPU %
NFSv3	55118	71.9	1.6ms	54.7 seconds	51%
NFSv4.1	25068	13.9	11.5ms	283.5 seconds	24%

As you can see, the latency is higher, IOPS and throughput is lower, and the completion time is nearly five times greater for this workload when using NFSv4.1. The CPU usage is lower because the storage is not being asked to do as much (such as, process as many IOPS).

This does not mean NFSv4.x always performs worse than NFSv3; in some cases, it can perform as well or better. It all depends on the type of workload being used.

For a highly sequential write workload, NFSv4.1 is able to compete with NFSv3 because there is less metadata to process. Using a multithreaded dd operation to create eight 10GB files, this is the NFSv3 workload profile:

```
Object: nfsv3
Instance: DEMO
    Counter                                                     Value
    -------------------------------- --------------------------------
    access_total                                                   18
    create_total                                                    8
    fsinfo_total                                                    4
    getattr_total                                                   7
    lookup_total                                                   11
    mkdir_total                                                    11
    null_total                                                      4
    pathconf_total                                                  2
    total_ops                                                    5357
    write_percent                                                  99%
    write_total                                                1248306
```

In this case, it is at nearly 100% writes. For NFSv4.1, the write percentage is lower but the accompanying metadata operations are not the types that incur performance penalties (COMPOUND and PUTFH).

```
Object: nfsv4_1
Instance: DEMO
    Counter                                                     Value
    -------------------------------- --------------------------------
    access_total                                                   25
    close_total                                                     4
    compound_percent                                               33%
    compound_total                                             1238160
    create_session_total                                            4
    create_total                                                   11
    destroy_clientid_total                                          3
    destroy_session_total                                           3
    exchange_id_total                                              16
    getattr_total                                                  72
    getdeviceinfo_total                                             8
    getfh_total                                                    28
    layoutget_total                                                 8
    layoutreturn_total                                              1
    lookup_total                                                    7
    open_total                                                      8
    putfh_percent                                                  33%
    putfh_total                                                1238107
    putrootfh_total                                                 2
    reclaim_complete_total                                          4
    sequence_percent                                               33%
    sequence_total                                             1238134
    total_ops                                                    15285
    write_percent                                                  33%
    write_total                                                1238032
```

This results in a much better performance comparison for NFSv4.1. IOPS, throughput, and latency are nearly identical and NFSv4.1 takes 8 seconds longer (~3.7% more) than NFSv3, as shown in Table 34.

Table 34) NFSv3 vs. NFSv4.1 performance — High sequential writes

Test	Average IOPS	Average MBps	Average latency	Completion time	Average CPU %
NFSv3	6085	383.1	5.4ms	216.6 seconds	28%
NFSv4.1	6259	366.3	4.4	224.7 seconds	26%

With a more standard benchmarking tool (vdbench), you can see the differences between read and write performance with different workload types. NFSv3 performs better in most cases but the gap is not especially wide for workloads that read/write existing datasets. Writes tend to have a wider performance disparity but read performance is nearly identical. Sequential writes actually perform a bit better for NFSv4.x in these tests as well, as shown in the following figures.

