

Dell PowerScale: OneFS NFS Design Considerations and Best Practices

Abstract

This document shows how to implement the Network File System (NFS) service on Dell PowerScale OneFS and provides key considerations and best practices. This paper applies to OneFS 9.0.x and later.

May 2023

Revisions

Date	Part number/ revision	Description
May 2018		Initial release
June 2020		PowerScale rebranding
May 2021		NFSv3 over RDMA new feature in OneFS 9.2.0
October 2021		NFSv4.1 support in OneFS 9.3.0
January 2023	H17240.4	Updated for SmartQoS in OneFS 9.5.0
May 2023	H17240.5	Minor updates

Acknowledgments

Author: Lieven Lin

The information in this publication is provided "as is." Dell Inc. makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any software described in this publication requires an applicable software license.

Copyright ©2018--2023 Dell Inc. or its subsidiaries. All Rights Reserved. May 2023. H17240.5.

Table of contents

Revisions	2
Acknowledgments	2
Table of contents	3
Executive summary	5
Audience	5
1 NFS protocol and OneFS	6
1.1 OneFS overview	6
1.2 NFS protocol introduction	6
1.2.1 NFSv3	7
1.2.2 NFSv4.x	7
1.2.3 Advantages of NFSv4.x	8
1.3 NFS compatibility with OneFS	8
1.3.1 NFSv4.1 support	9
2 Implementing NFS	10
2.1 Identity management and authentication	10
2.2 Create NFS export	11
2.3 Mount export over NFSv3/NFSv4.0/NFSv4.1/NFSv4.2	12
3 PowerScale OneFS considerations	13
3.1 NFS export considerations	13
3.2 SmartConnect	13
3.3 Access zone	15
3.4 AIMA (Authentication, Identity Management, Access)	16
3.5 OneFS protocol audit	17
4 NFS client considerations	19
4.1 Linux client	19
4.2 macOS client	22
5 NFS security considerations	23
5.1 Network security considerations	23
5.2 Authentication	24
5.3 NFSv4.x ACL	25
5.4 NFSv4.x pseudo-file system	29
6 NFSv3 over RDMA	31
6.1 NFSv3 over RDMA overview	31
6.2 Management options	32

6.2.1	Enable/disable NFS over RDMA globally	32
6.2.2	Filter RoCEv2 capable network interfaces for an IP pool.....	33
6.2.3	Check ROCEv2 capability of network interfaces	34
6.3	Key considerations	35
7	SmartQoS	36
7.1	SmartQoS overview	36
7.2	Configuration ops limitation	36
7.2.1	Enable SmartQoS in OneFS	36
7.2.2	Create SmartQoS dataset	36
7.2.3	Pin a workload from the dataset	38
7.2.4	Set an ops limit for a pinned workload	39
8	Useful NFS commands and tools	40
8.1	isi statistics command.....	40
8.2	Packet capture tool and analysis.....	42
A	Technical support and resources	44
A.1	Related resources	44

Executive summary

This document provides common configurations and considerations to help you implement, configure, and manage NFS storage service on Dell PowerScale products. This content includes the following:

- NFS protocol introduction and its compatibility with OneFS
- Quick start implementation guide to use NFS service on OneFS
- NFS considerations on OneFS
- NFS considerations on client
- NFS security considerations

Audience

This document is intended for administrators who are using NFS storage service on PowerScale OneFS. The document assumes you have knowledge of the following:

- Network Attached Storage (NAS) systems
- Network File System (NFS) protocol
- PowerScale OneFS distributed file system and scale-out architecture
- Directory service such as Active Directory and LDAP

You should also be familiar with Dell PowerScale documentation resources, including:

- [Dell PowerScale OneFS: A Technical Overview](#)
- [PowerScale OneFS Web Administration Guide](#)
- [PowerScale OneFS CLI Administration Guide](#)
- [Current PowerScale Software Releases](#)
- [OneFS Security Configuration Guide](#)

1 NFS protocol and OneFS

1.1 OneFS overview

OneFS is a fundamental component for Dell PowerScale storage. It is used to power all Dell PowerScale NAS storage solutions and offers the following key advantages:

- **Scale-out architecture:** OneFS is designed to scale in terms of machine, by adding more PowerScale nodes to a cluster, both performance and capacity is scaled. It enables PowerScale scale to a multi-petabyte large cluster which contains a maximum of 252 nodes.
- **Single file system and namespace:** Traditional storage model contains file system, volume manager, and data protection. OneFS combines all of them into a single intelligent distributed file system, and provides a single namespace that is accessible through multi-protocol (NFS, SMB, HDFS, S3, HTTP, and FTP) simultaneously.
- **Efficiency and ease of management:** OneFS provides unique features to improve PowerScale NAS storage system efficiency and ease of management. For example, with OneFS SmartPools, you can tier your cold data to lower-cost PowerScale nodes automatically.

OneFS is not only the operating system but also the underlying file system that drives and stores data in the PowerScale scale-out NAS cluster. For more details, see the document [Dell PowerScale OneFS: A Technical Overview](#).

1.2 NFS protocol introduction

The Network File System (NFS) protocol allows users to mount remote file systems transparently and access to shared files across networks. It uses a client/server model based on Remote Procedure Call Protocol ([RFC5531](#)), so NFS is portable across different machines, operating systems, network architecture, and transport protocols. NFS eliminates the need to keep copies of files on several machines by letting the clients all share a single copy of a file on the server.

There are three major NFS versions. Each of them is defined in an RFC specification as shown in the following table. This chapter provides a brief summary about NFSv3 and NFSv4.x as they are implemented in most enterprise environments. For more details, see the links in the table.

Table 1 NFS versions and evolution

Version	RFC	Status
NFSv2	RFC1094 (published in 1989)	Obsolete
NFSv3	RFC1813 (published in 1995)	Most popular
NFSv4.0	RFC3010 (published in 2000, obsolete)	Slowly replacing NFSv3
	RFC3530 (published in 2003, obsolete)	
	RFC7530 (published in 2015)	
NFSv4.1	RFC5661 (published in 2010)	Adopted gradually
NFSv4.2	RFC7862 (published in 2016)	

1.2.1 NFSv3

The NFSv3 is a stateless protocol. Statelessness means that the server does not need to maintain state about any of its clients in order to function correctly. The NFSv3 has following key enhancements compared with NFSv2:

- The file handle has been increased to a variable length up to 64 bytes, instead of 32 bytes.
- Supports files larger than 2 GB. Maximum file size depends on the NFS server's local file systems.
- Eliminates the 8 KB limit of the maximum size of an on-the-wire NFS read or write operation.
- Introduces the concept of Weak Cache Consistency. A server reply to a read or write operation returns extra attribute information which can be used by clients to decide whether its data and attribute caches are stale. So NFSv3 clients will detect changes to files faster that are modified by another client.
- Introduces safe asynchronous writes to improve performance. New COMMIT procedure is added to flush data to stable storage for an asynchronous write and to detect if retransmit the data is needed.

To function correctly, NFSv3 also relies on several auxiliary protocols.

Mount protocol

The [mount](#) protocol is separate from, but related to, the NFS protocol. It provides operating system-specific services to get NFS off the ground - looking up export pathnames, validating user identity, and checking access permissions. Clients use the mount protocol to get the first file handle, which allows them entry into a remote file system.

Network Lock Manager (NLM) protocol and Network Status Monitor (NSM) protocol

Because NFS is a stateless service, auxiliary protocols are needed to offer inherently stateful services such as file locking and access control synchronization. So the RPC-based [NLM](#) protocol and [NSM](#) protocol work together to provide file locking and access control capability.

Binding protocol ([RFC1833](#))

As NFS protocol and its auxiliary protocols mentioned above are all based on RPC, it is necessary to map RPC program number/version number pairs to the network port number (TCP/UDP port number) for that version of that program. Binding protocol provides such a lookup service to find a network port number for a specific RPC program number/version. There are three versions of a lookup service: RPCBIND (Versions 3 and 4) which uses a transport-independent format for the transport address, and Port Mapper (Version 2) which is an older protocol only specific for TCP and UDP transport.

1.2.2 NFSv4.x

The biggest change in NFSv4.x is that it is designed as a stateful protocol. Unlike earlier NFS versions which needs auxiliary protocols to provide additional functions, the NFSv4.x integrates the file locking (NLM/NSM) and the mount protocol. Besides, the NFSv4.x also provides some of the key features as follows:

- Introduces the COMPOUND procedure as a wrapper to coalesce one or more operations into a single RPC request.
- Introduces the NFSv4.x Access Control Lists (ACL) to support more expressive and granular access control while clients accessing the NFS shared files.

- The server can grant a read or write file delegation to the clients, which enables the clients to aggressively cache file data.
- Session model is introduced since NFSv4.1 for better connection management.

1.2.3 Advantages of NFSv4.x

The NFSv4.x retains the essential characteristics of previous version, such as independent of operating systems, simplicity, and good performance. It also has more advantages compared with older versions.

COMPOUND procedure

The compound procedure will combine multiple individual operations into a single request to reduce the number of RPC packets transmitted over the network. The client can avoid the cumulative latency of multiple RPCs. So the NFSv4.x will perform better in a potentially high latency network like Internet.

Firewall friendly

To access an NFSv3 server, the client needs to involve NFS protocol and its auxiliary protocols (port mapper, mount, NLM/NSM), each of them needs TCP/UDP ports which would not be the well-known ports listening on the network. This will cause problems for using NFS through firewall. In the NFSv4.x, there are no auxiliary protocols and it is designed as a single protocol using a single TCP port, usually listening on port 2049. So it traverses firewalls and network address translation (NAT) devices easily, and makes the network management and configuration easily. More details and considerations are discussed in 5.1 Network security considerations.

Stronger security

The NFSv4.x ACL file attribute is added to enable more expressive access control. The NFSv4.x ACL model is quite rich. With the use of NFSv4.x ACL, the server does all access control based on the server's interpretation of the ACL although the client can manipulate the ACL attributes. More details and considerations are discussed in 5.3 NFSv4.x ACL.

1.3 NFS compatibility with OneFS

This document focuses on OneFS 8.0.0 and above. However, to provide an overview of the NFS compatibility with OneFS families, a compatibility table is shown in Table 2.

Table 2 NFS compatibility with OneFS

NFS version	OneFS compatibility				
	**8.x	9.0.x	9.1.x	9.2.x	9.3.x
NFSv3	√	√	√	√	√
NFSv4.0	√	√	√	√	√
NFSv4.1					√

Note: NFSv4.2 just adds additional new features based on NFSv4.1, therefore, OneFS allows clients to mount NFS export with NFSv4.2, but OneFS does not implement NFSv4.2 features. ** indicates that the version of OneFS is EOSL. For information about the support and service life-cycle dates for PowerScale hardware and software products, see the [PowerScale Product Availability Guide](#).

1.3.1 NFSv4.1 support

NFSv4.1 introduced several 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 or 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. Starting from OneFS 9.3.0, OneFS supports NFSv4.1 and NFSv4.2 to access data by implementing all the required features defined in RFC-5661. This support excludes the Secret State Verifier (SSV) which is not implemented by any open-source Linux clients. OneFS 9.3.0 implements the [session model](#) which allows NFSv4.1 and NFSv4.2 clients to leverage [trunking](#). Trunking is the use of multiple connections between a client and server in order to widen the I/O path. OneFS 9.3.0 supports both session trunking and client ID trunking.