Figure 23) Random reads, 4K, NFSv3 versus NFSv4.x — IOPS/Latency

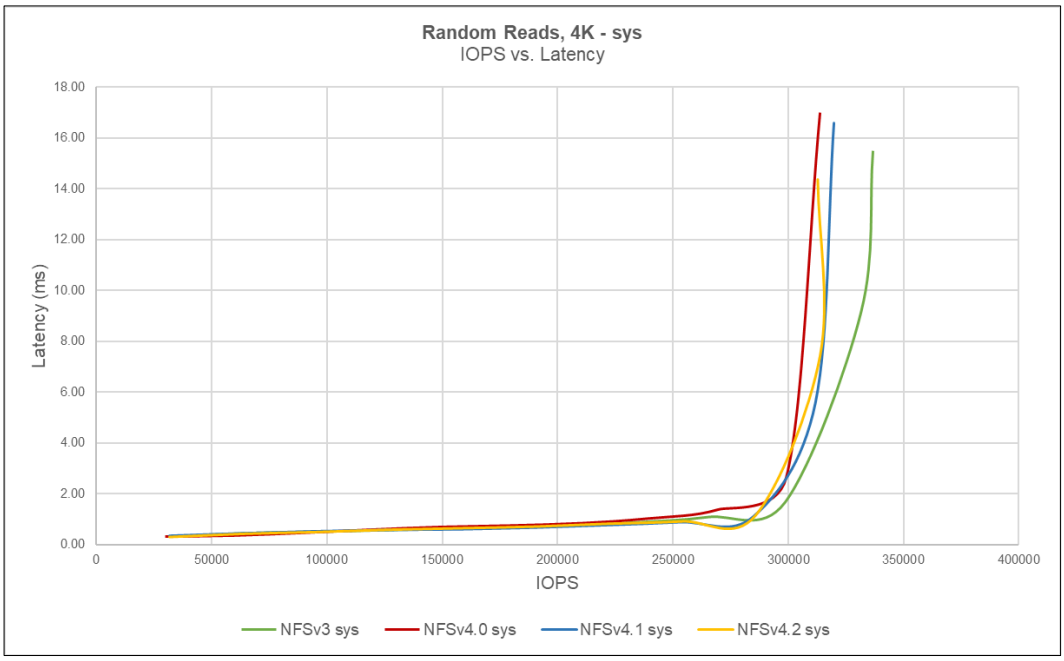


Figure 24) Random writes, 4K, NFSv3 versus NFSv4.x ——IOPS/Latency

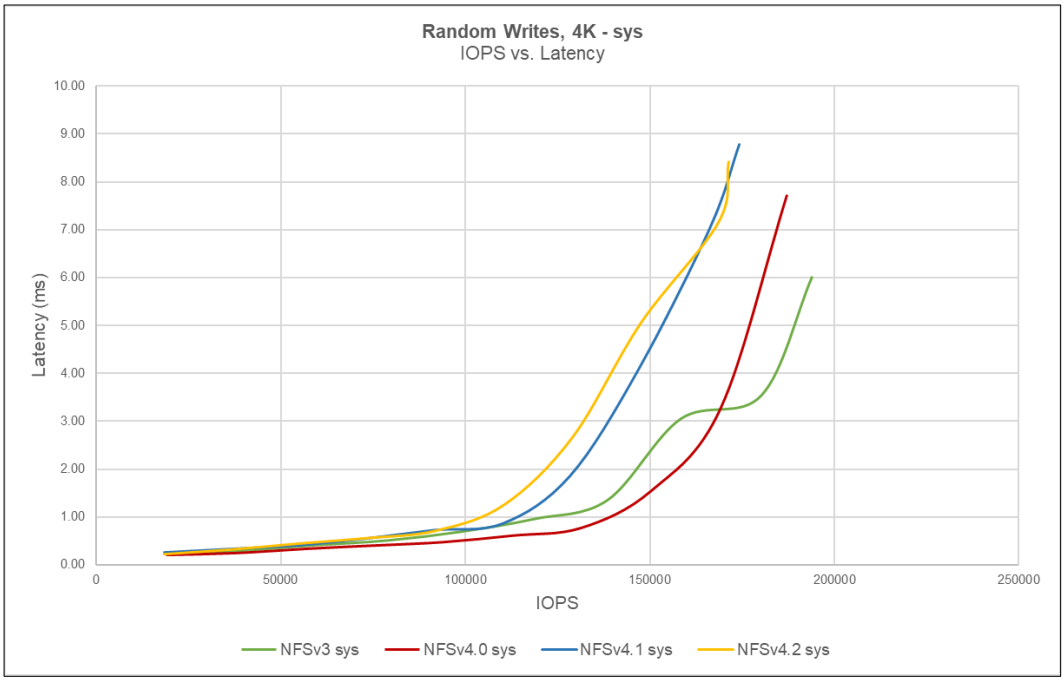


Figure 25) Sequential reads, 32K, NFSv3 versus NFSv4.x — IOPS/Latency

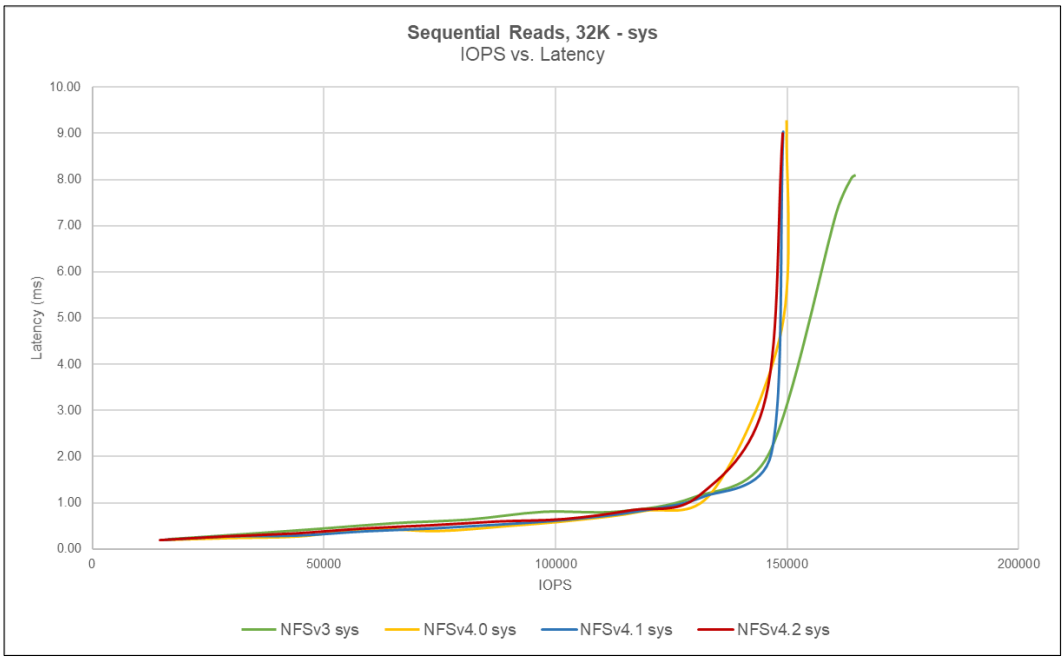
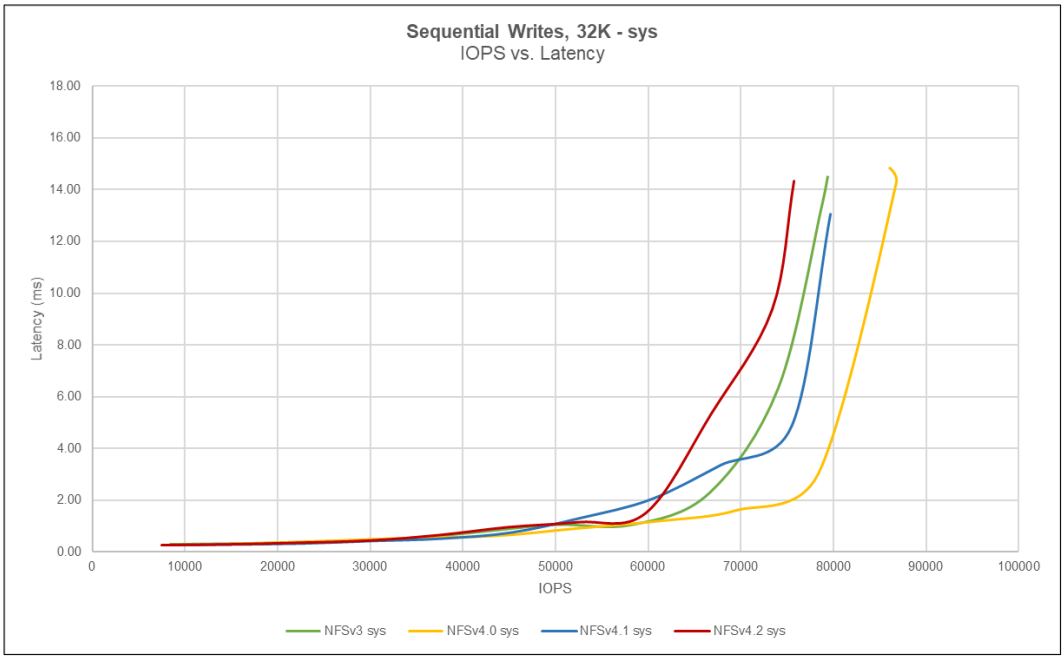


Figure 26) Sequential writes, 32K, NFSv3 versus NFSv4.x — IOPS/Latency



Performance examples for different TCP maximum transfer window sizes

The following examples include the following performance tests using multithreaded file and directory creations:

- High file count (one million files – 1,000 directories and 1,000x 4KB files per directory)
- Low file count, larger files (16 files – 16 directories, 1x 2GB files per directory; parallel dd writing to `/dev/urandom` with 4KB block size)

To find the scripts, see: https://github.com/whyistheinternetbroken/NetAppFlexGroup/.

The goal for these tests was not to show the maximum possible performance; to do that, we would have used larger AFF systems, more nodes in the cluster, more clients, and so on.

Instead, there are three goals for these tests:

- Show the impact of different wsize and rsize values on different workload types.
- Show NFSv3, NFSv4.1, and NFSv4.2 performance comparisons.
- Show the effect that NFS Kerberos has for these workloads.

These tests were run five times each and the values were averaged out for a more accurate representation of the results.

Here are some key points about the test environment:

- NFSv3, NFSv4.1 and NFSv4.2 (with and without pNFS)
- Clients:
  - Two RHEL 8.3 VMs
  - nconnect=8
  - TCP slot tables left as defaults (65536)
  - Virtual memory tuned to 40% `vm.dirty_ratio` and 20% `vm.dirty_background_ratio`
- AFF A300 HA pair — two nodes (ONTAP 9.9.1)
- FlexGroup volume (16 member volumes, 8 per node)
- 10GB network

Note: This test was not intended to show the maximum performance of NFS, or the systems being used, but to show comparisons between NFS versions and block sizes by using tests on the same network and hardware.

The NFS operations for the 4KB file test breakdown looked like this (even create/lookup/write):

```
Object: nfsv3
Instance: DEMO
    Counter                                                   Value
    ------------------------------ --------------------------------
    access_total                                               1017
    create_percent                                              33%
    create_total                                            1000000
    fsinfo_total                                                  4
    getattr_total                                                 3
    lookup_percent                                              33%
    lookup_total                                            1000003
    mkdir_total                                                1003
    null_total                                                    4
    pathconf_total                                                2
    total_ops                                                 36169
    write_percent                                               33%
    write_total                                             1000000
```

The NFS operations for the 2GB file test breakdown looked like this (100% write):

```
Object: nfsv3
Instance: DEMO
Number of Constituents: 2 (complete_aggregation)
    Counter                                                   Value
    ------------------------------ --------------------------------
    access_total                                                 32
    create_total                                                 16
    fsinfo_total                                                  4
    getattr_total                                                18
    lookup_total                                                 19
```

```
    mkdir_total                                    19
    null_total                                      4
    pathconf_total                                  2
    total_ops                                    3045
    write_percent                                 99%
    write_total                                511520
```

Test #1: High file count test — Many folders and files

This test created approximately one million 4KB files across 1,000 folders and used multiple different TCP transfer size values with the mounts. The file creation method was an f.write call from Python that wrote 16 characters over and over again to a file until it reaches 4KB in size. This generated more NFS operations. This test used multiple different transfer sizes, NFS versions (NFSv3, NFSv4.1, NFSv4.2) and NFSv4.1 with and without pNFS, using nconnect=8.

Table 35 breaks down the average completion times, average IOPs, and average throughput for each protocol and wsize/rsize value, as well as average CPU busy%.

The commands `qos statistics performance show` (for latency, IOPS and throughput) and `statistics show-periodic` (for CPU busy %) were used to collect this information.

Table 35) High file count test results — one million files.

Test	Completion time (seconds)	Average IOPS	Average MBps
NFSv3 – 64K wsize/rsize	~68.6	~55836	~72.6
NFSv3 – 256K wsize/rsize	~71	~55574	~72.2
NFSv3 – 1MB wsize/rsize	~73.1	~55865	~72.7
NFSv4.1 – 64K wsize/rsize	~251.1	~11182	~15.4
NFSv4.1 – 256K wsize/rsize	~259.5	~12041	~15.7
NFSv4.1 – 1MB wsize/rsize	~257.9	~11956	~15.6
NFSv4.1 – 64K wsize/rsize (pNFS)	~254	~11818	~15.4
NFSv4.1 – 256K wsize/rsize (pNFS)	~253.9	~11688	~15.2
NFSv4.1 – 1MB wsize/rsize (pNFS)	~253.1	~11850	~15.4
NFSv4.2 – 64K wsize/rsize (pNFS)	~256.3	~11756	~15.5
NFSv4.2 – 256K wsize/rsize (pNFS)	~255.5	~11890	~15.4
NFSv4.2 – 1MB wsize/rsize (pNFS)	~256.1	~11764	~15.2

Table 36) High file count test results — one million files — average CPU busy % and average latency.