- Session trunking is the association of multiple TCP connections to the same session. Most Linux supports session trunking by using **nconnect** option which is included in Linux kernel version higher than 5.3.
- Client ID trunking is the association of multiple sessions to the same client ID. Not observed any open-source Linux clients support client ID trunking.

Figure 1 shows the supported NFS operations in OneFS 9.3.0. Both NFSv4.1 and NFSv4.2 use the existing NFSv4.0 I/O stack in OneFS. NFSv4.2 is a superset of NFSv4.1, with all new features being optional. OneFS still allows clients to mount NFS export of OneFS with NFSv4.2 and access OneFS data even OneFS does not implement any NFSv4.2 features.

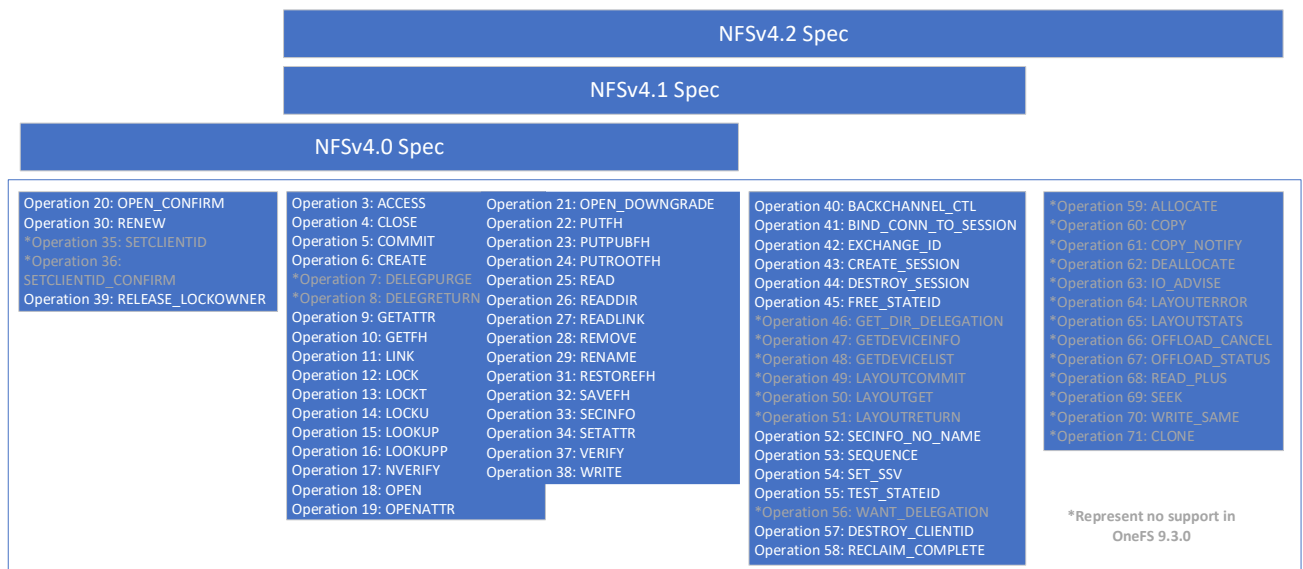


Figure 1 Supported NFS operations in OneFS 9.3.0

2 Implementing NFS

The Dell PowerScale OneFS operating system can enable seamless multiprotocol data access with unified security model. NFS is one of the protocols that gives UNIX and Linux system access to OneFS. This chapter provides a quick start guide to implement NFS as your data access protocol to OneFS in your environment: identity management and authentication, create NFS export and mount export to clients.

2.1 Identity management and authentication

It is important to understand the identity management and authentication methods before implementing NFS to your environment. Identity management provides a location to store user information, tell where a user is. Authentication is a process that validates the identity of a user. In OneFS, the identity management and authentication is offered by authentication providers. OneFS supports the following methods for authenticating user:

- Active Directory (AD)
- Lightweight Directory Access Protocol (LDAP)
- Network Information Service (NIS)
- Local users and groups

Use Active Directory service and LDAP for ease of user identity and access management.

Active Directory (AD)

Active Directory is implemented by Microsoft that provides several services: LDAP, Kerberos, and DNS. The primary reason for a PowerScale cluster to join an AD domain is to provide user/group identity management and authentication. Active Directory service is used to authenticate all Windows clients and users. OneFS is compliant with RFC2307, therefore in a multiprotocol environment it is recommended to integrate AD with OneFS to provide a centralized identity management and authentication.

RFC2307 allows you to implement unified authentication for UNIX and Windows Active Directory accounts by associating a user ID (UID), group ID (GID), home directory, and shell with an Active Directory object.

Windows Server supported some variations of these schema extensions in versions before Windows 2003 R2 with Microsoft Services for UNIX. Windows 2003 R2 and later versions provide full RFC 2307 support. This means that, when configured, the standard UNIX attributes exist in the schemas of domains created with Windows 2003 R2 and later.

To use Active Directory as authentication provider for NFS service. You need to configure the OneFS and Active Directory for RFC2307 compliance, and integration with AD for NFS is also needed on the NFS client side. For more details about how to enable RFC2307 for OneFS and Active Directory, refer to the blog [article](#). For more details about how to integrate Linux client with AD for NFS, refer to the associated official documentations, for example, refer to [Red Hat Windows Integration Guide](#) for Red Hat Linux distribution.

Lightweight Directory Access Protocol (LDAP)

OneFS cluster can also be configured to use LDAP as the authentication provider. The LDAP service in a OneFS cluster supports the following features:

- Users, groups, and netgroups
- Customized LDAP attribute mapping
- Simple BIND authentication

- Redundancy and load balancing across all server with identical directory data
- Encrypted passwords

For more details on configure OneFS cluster integrate with LDAP, refer to [OneFS Web Administration Guide](#).

To enable a Linux client using LDAP, you can refer to the corresponding Linux distribution official documentation, for example, refer to [Red Hat System-level Authentication Guide](#) for Red Hat Linux distribution.

Network Information Service (NIS)

The NIS is a directory services protocol designed by Sun Microsystems. It has inherent limitations, especially in the areas of scalability and security. So it is usually replaced by LDAP unless your environment has been using NIS for a long time.

Local users and groups

The OneFS cluster supports local users and groups for authentication. You can create local users and groups accounts directly on the cluster. Local authentication can be useful for a test environment or if there is no directory service available.

In a multi-protocol environment, there are usually multi-authentication providers with Active Directory for Windows client's access and LDAP for Linux or UNIX client's access. If a user exists in both Active Directory and LDAP, it is required to configure a user-mapping rule for the user to have enough permission to access files. You can use `isi auth mapping create/view` to create or view the user mapping rules.

2.2 Create NFS export

OneFS supports NFSv3, NFSv4.0, NFSv4.1 and NFSv4.2. OneFS does not implement NFSv4.2 new features, but it allows clients to mount NFS export with NFSv4.2. By default, OneFS has NFS service disabled. You need to enable NFS service first from WebUI or using `isi` command. Enabling NFSv4.x is non-disruptive on a OneFS cluster, and it will run concurrently with NFSv3. Any existing NFSv3 clients will not be affected by enabling NFSv4.x.

OneFS NFS export is zone-aware, every NFS export is associated with an Access Zone. By default, an access zone named "System" is used if you do not specify an Access Zone name when the export is created. More details about access zones are provided in 3.3 Access zone.

The NFS export can be created using both WebUI and `isi` command. For details about creating NFS exports, refer to [OneFS Web Administration Guide](#) and [OneFS CLI Administration Guide](#).

By default, the NFS service applies a root-squashing rule (map the root user to nobody user) for the NFS export. This prevents the client from gaining root privileges to the server despite the user's credential. Keeping the rule as the default setting is recommended because the root account is the super user in Linux and UNIX environments.

Note: If you are creating an NFSv4.x export, you need to configure a same NFSv4.x domain on both the OneFS cluster and NFSv4.x clients. You can configure the NFSv4.x domain for the OneFS from WebUI or using `isi nfs settings zone modify`. To configure the NFSv4.x domain for NFSv4.x client, you can edit the `Domain = example.local` to your NFSv4.x domain in the `/etc/idmapd.conf` file on the client.

2.3 Mount export over NFSv3/NFSv4.0/NFSv4.1/NFSv4.2

NFS v3/v4 are supported on Linux 2.6 kernels and later. In this white paper, we use Centos 6.9 as NFS client to illustrate the client side configuration.

NFS exports are mounted on the client using the `mount` command. The format of the command shown as below:

```
# mount -t nfs -o options server:/remote/export /local/directory
```

When you mount an NFS export, the NFS protocol version is determined at mount time, and can be modified by specifying the version of the NFS protocol using mount options `nfsvers` or `vers`. For example, the command `mount -t nfs -o nfsvers=4.1 server:/remote/export /local/directory` mounts the export with NFSv4.1.

The drawback of using `mount` command is that the mounted NFS export is not persistent across client reboots. So you can use `/etc/fstab` and `autofs` to mount the NFS file system. For more details about `mount`, `/etc/fstab` and `autofs`, refer to [Red Hat Storage Administration Guide](#). More mount options and considerations are discussed in [3.3 NFS Client Considerations](#).

Note: mount options `nfsvers` and `vers` have the same meaning in the mount command. The option `vers` is compatible with NFS implementations on Solaris and other vendors.

3 PowerScale OneFS considerations

3.1 NFS export considerations

NFS export read size and write size on OneFS

While mounting export to a client, you can specify the read size (`rsize`) and write size (`wsiz`) options. Larger `rsize` and `wsiz` improve the throughput performance. By default in OneFS, the `rsize` of 128 KB and `wsiz` of 512 KB are advertised for NFSv3 Linux clients, but can be set as high as 1 MB. NFSv4.x defaults to 1 MB for both `rsize` and `wsiz`. Explicitly setting these values too small will override the default value and might result in slower than optimal performance.

NFS aliases

In general, create an export with short path when possible. If the long directory path must be used, you can create an NFS alias for the directory path. An NFS alias is designed to give functional parity with SMB share name within the context of NFS. It provides a shortcut for the path. Like exports, NFS aliases are also access zone-aware. You can specify the access zone when you create an alias. You can check the status of an alias from the WebUI or using `isi nfs aliases` with `--check` option (status can be: good, illegal path, name conflict, not exported, or path not found).

NFSv3 encoding support

OneFS 8.1.1.1 includes enhancements to NFSv3 encoding. This sets the cluster encoding and export encoding to the same character set using RFC-compliant filenames, enabling exports to any NFSv3 client that uses western or nonwestern characters.

Before OneFS 8.1.1.1, non-utf-8 encoded files or directories may be inaccessible to NFSv3 and NFSv4.x clients. If this issue occurs in your environment, contact Dell technical support service to help solve the issue or upgrade your OneFS cluster to OneFS 8.1.1.1 or higher.

32-bit file IDs over NFS

The NFS protocol supports 64-bit file IDs from NFSv3 onwards. However, some applications do not support 64-bit file IDs. To accommodate these applications, OneFS allows enabling 32-bit file IDs from WebUI or using `isi nfs export <exportID> --return-32bit-file-ids`.

Note: When this change is made, all clients must be remounted. Clients that are not remounted will encounter errors such as "error: fileid changed," or the clients will continue to receive 64-bit file IDs. You might need to schedule a maintenance window before making this change.

3.2 SmartConnect

SmartConnect, a licensable software module of the PowerScale OneFS operating system, helps to greatly simplify client management across the enterprise. Through a single host name, SmartConnect enables client connection load balancing and dynamic NFS failover and failback of client connections across storage nodes to provide optimal utilization of the cluster resources.

Client Connection Load Balancing

SmartConnect balances client connections across nodes based on policies that ensure optimal use of cluster resources. By leveraging your existing network infrastructure, SmartConnect provides a layer of intelligence that allows all client and user resources to point to a single hostname, enabling easy management of a large and growing numbers of clients. Based on user configurable policies, SmartConnect applies intelligent algorithms (CPU utilization, aggregate throughput, connection count, or round robin) and distributes clients across the cluster to optimize client performance and end-user experience. SmartConnect can be configured into multiple zones that can be used to ensure different levels of service for different groups of clients. All of this is transparent to the end user.

NFS failover using dynamic IP pool

SmartConnect uses a virtual IP failover scheme that is specifically designed for PowerScale scale-out NAS storage and does not require any client-side drivers. The PowerScale cluster shares a “pool” of virtual IPs that is distributed across all nodes of the cluster. The cluster distributes an IP address across NFS (Linux and UNIX) clients based on a client connection balancing policy.

This is an example illustrating how NFS failover works. As shown in Figure 2, in the six-node OneFS cluster, an IP address pool provides a single static node IP (10.132.0.140 – 10.132.0.145) to an interface in each cluster node. Another pool of dynamic IPs (NFS failover IPs) has been created and distributed across the cluster (10.132.0.150 – 10.132.0.161).

	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6
Static IPs	10.132.0.140	10.132.0.141	10.132.0.142	10.132.0.143	10.132.0.144	10.132.0.145
Dynamic IPs	10.132.0.150 10.132.0.156	10.132.0.151 10.132.0.157	10.132.0.152 10.132.0.158	10.132.0.153 10.132.0.159	10.132.0.154 10.132.0.160	10.132.0.155 10.132.0.161

Figure 2 Dynamic IPs and Static IPs

When Node 1 in the cluster goes offline, the NFS failover IPs and connected clients associated with Node 1 failover to the remaining nodes based on the configured IP failover policy (Round Robin, Connection Count, Network Throughput, or CPU Usage). The static IP for Node 1 is no longer available as shown in Figure 3.

	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6
Static IPs	10.132.0.140	10.132.0.141	10.132.0.142	10.132.0.143	10.132.0.144	10.132.0.145
Dynamic IPs	10.132.0.150 10.132.0.156	10.132.0.151 10.132.0.157	10.132.0.152 10.132.0.158 10.132.0.150	10.132.0.153 10.132.0.159 10.132.0.156	10.132.0.154 10.132.0.160	10.132.0.155 10.132.0.161

Figure 3 NFS Failover with Dynamic IP Pool

Therefore, using dynamic IP pool for NFS workload is recommended to provide NFS service resilience. If a node with client connections established goes offline, the behavior is protocol-specific. Because NFSv3 is a

stateless protocol, after the node failure, workflows can be moved easily from one cluster node to another. It will automatically reestablish the connection with the IP on the new interface and retries the last NFS operation. NFSv4.x is a stateful protocol, the connection state between the client and the node is maintained by OneFS to support NFSv4.x failover and in OneFS 8.x versions and higher, OneFS keeps that connection state information for NFSv4.x synchronized across multiple nodes. Once the node failure occurs, the client can resume the workflow with the same IP on the new node using the previous maintained connection state.

The number of IPs available to the dynamic pool directly affects how the cluster load balances the failed connections. For small clusters under N ($N \leq 10$) nodes, the formula $N*(N-1)$ will provide a reasonable number of IPs in a pool. For larger clusters, the number of IPs per pool is between the number of nodes and the number of clients. Requirements for larger clusters are highly dependent on the workflow and the number of clients.

3.3 Access zone

OneFS provides a single namespace while enabling multi-protocol access, such as NFS and SMB. Linux machines access the data using NFS; Windows computers access the data using SMB. There is a default shared directory (**ifs**) of OneFS, which lets clients running Windows, UNIX, Linux, or Mac OS X access the same directories and files. We recommended that you disable the **ifs** shared directory in a production environment and create dedicated NFS exports and SMB shares for your workload.

To securely support data access to OneFS, it does three main things:

- Connects to directory services, such as Active Directory, NIS, and LDAP, which are also known as identity management systems and authentication providers. A directory service provides a security database of user and group accounts along with their passwords and other account information.
- Authenticates users and groups. Authentication verifies users identity and triggers the creation of an access token that contains information about a user's identity.
- Controls access to directories and files. OneFS compares the information in an access token with the permissions associated with a directory or a file to allow or deny access to it.

All three of these functions take place in an access zone -- a virtual security context to control access based on an incoming IP address (groupnet) and provides a multi-tenant environment. In an access zone, OneFS connects to directory services, authenticates users, and controls access to resources. A cluster has a default single access zone, which is known as the **System** access zone. Until you add an access zone, NFS exports are in the default access zone.

The considerations for access zone are as below:

- Each access zone may include at most one MIT Kerberos provider.
- An access zone is limited to a single Active Directory provider; however, OneFS allows multiple LDAP, NIS, and file authentication providers in each access zone. Assign only one type of each provider per access zone to simplify administration.
- Creating a large number of local users and groups may affect system performance. Therefore, we recommend limiting the number of local users and groups per cluster to 25,000 each.
- Use the **System** access zone for cluster management, and create additional access zones for data access.
- Separate organization tenants using access zone with no more than 50 zones.
- Designate separate directory path for each access while you are creating multiple access zones.

- If DNS settings are different for your different NFS workflows, you can specify the dedicated DNS settings for each workflow using groupnet.

3.4 AIMA (Authentication, Identity Management, Access)

When a user connects to a PowerScale cluster, OneFS checks the directory services to which the user's access zone is connected for an account for the user. If OneFS finds an account that matches the user's login name, OneFS verifies the user's identity to authenticate the user. During authentication, OneFS creates an access token for the user. The token contains the user's full identity including group memberships and OneFS uses the token later to check access to directories and files.

When OneFS authenticates users with different directory services, OneFS maps a user's account from one directory service to the user's accounts in other directory services within an access zone—a process known as user mapping. A Windows user account managed in Active Directory, for example, is mapped by default to a corresponding UNIX account with the same name in NIS or LDAP.

As a result, with a single token, a user can access files that were stored by a Windows computer over SMB and files that were stored by a Linux computer over NFS.

Similarly, because OneFS provides multiprotocol access to files, it must translate the permissions of Linux and UNIX files to the access control lists of Windows files. As a result, a user who connects to the cluster with a Linux computer over NFS can access files that were stored on the cluster by a Windows user with SMB.

The following diagram Figure 4 summarizes how directory services, identity mapping, policies, and permissions play a role in the OneFS system of authentication and access control. For more details about AIMA, refer to [OneFS Multiprotocol Security Guide](#).

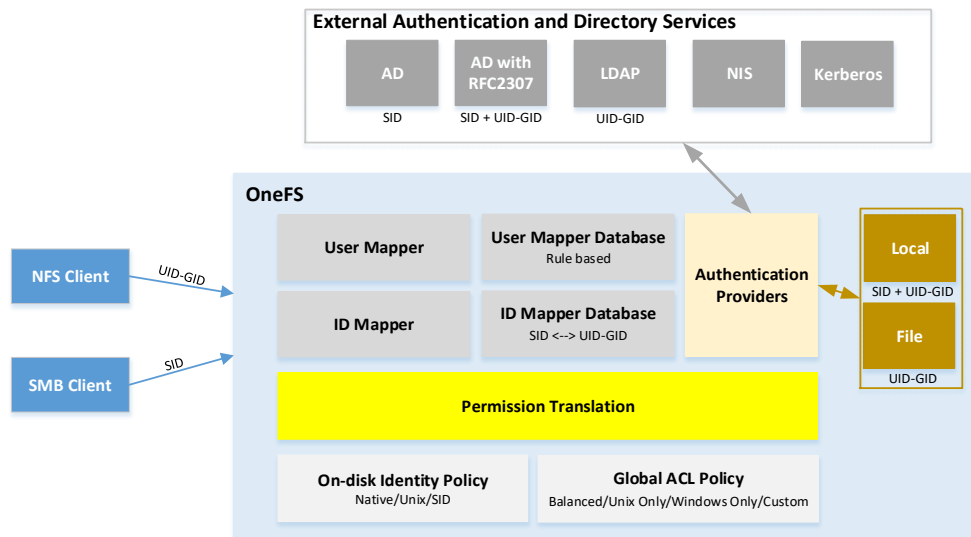


Figure 4 OneFS authentication and access control

No overlapping ID ranges and avoid common ID ranges

In the case that you contain multiple identity sources, like LDAP and Active Directory with RFC2307, you should ensure that the UID and GID ranges do not overlap. Besides, OneFS also allocates a UID and GID as needed. The default range from which OneFS automatically allocate a UID and GID is 1,000,000 to

2,000,000. Other identity source ID range must not overlap with the OneFS default range. If UIDs and GIDs overlap across two or more directory services, some users could have right to access to other users' files.

In addition, there are UIDs and GIDs below 1000 are reserved for system, do not assign them to users or groups in the identity source.

User mapping in multiple directory services

When the name of an account in different directory services match exactly, OneFS automatically combines their access tokens into a single token. AIMA requires that SAMAccount name is populated in AD for each user. For example, the user-mapping service maps, by default, a Windows account named CORP\jane from Active Directory to a UNIX account named jane from LDAP and generates an access token that combines the group membership information from the two accounts. OneFS also automatically maps two groups with the same name. Besides the automatic user mapping, OneFS allows the administrator to map users from different directory services manually from the WebUI and CLI. See the [OneFS Web Administration Guide](#) and [OneFS CLI Administration Guide](#).

Below are some considerations for user mapping:

- **Employ a consistent username strategy:** The simplest configurations name users consistently, so that each UNIX user corresponds to a similarly named Windows user. Such a convention allows rules with wildcard characters to match names and map them without explicitly specifying each pair of accounts.
- **Do not use UPNs in mapping rules:** You cannot use a user principal name (UPN) in a user-mapping rule. A UPN is an Active Directory domain and username that are combined into an Internet-style name with an @ symbol, such as an email address: jane@example. If you include a UPN in a rule, the mapping service ignores it and may return an error. Instead, specify names in the format DOMAIN\user (as the backslash is a special character, using additional backslash as the escape character on OneFS).
- **Group rules by type and order them:** The system processes every mapping rule by default, which can present problems when you apply a deny-all rule—for example, to deny access to all unknown users. In addition, replacement rules might interact with rules that contain wildcard characters. To minimize complexity, group rules by type and organize them in the following order: replacement rules, join/add/insert rules, allow/deny rules.

3.5 OneFS protocol audit

OneFS allows you to audit protocol activities. All audit data is stored and protected in the cluster file system and organized by audit topic. Starting in OneFS 8.0, protocol audit tracks and stores activity performed through SMB, NFS, and HDFS protocol connections in an access zone level. You can specify which events to log in each access zone. For example, you might want to audit the default set of protocol events in the System access zone but audit only successful attempts to delete files in a different access zone.

The audit events are logged on the individual nodes where the SMB, NFS, or HDFS client initiated the activity. The events are then stored in a binary file under `/ifs/.ifsvvar/audit/logs`. The logs automatically roll over to a new file after the size reaches 1 GB.

OneFS integration with Dell Common Event Enabler (CEE) enables third-party auditing applications to collect and analyze protocol auditing logs. Figure 5 shows a logical data flow in an NFS environment when using CEE and third-party applications. During the NFS client access the PowerScale cluster over NFS, the event

will be stored as configured to the cluster, the OneFS daemon `isi_adaudit_cee` exports protocol audit events to CEE, and then the CEE will forward protocol audit events to the consumer application.