Test	Average total latency (ms)	Average CPU busy %
NFSv3 – 64K wsize/rsize	~15.4	~64%
NFSv3 – 256K wsize/rsize	~15.1	~63%
NFSv3 – 1MB wsize/rsize	~15.3	~63%
NFSv4.1 – 64K wsize/rsize	~30.3	~27%
NFSv4.1 – 256K wsize/rsize	~29.8	~27%
NFSv4.1 – 1MB wsize/rsize	~30	~27%
NFSv4.1 (pNFS) – 64K wsize/rsize	~30.5	~26%
NFSv4.1 (pNFS) – 256K wsize/rsize	~30.9	~26%
NFSv4.1 (pNFS) – 1MB wsize/rsize	~30.4	~26%
NFSv4.2 (pNFS) – 64K wsize/rsize	~30.3	~26%

Test	Average total latency (ms)	Average CPU busy %
NFSv4.2 (pNFS) – 256K wsize/rsize	~30.1	~26%
NFSv4.2 (pNFS) – 1MB wsize/rsize	~30.5	~27%

Observations

- Regardless of the mount's wsize/rsize options, roughly the same number of IOPS were generated because of the nature of the workload (many small files). Regardless of whether we sent blocks in 64K or 1MB chunks, we still only sent 4K files, so the client sent those one by one. The average maximum throughput did not change much for NFSv3 but had some increases with larger wsize/rsize values.
- Overall, NFSv3 performed substantially better than NFSv4.x for this type of workload (up to 3.5x better) in ONTAP 9.9.1, generally because of better throughput and higher IOPS.
- NFSv4.x had lower overall CPU busy % because it generated fewer overall IOPS. Less operations to process = less CPU to use.
- pNFS didn't help performance much here.
- NFSv4.2 did not offer benefits for performance over NFSv4.1.

Test #2: Low file count test — Fewer, larger files

This test created 16x 2GB files (16 folders, one file per folder) by using a multithreaded dd operation, where a process per folder was created on each client. This test used multiple different transfer sizes, NFS versions (NFSv3, NFSv4.1, NFSv4.2) and NFSv4.x with and without pNFS. The file creation was done by using dd in parallel with 4KB block sizes using `/dev/urandom` to ensure storage efficiencies didn't affect file sizes too much. This generated more of a pure throughput test.

Table 37) Low file count test results — 2GB files.

Test	Completion time (seconds)	Average IOPS	Average MBps
NFSv3 – 64K wsize/rsize	~148.4	~3566	~222.8
NFSv3 – 256K wsize/rsize	~147.7	~901	~225.2
NFSv3 – 1MB wsize/rsize	~148.7	~226	~225
NFSv4.1 – 64K wsize/rsize	~148.6	~3578	~223.6
NFSv4.1 – 256K wsize/rsize	~147.6	~900	~224.8
NFSv4.1 – 1MB wsize/rsize	~148.1	~226	~225
NFSv4.1 (pNFS) – 64K wsize/rsize	~148.1	~3572	~222.4
NFSv4.1 (pNFS) – 256K wsize/rsize	~147.3	~902	~224.5
NFSv4.1 (pNFS) – 1MB wsize/rsize	~147.5	~228	~223.4
NFSv4.2 (pNFS) – 64K wsize/rsize	~147.9	~3593	~224.3
NFSv4.2 (pNFS) – 256K wsize/rsize	~148.1	~903	~224.6
NFSv4.2 (pNFS) – 1MB wsize/rsize	~147.7	~229	~224.3

Table 38) Low file count test results — Average CPU busy % and average latency.