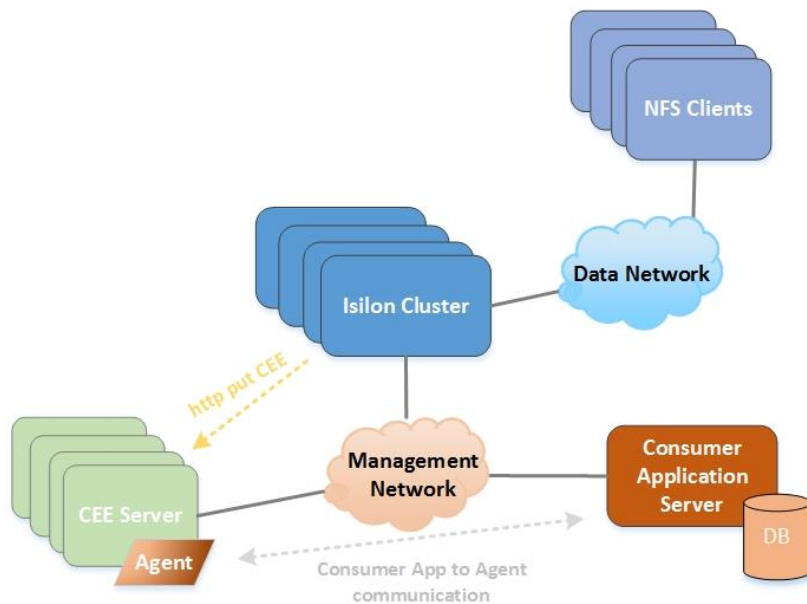


Figure 5 Protocol audit logical data flow

Note: Dell CEE does not support forwarding HDFS protocol events to a third-party application.

By default, the tracked events are create, close, delete, rename, and set_security. For the details of supported event types and its description, refer to the Auditing chapter of [OneFS Web Administration Guide](#). Because each audited event consumes system resources, we recommend that you configure zones only for events that are needed by your auditing application. In addition, we recommend that you install and configure third-party auditing applications before you enable the OneFS auditing feature. Otherwise, all the events that are logged are forwarded to the auditing application, and a large backlog causes a delay in receiving the most current events.

4 NFS client considerations

4.1 Linux client

Use NFS over TCP

The advantage using NFS over TCP is that it works far better than UDP over unstable networks. When using TCP, a single dropped packet can be retransmitted, without the retransmission of the entire RPC request, resulting in better performance over unstable networks.

In addition, TCP will handle network speed differences better than UDP, due to the underlying flow control at the network level. When using UDP, if network speeds of the client and server are not identical, dropped packets and retransmissions might cause performance to be extremely slow when the faster entity tries to send data to the slower entity.

The overhead incurred by the TCP protocol will result in somewhat slower performance than UDP under ideal network conditions, but the cost is not severe, and is often not noticeable without careful measurement. If you are using 10 GB Ethernet or above from end to end, you might also investigate the usage of jumbo frames. The high-speed network may allow the larger frame sizes without encountering increased collision rates, particularly if you have set the network to full duplex.

Client support for NFS over TCP is integrated into all Linux kernel 2.4 and later. You can check your client kernel version with command `uname -r` before you use NFS over TCP. By default, the client will attempt to mount NFS export with TCP if supported, you can also use option `-proto=tcp` to explicitly use TCP.

Security-Enhanced Linux (SELinux)

Security-Enhanced Linux (SELinux) is an implementation of a mandatory access control mechanism in the Linux kernel, checking for allowed operations after standard discretionary access controls are checked.

By default, NFS mounts on the client side are labeled with a default context defined by policy for NFS volumes. In common policies, this default context uses the `nfs_t` type. Depending on policy configuration, services such as Apache HTTP Server and MySQL may not be able to read files labeled with the `nfs_t` type. This may prevent file systems labeled with this type from being mounted and then read or exported by other services.

If you would like to mount an NFS volume and read or export that file system with another service, use the `-context` option when mounting to override the `nfs_t` type. For example, use the following context option to mount NFS volumes so that they can be shared through the Apache HTTP Server:

```
mount -o context="system_u:object_r:httpd_sys_content_t:s0" onefs_server:/export
/mnt/mount_point
```

SELinux can enforce rules on files and processes in a Linux system and on their actions based on defined policies. Because SELinux will introduce a performance overhead, disable the SELinux in the Linux client unless it is explicitly required. To permanently disable SELinux, follow these steps:

1. Configure `SELINUX=disabled` in the `/etc/selinux/config` file.
2. Reboot your system.
3. Check that your SELinux is disabled. Use the command `getenforce`, which returns `Disabled`, or check the status by using the command `sestatus`.

NFS read size (`rsize`) and write size (`wsiz`)

The mount options `rsiz` and `wsiz` specify the size of the data block size in bytes to be transferred at one time between the client and server. Setting a larger data block size would improve the performance. But be careful when changing these values; some older Linux kernels and network cards do not work well with larger block sizes. For most of modern systems, these settings are negotiated during mount and the server will specify the recommended settings. By default in OneFS, a `rsiz` of 128 KB and `wsiz` of 512 KB are advertised for NFSv3 Linux clients, but can be set as high as 1 MB. NFSv4.x defaults to 1 MB for both `rsiz` and `wsiz`.

Setting these values incorrectly will result in slower performance, so we recommended that you experiment before you change `rsiz` and `wsiz`. You can test your options with some simple commands if your network environment is not heavily used. Here is an example using command `dd` to do a simple test to see the speed while using different `rsiz` and `wsiz`.

Commands use to write file to a OneFS cluster and read file from a OneFS cluster:

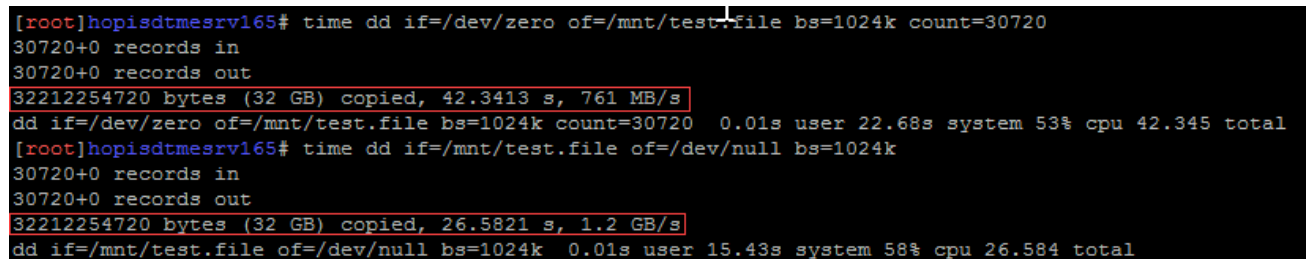
- Write file to OneFS cluster: this command transfers 30720 blocks of 1024k each (32 GB file of zeroed bytes) from a special file `/dev/zero` to the mount point `/mnt/`. We will measure the completion time. On a client machine, type:

```
time dd if=/dev/zero of=/mnt/test.file bs=1024k count=30720
```

- Read back the file from OneFS cluster into the null device on the client machine (`/dev/null`) by typing the following:

```
time dd if=/mnt/test.file of=/dev/null bs=1024k
```

Result shows in Figure 6 while mounting NFS with options `nfsvers=4,rsiz=1048576,wsiz=1048576`, which is the default value advised by OneFS. And leaves the other mount options as default.



```
[root]hopisdtmesrv165# time dd if=/dev/zero of=/mnt/test.file bs=1024k count=30720
30720+0 records in
30720+0 records out
32212254720 bytes (32 GB) copied, 42.3413 s, 761 MB/s
dd if=/dev/zero of=/mnt/test.file bs=1024k count=30720 0.01s user 22.68s system 53% cpu 42.345 total
[root]hopisdtmesrv165# time dd if=/mnt/test.file of=/dev/null bs=1024k
30720+0 records in
30720+0 records out
32212254720 bytes (32 GB) copied, 26.5821 s, 1.2 GB/s
dd if=/mnt/test.file of=/dev/null bs=1024k 0.01s user 15.43s system 58% cpu 26.584 total
```

Figure 6 Result with options `nfsvers=4,rsiz=1048576,wsiz=1048576`

Result shows in Figure 7 while mounting NFS with options `nfsvers=4,rsiz=32768,wsiz=32768`. And leaves the other mount options as default.

```
[root]hopisdtmesrv165# time dd if=/dev/zero of=/mnt/test.file bs=1024k count=30720
30720+0 records in
30720+0 records out
32212254720 bytes (32 GB) copied, 48.0409 s, 671 MB/s
dd if=/dev/zero of=/mnt/test.file bs=1024k count=30720 0.02s user 21.65s system 45% cpu 48.044 total
[root]hopisdtmesrv165# time dd if=/mnt/test.file of=/dev/null bs=1024k
30720+0 records in
30720+0 records out
32212254720 bytes (32 GB) copied, 78.4202 s, 411 MB/s
dd if=/mnt/test.file of=/dev/null bs=1024k 0.01s user 17.74s system 22% cpu 1:18.42 total
```

Figure 7 Result with options `nfsvers=4, rsize=32768, wsize=32768`

From these results, we can see that the performance will decrease if the inappropriate `rsize` and `wsize` values are used. We recommend that you do not specify these options when mounting a OneFS cluster NFS export. If you want to specify these options, we recommend that you verify them, as shown, before you apply these setting in your production environment.

Note: The test result shown may vary widely in different test environments—for example, in a different Linux OS/Kernel or network. If you achieve unexpected test results or if you would like to test different types of workloads (sequential, random, mix), use more complex benchmarks such as IOZone, FIO.

Hard mounts and soft mounts, timeout, and retransmissions

- **Hard mounts:** If you have mounted the NFS file system using a hard mount and if for any reason that share is not available, the client will repeatedly retry to contact the server. Once the NFS server is back online, the program will continue to perform undisturbed from the state where it was during server crash. Therefore, with a tradeoff of unavailable applications, the client will not lose any data during the NFS server is unavailable. We can use the mount option `intr`, which allows NFS requests to be interrupted if the server goes down or cannot be reached. The recommended settings are `hard` and `intr` options.
- **Soft mounts:** In case of a soft mount, when a program or application requests a file from the NFS file system, NFS client daemons will try to retrieve the data from the NFS server. But, if it does not get any response from the NFS server (due to any crash or failure of NFS server), the NFS client will report an error to the process on the client machine requesting the file access. A so-called "soft" timeout can cause silent data corruption in certain cases. As such, it is not suggested to use the `soft` option unless the application requires I/O operations to fail when the NFS server is unresponsive for a specified period. Failure will be the time requires for the iterations of the timeout value. Using NFS over TCP or increasing the value of the `retrans` option may mitigate some of the risks of using the `soft` option.
- **Timeout:** The timeout period is specified by the mount parameter `timeo` and is expressed in deciseconds (tenths of a second). By default, for NFS over TCP the default `timeo` value is 600 (60 seconds). The NFS client performs linear back off: After each retransmission the timeout is increased by time up to the maximum of 600 seconds.
- **Retransmissions:** The number of times the NFS client retries a request before it attempts further recovery action. This setting is irrelevant with hard mounts because they will retry indefinitely.

Setting these options incorrectly would cause unnecessary retransmissions in some cases, for example, in a high latency network. So unless you have an explicitly requirement to these options for your application and system, you should leave these options as default.

Sync and async mounts

The options is async by default, the NFS client delays sending application writes to the server until any of these events occur:

- Memory pressure forces reclamation of system memory resources.
- An application flushes file data explicitly with sync, msync, or fsync.
- An application closes a file with close.
- The file is locked/unlocked by fcntl.

In other words, under normal circumstances, data written by an application may not immediately appear on the server that hosts the file. If the `sync` option is specified on a mount point, any system call that writes data to files on that mount point causes that data to be flushed to the server before the system call returns control to user space.

So the sync write introduces a significant performance overhead while providing better data cache coherence between clients. Applications can use the `O_SYNC` open flag to force application writes to individual files to go to the server immediately without the use of the sync mount option. It is recommended to use async mounts and the application control the safe write behavior by writing with the `O_SYNC` open flag or flush data with sync.

Attribute caching (`ac/noac`)

Use the `noac` mount option to achieve attribute cache coherence among multiple clients. Almost every file system operation checks file attribute information. The client keeps this information cached for a period to reduce network and server load. When `noac` is in effect, a client's file attribute cache is disabled, so each operation that needs to check a file's attributes is forced to go back to the server. Besides, the `noac` option forces application writes to become synchronous so that a client sees changes to a file upon opening, at the cost of many extra network operations. By default, the attribute caching is enabled when mounting the NFS. Enable the attribute caching to improve the attribute checking performance and reduce the NFS operation latency.

nconnect

This mount option exists in all Linux distributions with kernel 5.3 or higher and can be set up to a limit of 16. It allows clients establish multiple TCP connections to a OneFS cluster node for a specific NFS version, as it can be used with NFSv3 and NFSv4.x with each individual version being tracked separately. This option is set during the client's first mount for a particular OneFS node and NFS version combination. If the client performs another NFS mount for the same OneFS node and NFS version, it will reuse the connections established by the first mount. Subsequent mount commands cannot override the `nconnect` value already established. To set a new `nconnect` value, all client-mounted NFS file systems which point to a certain OneFS node and NFS version must be unmounted, and you must remount the NFS mount with the desired `nconnect` value.

4.2 macOS client

For more details about using OneFS NFS service on macOS client and its configurations, see [Using macOS Clients with PowerScale OneFS](#).

5 NFS security considerations

5.1 Network security considerations

Network security is always the important area to focus on, attacks from a malicious attacker would result in a disaster and may result in service interruption to end users. As a security recommendation, shown in Figure 8, you should setup an external firewall with appropriate rules and policies to allow only the trusted clients and servers can access the cluster. Meanwhile, allow restricted access only to the ports that are required for communication and block access to all other ports on the cluster.

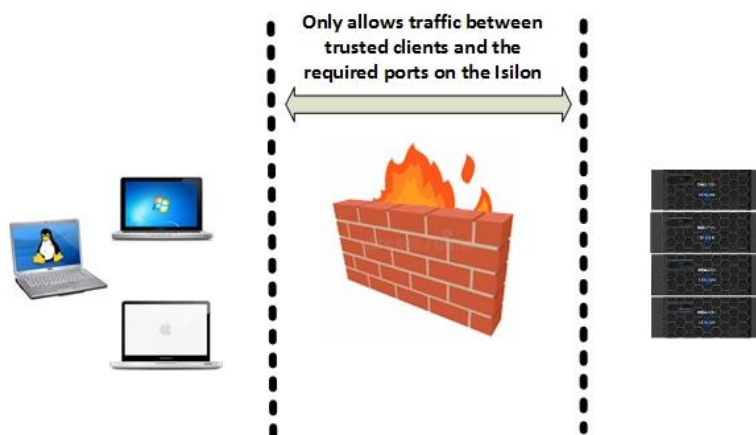


Figure 8 Protect PowerScale system with an external firewall

Table 3 shows the required ports for a client to access data in OneFS cluster over NFS protocol. As NFSv3 requires additional auxiliary protocol (mount, NLM, NSM) to provide mount service and lock capability, all of the ports in the table are required to access cluster using NFSv3. For NFSv4.x, a single protocol provides all functionalities that NFSv3 offers and only supports TCP as the transport protocol, so it is firewall friendly and only the TCP 2049 is required for a client to access cluster using NFSv4.x.

Table 3 TCP/UDP port requirement for NFS service

Port	Service	Protocol	Connection	Usage description
2049	nfs	TCP/UDP	Inbound	As NFSv3 supports both TCP and UDP in OneFS, so both of two transport protocols ports are required for NFSv3. However, NFSv4.x supports only TCP in OneFS, so only the TCP 2049 port is needed if only the NFSv4.x service is required in your environment.
300	mountd	TCP/UDP	Inbound	NFSv3 mount service.
302	statd	TCP/UDP	Inbound	NFSv3 Network Status Monitor (NSM)
304	lockd	TCP/UDP	Inbound	NFSv3 Network Lock Manager (NLM)
111	rpc.bind	TCP/UDP	Inbound	ONC RPC portmapper that is used to locate services such as NFS, mountd. Only used by NFSv3 if NFSv4.x running on the standard registered TCP port 2049.

See section 6 (NFSv3 over RDMA) for details about the RDMA-related port.

5.2 Authentication

OneFS can be configured to authenticate users with Kerberos by using Active Directory Kerberos or a stand-alone MIT Kerberos. The recommendation is to authenticate all users with Kerberos if high security level is required, but be aware of the performance impact by Kerberos. If you are using Kerberos, ensure both the OneFS cluster and your client use either Active Directory or the same NTP server as their time source. Kerberos is a protocol that relies on time synchronization between system components. A time drift among the system components will cause authentication failure. Kerberos on OneFS writes log messages to `/var/log/lsassd.log` and `/var/log/lwiod.log`. When Kerberos is used with NFS, Kerberos writes log messages to `/var/log/nfs.log`.

With NFSv3 and prior, when you authenticate the user using AUTH_SYS security flavor, the UID will be included in every NFS operation and checked by the server. Therefore, someone on a different computer can access the user Jane (UID 1000) file by creating a user Jane (UID 1000) on the computer. Using Kerberos authentication would mitigate this situation, but is still not completely secure, because Kerberos was only applied to the NFS packets and not the auxiliary services like NLM, NSM, and mountd.

NFSv4.x improved NFS security greatly by implementing a single port, ACLs, domain names and contains tightly integrated support for Kerberos, among other improvements. You must have an identical NFSv4.x domain name on OneFS cluster and NFSv4.x clients. With NFSv4.x domain, the NFSv4.x represents users and groups in the form of `user@domain` or `group@domain` in the results of a get attribute (GETATTR) operation and in the arguments of a set attribute (SETATTR) operation. Figure 9 is a capture of NFSv4.x GETATTR operation. As Figure 9 shows, the user/group names have an NFSv4.x domain suffix `@vlab.local` in the GETATTR operation.

```

4 Attr mask[1]: 0x00b0a23a (Node, NumLinks, Owner, Owner_Group, RawDev, Space_Used, Time_Access, Time_Metadata, Time_Modify, Mounted_on_FileId)
  ▸ reco_attr: Node (33)
  ▸ reco_attr: NumLinks (35)
  4 reco_attr: Owner (36)
    4 fattr4_owner: test01@vlab.local
      length: 17
      contents: test01@vlab.local
      fill bytes: opaque data
    4 reco_attr: Owner_Group (37)
      4 fattr4_owner_group: Isilon Users@vlab.local
        length: 23
        contents: Isilon Users@vlab.local
        fill bytes: opaque data

```

Figure 9 NFSv4 user and group format

Therefore, in the environment that requires high security for NFS, use NFSv4.x instead of NFSv3 and integrate Kerberos authentication with NFS. Note that the configuration is different when using Active Directory Kerberos or MIT Kerberos. Before configuring Kerberos in your NFS environment, it is important to understand how it works. You can obtain a thorough explanation from the online documentation [How Kerberos Authentication Works](#). For the configuration of OneFS NFS Kerberos, refer to white paper [Integrating OneFS with Kerberos Environment for Protocols](#). Kerberos is tied to time synchronization, so whenever you use Kerberos in your environment, ensure your cluster and clients have an NTP server to synchronize time.

As OneFS supports Kerberos authentication for both NFSv3 and NFSv4.x. There are four types of security type supported by OneFS (UNIX, Kerberos5, Kerberos5 Integrity, Kerberos5 Privacy). You can use `sec` mount option on NFS client to enable Kerberos for a mount. Table 4 shows the types of security for `sec` option.

Table 4 Mount security types

Options	Description
sec=sys	The default setting, which uses local UNIX UIDs and GIDs by means of AUTH_SYS to authenticate NFS operations.
sec=krb5	Use Kerberos V5 instead of local UNIX UIDs and GIDs to authenticate users.
sec=krb5i	Use Kerberos V5 for user authentication and performs integrity checking of NFS operations using secure checksums to prevent data tampering.
sec=krb5p	Use Kerberos V5 for user authentication, integrity checking, and encrypts NFS traffic to prevent traffic sniffing. This is the most secure setting, but it also has the most performance overhead involved.

Client configuration is required before you can mount a NFS using Kerberos, several key configurations are listed below:

- The kernel needs to have the `rpcsec_gss_krb5` and `auth_rpcgss` options configured as a module. To configure the module, using these commands in the following order `modprobe auth_rpcgss`, `modprobe rpcsec_gss_krb5`, `depmod -a`. If the module is not configured, you will find an error message in the client's syslog as shown below.

```
gss_create: Pseudoflavor 390003 not found!
RPC: Couldn't create auth handle (flavor 390003)
```

- Add `SECURE_NFS="yes"` to file `/etc/sysconfig/nfs` on the client. And start the `rpc.gssd` service using command `service rpcgssd restart`. If this setting is not configured, when you mount the NFS, the mount becomes unresponsive and the below error displays in the log.

```
May 6 18:12:29 client1 kernel: [485467.135178] RPC: AUTH_GSS upcall timed out.
May 6 18:12:29 client1 kernel: [485467.135182] Please check user daemon is running.
```

The Kerberos will provide high secure authentication, integrity, privacy service while introducing extra cost on the computer resources, and it might affect your system performance. It is highly recommended to make enough measurement before applying Kerberos settings on your NFS environment.

5.3 NFSv4.x ACL

OneFS has its own internal ACL representation, and it is compatible with NFSv4.x ACL. When NFSv4.x clients access the files/directories on OneFS, OneFS translates its internal ACL to NFSv4.x ACL and sends it to the client. On OneFS, you can use `chmod` command to manage and manipulate ACL, for detailed usage, refer to the man page of `chmod` on OneFS. On NFSv4.x client, you can use `nfs4_setfacl` and `nfs4_getfacl` to manage ACL, for detailed usage, refer to their man pages.

OneFS ACE permissions for file system objects

Similar to the Windows permission level, the OneFS divides permissions into the following three types.

- Standard ACE permissions: Apply to any object in the file system, see Table 5.
- Generic ACE permissions: Each of them maps to a bundle of specific permissions, see Table 6.

- Constant ACE permissions: Each of them is a specific permission for a file system object, see Table 7.

The standard ACE permissions that can appear for a file system object are shown in Table 5.