Test	Average total latency (ms)	Average CPU busy %
NFSv3 – 64K wsize/rsize	~0.44	~28%
NFSv3 – 256K wsize/rsize	~0.87	~25%
NFSv3 – 1MB wsize/rsize	~3.98	~25%
NFSv4.1 – 64K wsize/rsize	~0.44	~30%

Test	Average total latency (ms)	Average CPU busy %
NFSv4.1 – 256K wsize/rsize	~0.89	~27%
NFSv4.1 – 1MB wsize/rsize	~4.15	~26%
NFSv4.1 (pNFS) – 64K wsize/rsize	~0.31	~29%
NFSv4.1 (pNFS) – 256K wsize/rsize	~0.65	~25%
NFSv4.1 (pNFS) – 1MB wsize/rsize	~2.51	~26%
NFSv4.2 (pNFS) – 64K wsize/rsize	~0.35	~28%
NFSv4.2 (pNFS) – 256K wsize/rsize	~0.65	~25%
NFSv4.2 (pNFS) – 1MB wsize/rsize	~2.1	~25%

Observations

- Generally speaking, NFSv3 and NFSv4.x performed nearly identically for this type of workload. The completion times for the tests were within 1–2 seconds of each other and the throughput numbers were all similar across the board. The main difference was in average latency.
- NFSv4.x had lower latency than NFSv3 – especially when using pNFS to ensure data locality. Having data requests use the cluster network added between 0.1 and 1.5 ms write latency to the workloads (depending on mount wsize) and 100–110MB of additional traffic across the cluster network.
- With NFSv4.2, there was no real noticeable performance increase from NFSv4.1 except when using 1MB wsize (~.4ms improvement).
- For sequential workloads that were mostly reads/writes, NFSv4.1 and later with pNFS performed just as well as NFSv3 for the same workloads.

NFS Kerberos performance testing

This section on NFS Kerberos performance testing has been removed. A future version of this document will be published with corrected numbers.

Default NFS ports in ONTAP

Table 39) Default NFS Ports in ONTAP.

NFS service	ONTAP port	Applicable NFS version	Option to change the port
mountd	635	`NFSv3`	`-mountd-port`
portmapper	111	`NFSv3`	N/A – Cannot be changed
NLM	4045	`NFSv3`	`-nlm-port`
NSM	4046	`NFSv3`	`-nsm-port`
NFS	2049	`NFSv3 and NFSv4.x`	N/A: cannot be changed
rquota	4049	`NFSv3`	`-rquotad-port`

A special thanks to Kelly Alexander, CPOC labs, NetApp (Kerberos performance information).

Where to find additional information

Request for comments (RFC)

- RFC 2203: RPCSEC_GSS Protocol Specification
- RFC 3530: Network File System (NFS) Version 4 Protocol
- RFC 5661: Network File System (NFS) Version 4 Minor Version 1 Protocol

- RFC 5331: RPC: Remote Procedure Call Protocol Specification Version 2

Technical reports

- TR-3580: NFSv4 Enhancements and Best Practices Guide: Data ONTAP Implementation
- TR-3633: Oracle Databases on NetApp ONTAP
- TR-4523: DNS Load Balancing in ONTAP
- TR-4543: SMB Best Practices, ONTAP 9.x
- TR-4571: NetApp FlexGroup Volume Best Practices and Implementation Guide
- TR-4668: Name Services Best Practices
- TR-4616: NFS Kerberos in ONTAP with Microsoft Active Directory
- TR-4617: Electronic Design Automation Best Practices
- TR-4743: FlexCache in ONTAP
- TR-4835: How to Configure LDAP in ONTAP
- TR-4887: Multiprotocol NAS in NetApp ONTAP — Overview and Best Practices

Version history

Version	Date	Document version history
Version 1.0	June 2013	Initial release
Version 2.0	October 2013	Updated for ONTAP 8.2
Version 2.1	January 2014	Updated for ONTAP 8.2.1
Version 2.2	September 2014	Updated for ONTAP 8.2.2
Version 3.0	February 2015	Updated for ONTAP 8.3
Version 3.1	July 2015	Updated for ONTAP 8.3.1
Version 3.2	February 2016	Updated for ONTAP 8.3.2
Version 4.0	July 2016	Updated for ONTAP 9.0
Version 4.1	October 2016	Updated for ONTAP 9.1
Version 4.2	July 2017	Updated for ONTAP 9.2
Version 5.0	June 2020	Major revision; Updated for ONTAP 9.7
Version 5.1	February 2021	Updated for ONTAP 9.8
Version 5.2	June 2021	Updated for ONTAP 9.9.1
Version 5.3	November 2021	Updated for ONTAP 9.10.1
Version 5.3.1	June 2023	Removed Kerberos performance numbers

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.

TR-4067-00523

NetApp