Table 5 OneFS standard ACE permissions

ACE permission	Apply to	Description
std_delete	Directory/File	The right to delete the object
std_read_dac	Directory/File	The right to read the security descriptor, not including the SACL
std_write_dac	Directory/File	The right to modify the DACL in the object's security descriptor
std_write_owner	Directory/File	The right to change the owner in the object's security descriptor
std_synchronize	Directory/File	The right to use the object as a thread synchronization primitive
std_required	Directory/File	Maps to std_delete, std_read_dac, std_write_dac, and std_write_owner

The generic ACE permissions that can appear for a file system object are shown in Table 6.

Table 6 OneFS generic ACE permissions

ACE permission	Apply to	Description
generic_all	Directory/File	Read, write, and execute access. Maps to file_gen_all or dir_gen_all
generic_read	Directory/File	Read access. Maps to file_gen_read or dir_gen_read
generic_write	Directory/File	Write access. Maps to file_gen_write or dir_gen_write
generic_exec	Directory/File	Execute access. Maps to file_gen_execute or dir_gen_execute
dir_gen_all	Directory	Maps to dir_gen_read, dir_gen_write, dir_gen_execute, delete_child, and std_write_owner
dir_gen_read	Directory	Maps to list, dir_read_attr, dir_read_ext_attr, std_read_dac, and std_synchronize
dir_gen_write	Directory	Maps to add_file, add_subdir, dir_write_attr, dir_write_ext_attr, std_read_dac, and std_synchronize
dir_gen_execute	Directory	Maps to traverse, std_read_dac, and std_synchronize
file_gen_all	File	Maps to file_gen_read, file_gen_write, file_gen_execute, delete_child, and std_write_owner
file_gen_read	File	Maps to file_read, file_read_attr, file_read_ext_attr, std_read_dac, and std_synchronize
file_gen_write	File	Maps to file_write, file_write_attr, file_write_ext_attr, append, std_read_dac, and std_synchronize
file_gen_execute	File	Maps to execute, std_read_dac, and std_synchronize

The constant ACE permissions that can appear for a file system object are shown in Table 7.

Table 7 OneFS constant ACE permissions

ACE permission	Apply to	Description
modify	File	Maps to file_write, append, file_write_ext_attr, file_write_attr, delete_child, std_delete, std_write_dac, and std_write_owner
file_read	File	The right to read file data
file_write	File	The right to write file data
append	File	The right to append to a file
execute	File	The right to execute a file
file_read_attr	File	The right to read file attributes
file_write_attr	File	The right to write file attributes
file_read_ext_attr	File	The right to read extended file attributes
file_write_ext_attr	File	The right to write extended file attributes
delete_child	Directory/File	The right to delete children, including read-only files within a directory. It is currently not used for a file, but can still be set for windows compatibility.
list	Directory	List entries
add_file	Directory	The right to create a file in the directory
add_subdir	Directory	The right to create a subdirectory
traverse	Directory	The right to traverse the directory
dir_read_attr	Directory	The right to read directory attributes
dir_write_attr	Directory	The right to write directory attributes
dir_read_ext_attr	Directory	The right to read extended directory attributes
dir_write_ext_attr	Directory	The right to write extended directory attributes

Mapping OneFS ACE permissions to NFSv4.x

This section describes how OneFS maps file and directory permissions when using `chmod` command to modify the ACL from the OneFS or using `nfs4_setfacl` to modify the ACL from the NFSv4.x client. For the details of NFSv4.x ACE permission in the Linux tool (`nfs4_setfacl/nfs4_getfacl`), see the man page for [nfs4_acl](#). For details of NFS4.x ACE permission standard access mask, see the NFSv4.0 [RFC3530](#) section 5.11.2. ACE Access Mask.

The Table 8 shows the ACE permission mapping between OneFS and NFSv4.x.

Table 8 ACE permission mapping between OneFS and NFSv4.x

OneFS internal ACE permission	NFSv4.x ACE permission letter in Linux	RFC3530 standard permission bitmask	Apply to
std_delete	d	ACE4_DELETE	Directory/File
std_read_dac	c	ACE4_READ_ACL	Directory/File
std_write_dac	C	ACE4_WRITE_ACL	Directory/File
std_write_owner	o	ACE4_WRITE_OWNER	Directory/File
std_synchronize	y	ACE4_SYNCHRONIZE	Directory/File
file_read	r	ACE4_READ_DATA	File
file_write	w	ACE4_WRITE_DATA	File
append	a	ACE4_APPEND_DATA	File
execute	x	ACE4_EXECUTE	File
file_read_attr	t	ACE4_READ_ATTRIBUTES	File
file_write_attr	T	ACE4_WRITE_ATTRIBUTES	File
file_read_ext_attr	n	ACE4_READ_NAMED_ATTRS	File
file_write_ext_attr	N	ACE4_WRITE_NAMED_ATTRS	File
delete_child	D	ACE4_DELETE_CHILD	Directory
list	r	ACE4_LIST_DIRECTORY	Directory
add_file	w	ACE4_ADD_FILE	Directory
add_subdir	a	ACE4_ADD_SUBDIRECTORY	Directory
traverse	x	ACE4_EXECUTE	Directory
dir_read_attr	t	ACE4_READ_ATTRIBUTES	Directory
dir_write_attr	T	ACE4_WRITE_ATTRIBUTES	Directory
dir_read_ext_attr	n	ACE4_READ_NAMED_ATTRS	Directory
dir_write_ext_attr	N	ACE4_WRITE_NAMED_ATTRS	Directory
std_required	dcCo		Directory/File
modify	wadTNC		File
generic_all OR file_gen_all	rwadxtTnNcCoy		File

OneFS internal ACE permission	NFSv4.x ACE permission letter in Linux	RFC3530 standard permission bitmask	Apply to
generic_read OR file_gen_read	rtncy		File
generic_write OR file_gen_write	waTNcy		File
generic_exec OR file_gen_execute	xcy		File
generic_all OR dir_gen_all	rwaDdxtTnNcCoy		Directory
generic_read OR dir_gen_read	rtncy		Directory
generic_write OR dir_gen_write	waTNcy		Directory
generic_exec OR dir_gen_execute	xcy		Directory

5.4 NFSv4.x pseudo-file system

The OneFS cluster supports the NFSv4.x pseudo-file system in compliance with the RFC3530 standard. NFSv4.x servers present all the exported file systems within a single hierarchy. When the server exports a portion of its namespace, the server creates a pseudo-file system which is a structure containing only directories. It has a unique file system id (fsid) that allows a client to browse the hierarchy of an exported file system. An NFSv4.x client can use LOOKUP and READDIR operations to browse seamlessly from one export to another. The clients' view of a pseudo-file system will be limited to paths to which the clients has permission to access.

To have a better understanding about pseudo-file system, assume an OneFS cluster has the following directory structure, shown as Figure 10.

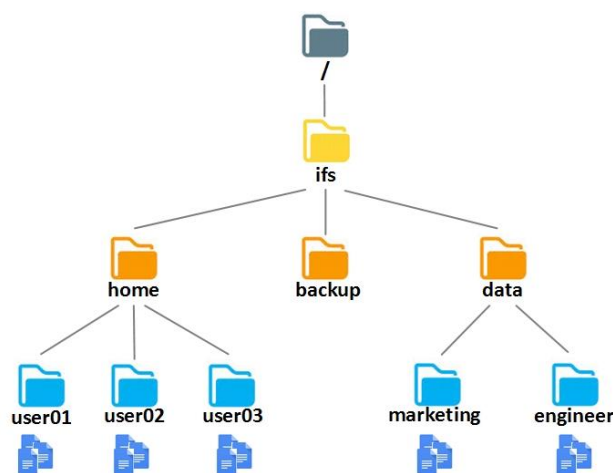


Figure 10 Server side directory structure

Consider a scenario where an application on a server need to access portions of the directories (assuming `/ifs/home/user01`, `/ifs/home/user02`, and `/ifs/data/marketing`) but require mounting these directories using a single mount point to access the files. To satisfy the requirement of the application, we will export these directories separately. Meanwhile, there is an export for `/ifs/data/engineer`, and this export is not accessible by the application.

In NFSv4.x, the export list and the server hierarchy are disjointed, as illustrated in Figure 11. When the cluster exports the portions of directories, the server creates a pseudo-file system to allow the client to access the authorized exported paths from a single common root. The client only needs to mount the appropriate path of the pseudo-file system, for example, mount the `/ifs` to the client directly, and the client can access any one of the export paths that are required by the application.

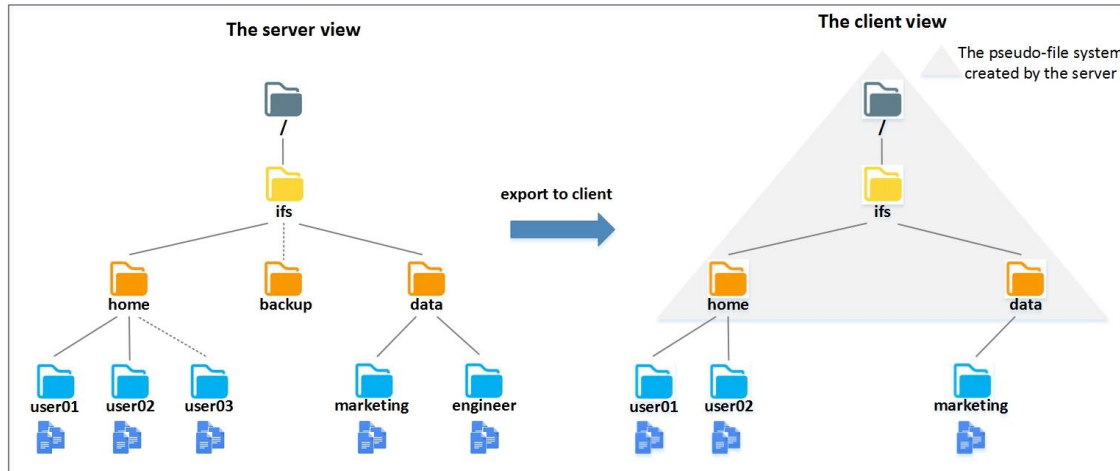


Figure 11 NFSv4.x pseudo-file system

If NFSv3 is used in this scenario, the client must export the entire `/ifs` namespace to allow the application access data in the disjoint directories with a single mount point. This will result in a huge security problem as the whole OneFS cluster namespace is exposed to the client and the client can even access the data that is not used for the application.

The pseudo-file system is a considerable advantage for its access security and flexibility of limiting only part of the namespace that the client can see and access. Use NFSv4.x pseudo-file system instead of NFSv3 in a similar scenario above to provide a more secure access control.

Note: In NFSv3, a client browsing the server exports is provided through the MOUNT protocol, every export has its own root file handle. When the client running the command `showmount -e server_address` to obtain the exports list on the server, the MOUNT protocol will enumerate the server's exports list and return to the client. In NFSv4.x, a client browses the server exports which uses the same root handle through the pseudo-file system, so in an NFSv4.x environment, `showmount` command is not supported to get an exports list on server.

6 NFSv3 over RDMA

6.1 NFSv3 over RDMA overview

Remote Direct Memory Access (RDMA) originated with InfiniBand and evolved gradually on Ethernet network environment. Currently, the following network protocols support RDMA on Ethernet, including Internet Wide Area RDMA Protocol (iWARP), and RDMA Over Converged Ethernet (RoCE). For more details, see [RoCE](#).

NFS over RDMA is defined in [RFC8267](#). Starting with OneFS 9.2.0, OneFS supports NFSv3 over RDMA by leveraging the ROCEv2 (also known as Routable RoCE or RRoCE) network protocol. neither ROCEv1 nor NFSv4.x over RDMA IS supported in the OneFS 9.2 release. With NFSv3 over RDMA support, direct memory access between OneFS and NFSv3 clients is available with consuming less client CPU resource, improving OneFS cluster network performance with lower latency, lower CPU load and higher throughput.

The Figure 12 shows the architecture of NFSv3 over RDMA in OneFS. NFSv3 is implemented over RDMA for data transferring, while its auxiliary protocols (mount, nlm, nsm, rpc portmapper) still works on TCP/UDP. You must add PowerScale nodes RoCEv2 capable front-end network interfaces into an IP pool before the NFSv3 clients can access OneFS cluster data over RDMA.

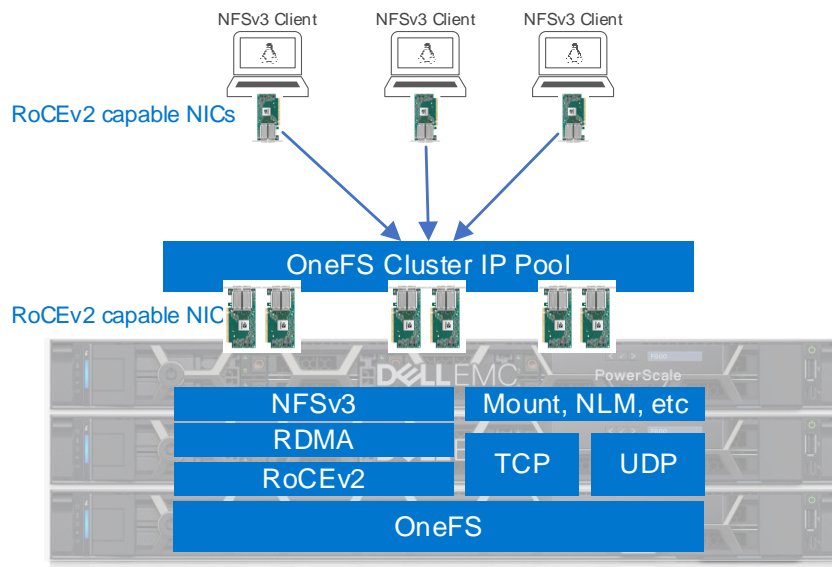


Figure 12 OneFS NFSv3 over RDMA architecture

The following is the list of PowerScale/Isilon node types and hardware that support NFSv3 over RDMA.

- **Node types:** All Gen6 (F800/F810/H600/H500/H400/A200/A2000), F200, F600, F900
- **Front-end network:** Mellanox ConnectX-3 Pro, ConnectX-4 and ConnectX-5 network adapters that deliver 25/40/100 GigE speed.
- **TCP/UDP port requirement:** See Table 9.

Table 9 TCP/UDP port requirement for NFSv3 over RDMA

Port	Service	Protocol	Usage description
4791	RoCEv2	UDP	In RoCEv2, the RDMA payload is encapsulated as UDP payload with the 4791 UDP destination port.
300	mountd	TCP/UDP	NFSv3 mount service.
302	statd	TCP/UDP	NFSv3 Network Status Monitor (NSM)
304	lockd	TCP/UDP	NFSv3 Network Lock Manager (NLM)
111	rpc.bind	TCP/UDP	ONC RPC portmapper that is used to locate services such as NFS, mountd. Only used by NFSv3 if NFSv4.x running on the standard registered TCP port 2049.

Note: We recommend enabling flow control on switch ports to achieve good performance when losing network packet. When mounting NFS export over RDMA, you need to specify an NFSv3 over RDMA port 20049. The port is used to for RPC binding of RDMA interconnect internally and is not required to be allowed in network firewalls. For more details, see [RFC5666 RPC Binding](#).

6.2 Management options

New configuration options are introduced in order to manage NFSv3 over RDMA feature. Including enable/disable NFS over RDMA globally, filter RoCEv2 capable network interfaces for an IP pool, and check ROCEv2 capability of network interfaces.

6.2.1 Enable/disable NFS over RDMA globally

This allows storage administrators to enable or disable NFSv3 over RDMA capability cluster wide. Below shows the option in CLI command and Figure 13 shows the option in WebUI.

```
# isi nfs settings global view
NFS Service Enabled: Yes
  NFSv3 Enabled: Yes
    NFSv3 RDMA Enabled: Yes
  NFSv4 Enabled: Yes
    v4.0 Enabled: No
    v4.1 Enabled: Yes
    v4.2 Enabled: Yes
Rquota Enabled: No
```


UNIX sharing (NFS) Current Access Zone : System

NFS exports
NFS aliases
Export settings
Global settings
Zone settings

Edit NFS global settings

☒ NFS export service enabled

☒ NFSv3 enabled

☒ NFSv3 over RDMA enabled i

NFSv4

☐ NFSv4.0 enabled

☒ NFSv4.1 enabled

☒ NFSv4.2 enabled

Figure 13 Enable/disable NFS over RDMA globally

6.2.2 Filter RoCEv2 capable network interfaces for an IP pool

This option allows administrators to proactively create IP pools that contains only RoCEv2 capable network interfaces. It is not allowed if you try to add a RoCEv2 incapable network interface into the NFSv3 RDMA RRoCE only IP pools. More specifically, this option makes NFS failover using dynamic IP pool still work with NFSv3 over RDMA scenarios. See section 3.2 for more details about dynamic IP pool failover.

In CLI, this option is **--nfsv3-rroce-only** shown as below. The equivalent option in WebUI is called **Enable NFSv3 over RDMA**, highlighted in Figure 14, once the option enabled, all RoCEv2 incapable network interfaces are hidden and removed from the IP pool.

```
# isi network pools view groupnet0.40g.40gpool
      ID: groupnet0.40g.40gpool
  Groupnet: groupnet0
    Subnet: 40g
      Name: 40gpool
      ...
      ...
      ...
  Static Routes: -
  NFSv3 RDMA RRoCE only: Yes
```

Edit pool details

* = Required field

Settings

* **Name**
40gpool

Description
NFSv3 over RDMA IP pool

* **Access Zone**
System

IP range
Remove IP range 172.16.200.29 - 172.16.200.36
+ Add an IP range

Pool interface members

☒ **Enable NF SoRDMA**

Available

LNN	Interface
-----	-----------

In pool

LNN	Interface
1	40gige-1
1	40gige-2
2	40gige-1
2	40gige-2
3	40gige-1
3	40gige-2
4	40gige-1

Add →
← Remove

Figure 14 Enable NFSv3 RDMA RRoCE only for an IP pool

6.2.3 Check ROCEv2 capability of network interfaces

Starting from OneFS 9.2, RoCEv2 capable network interface contains a flag **SUPPORTS_RDMA_RRoCE**. This flag is only usable through the CLI command shown below.

```
f8101-1# isi network interfaces list -v --nodes=1
IP Addresses: 172.16.200.29
LNN: 1
Name: 40gige-1
NIC Name: mlxen0
Owners: groupnet0.40g.40gpool
Status: Up
VLAN ID: -
Default IPv4 Gateway: -
Default IPv6 Gateway: -
MTU: 9000
Access Zone: System
Flags: ACCEPT_ROUTER_ADVERT, SUPPORTS_RDMA_RRoCE
```

6.3 Key considerations

This section lists several key considerations when using OneFS NFSv3 over RDMA feature.

- Match the Maximum Transfer Unit (MTU) on both the OneFS cluster and NFSv3 client. Mismatch MTU size may result in NFS operations becoming unresponsive and breaking your workload.
- Dynamic IP pools failover considerations.
 - Dynamic IP pools is the current network configuration recommendation for OneFS NFSv3. The purpose of dynamic IP pools is to allow client workflow to continue processing when a node goes down. Dynamic IP pools provide an IP-failover ability to move an IP from one network interface card (NIC) to another NIC on any node.
 - IP failover from a ROCEv2 capable interface to a ROCEv2 incapable interface is not supported. Therefore, enabling NFSv3 RDMA RRoCE only option in the RDMA IP pool is recommended.
 - When OneFS cluster and NFSv3 clients are connected through L2 Switch directly, the IP failover may fail for NFSv3 over RDMA workflow. This is caused by the client RDMA stack cannot handle [Gratuitous ARP](#) properly. Therefore, we recommend placing a router or L3 Switch between the OneFS cluster nodes and the NFSv3 over RDMA clients.
- Enable flow control on switch ports to achieve good performance when losing network packet.
- NFSv3 over RDMA does not support aggregated interfaces and VLAN tagged interfaces.
- IPv6 is not supported when using NFSv3 over RDMA.
- Making sure your NFSv3 client is running on RoCEv2 mode.

7 SmartQoS

7.1 SmartQoS overview

SmartQoS allows administrators to apply an OPS ceiling or limit to specified workloads for business workload prioritization: SmartQoS:

- Is applicable to NFSv3, NFSv4, NFSoRDMA, SMB, and S3
- Enables IT infrastructure teams to achieve performance SLAs
- Enables throttling of rogue or low-priority workloads, thus prioritizing other business-critical workloads
- Helps minimize DU events due to overloaded clusters

Note: SmartQoS is a post-commit feature, which means it will only work after the upgrade has been committed successfully.

7.2 Configuration ops limitation

SmartQoS is implemented based on the partitioned performance in OneFS. Concepts such as dataset, workload, and so on are still applied to SmartQoS. To configure an ops limitation using SmartQoS:

1. Enable SmartQoS in OneFS.
2. Create a SmartQoS dataset.
3. Pin a workload from the dataset.
4. Set an ops limitation for a pinned workload.

7.2.1 Enable SmartQoS in OneFS

The SmartQoS feature is enabled by default. You can view the current settings by running the following CLI command:

```
# isi performance settings view
                        Top N Collections: 1024
                Time In Queue Threshold (ms): 10.0
    Target read latency in microseconds: 12000.0
    Target write latency in microseconds: 12000.0
                Protocol Ops Limit Enabled: Yes
```

To disable the SmartQoS feature, run the following command:

```
# isi performance settings modify --protocol-ops-limit-enabled=false
protocol_ops_limit_enabled: True -> False
```

7.2.2 Create SmartQoS dataset

A dataset is used to categorize workloads by various metrics including:

- export_id
- local_address
- protocol

- share_name
- zone_name
- groupname
- path
- remote_address
- username

In this example, we use protocol and path to create a test dataset.

1. Use the following command for creation:

```
# isi performance datasets create --name ds1 protocol path
Created new performance dataset 'ds1' with ID number 1.
Note: Resource usage tracking by 'path' metric is only supported by SMB
and NFS.
```

You can use the WebUI for the same purpose:

- a. In the WebUI, click **Smart QOS** under **Cluster management**.

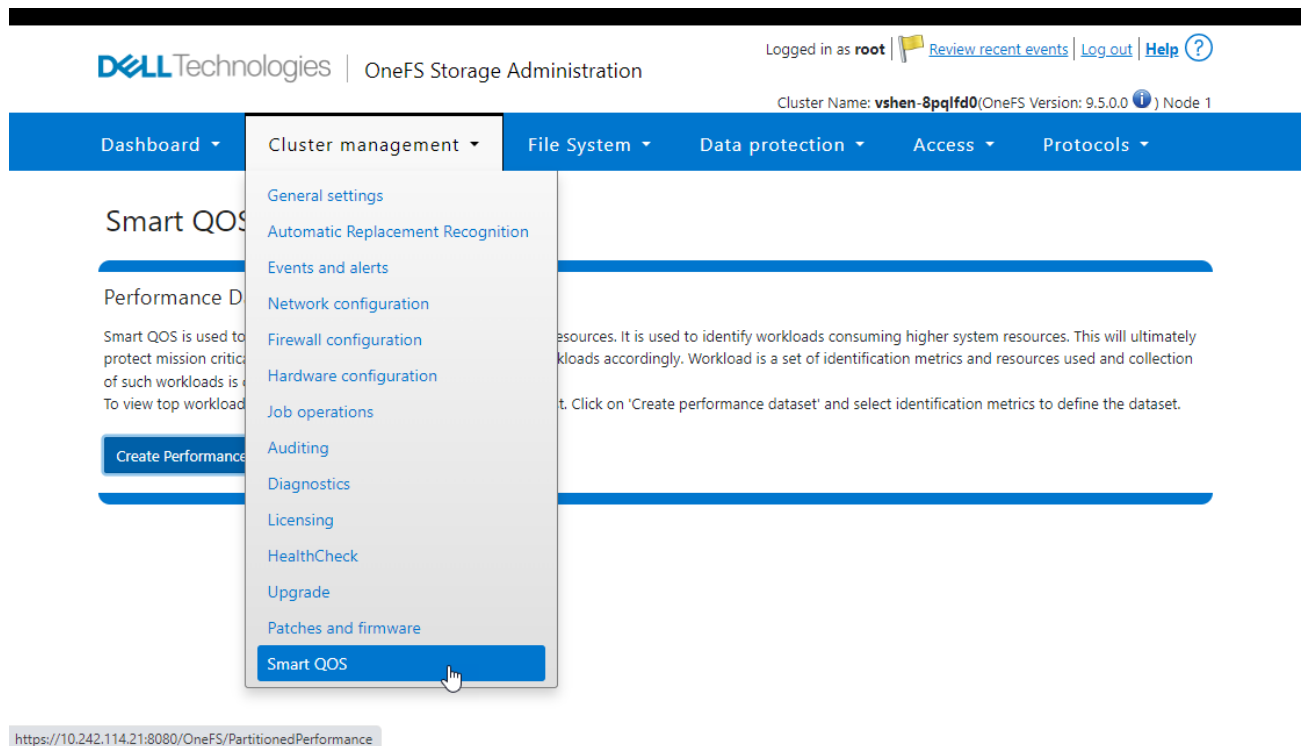


Figure 15 Smart QoS selection

- b. Click **Create Performance Dataset**:

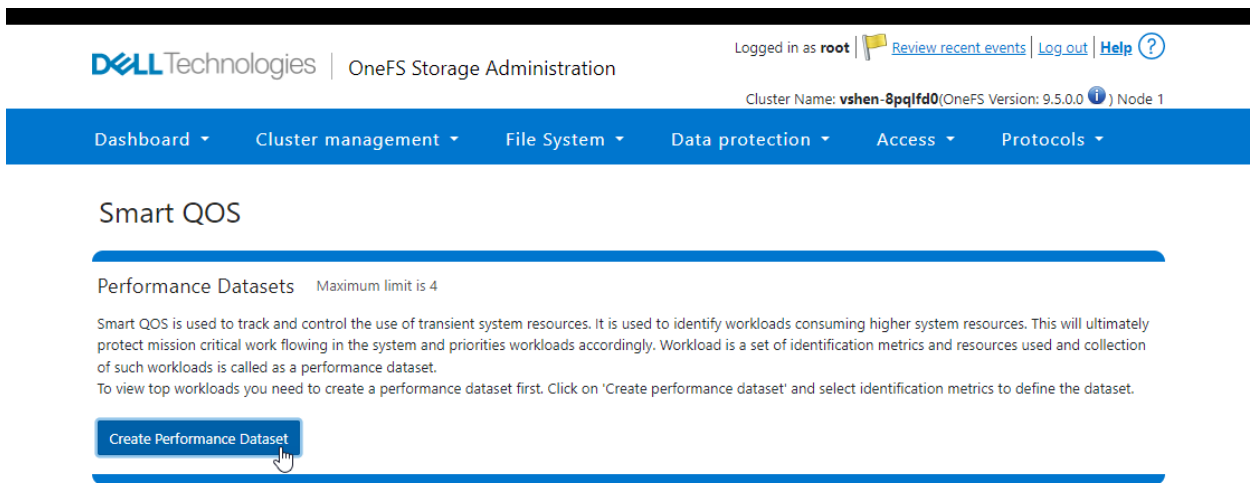


Figure 16 Crate Performance Dataset

- c. Enter your **dataset name**, select **Path** and **Protocol**, and click **Save**:

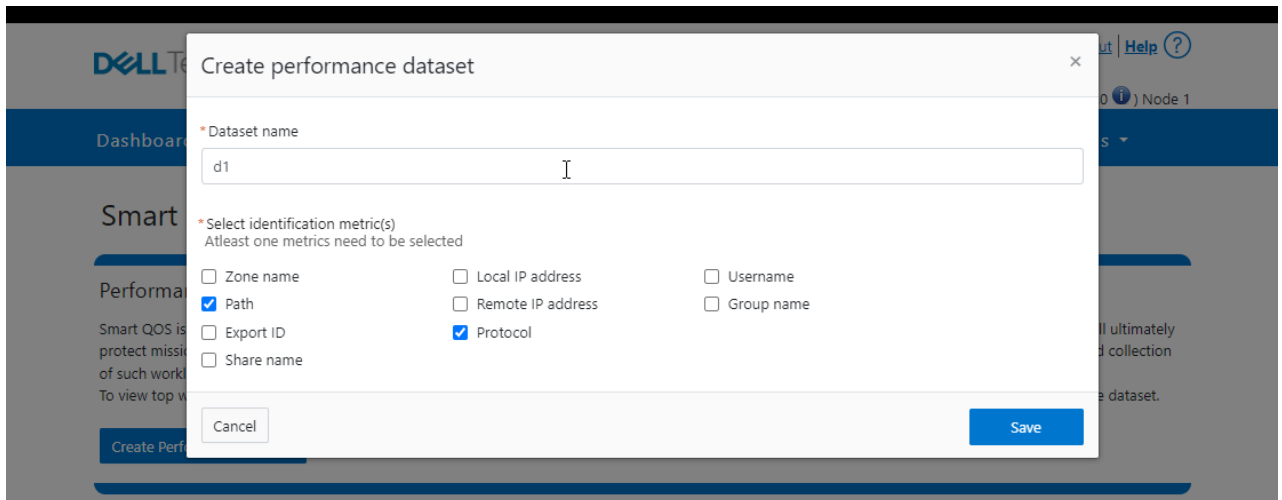


Figure 17 Create performance dataset

2. Run the following command to list all the datasets

```
# isi performance datasets list
```

7.2.3 Pin a workload from the dataset

After you create the dataset, you can pin a workload by specifying the metric values. In this example, we use the dataset created in section 7.2.2 and set the following metric values to pin the workload we need:

- **Protocol** is `nfs3`
- **Path** is `/ifs/data/client_export`

Run the following command:

```
# isi performance workloads pin ds1 protocol:nfs3 path:/ifs/data/client_export
Pinned performance dataset workload with ID number 100.
```

To pin the workload in the WebUI:

1. Click the **Pin Workload**.

Smart QoS

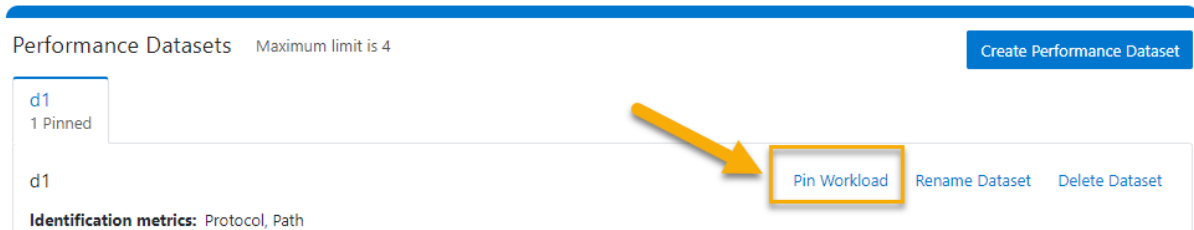


Figure 18 Pin Workload

2. Select the protocol from the drop-down list, enter the path, and click **Pin workload**.

To list all the pinned workloads from a specified dataset, run the following command:

```
# isi performance workloads list ds1
```

7.2.4 Set an ops limit for a pinned workload

For a pinned workload in a dataset, you can set the limit for the protocol ops by running the following command:

```
# isi performance workloads modify <dataset> <workload ID> --limits
protocol_ops:<value>
```

In this example, the name of the dataset is ds1 and the workload ID is 100. We also set the ops limit to 10:

```
# isi performance workloads modify ds1 100 --limits protocol_ops:10
protocol_ops: 18446744073709551615 -> 10
```

8 Useful NFS commands and tools

8.1 isi statistics command

The isi statistics command is an advanced tool that can be used to obtain various kinds of statistics that can help you measure the usage and performance of a PowerScale OneFS cluster. Isi statistics is a versatile utility with the various subcommand-level. It provides two subcommand-levels for protocols:

- Protocol: Display cluster usage statistics organized by communication protocol.
- Pstat: Generate the specified protocol detailed statistics along with CPU, OneFS, network, and disk statistics.

The NFS operations are grouped according to the classes in isi statistics as the Table 10 shows.

Table 10 NFS operations classes in isi statistics

Class	NFSv3 operations	NFSv4.x operations	Description
read	read	read	File and stream reading
write	write	write	File and stream writing
create	create, link, mkdir, mknod, symlink	create, link	File, link, node, stream, and directory creation
delete	remove, rmdir	remove	File, link, node, stream, and directory deletion
namespace_read	access, getattr, lookup, readdir, readdirplus, readlink	access, getattr, getfh, lookup, lookupp, nverify, readdir, readlink, secinfo, verify	Attribute, stat, and ACL reads; lookup, directory reading
namespace_write	rename, setattr	rename, setattr	Renames; attribute setting; permission, time, and ACL writes
file_state		close, delegpurge, delegreturn, lock, lockt, locku, open, openattr, openconfirm, opendowngrade, release_lockowner	Open, close; locking: acquire, release, break, check; notification
session_state		Renew, setclientid, setclientid_cfrm	Negotiation, inquiry, or manipulation of protocol connection or session state
other	commit, fsinfo, noop, null, pathconf, statfs	cb_compound, cb_getattr, cb_null, cb_recall, commit, compound, null, putfh, putpubfh, putrootfh, restorefh, savefh	File-system information, other uncategorizable operations

Isi statistics protocol

You get the detailed NFS protocol operations performance data by running the following command:

```
isi statistics protocol list --protocols=nfs3,nfs4 --sort=TimeAvg --degraded
```


As shown in Figure 11, the result is sorted by TimeAvg. You can also filter the output by adding the `--classes` option to get the specific class of the NFS operations, or use the `--totalby=class` to observe which class of operation is taking the longest.

- **Ops** - Total number of operations per second coming to and from the node.
- **In** - The amount of data in B/s coming into the cluster, this correlates to write operations from clients to the node.
- **Out** - The amount of data in B/s coming out of the cluster, this correlates to read operations from the node to clients.
- **TimeAvg** - The average amount of latency measured in microseconds (1000 microseconds = 1 millisecond) for the protocol ops to the node.
- **TimeStdDev** - Measures the standard deviation of ops, the lower the number the closer to the average latency most ops are. The larger the number the more varied the dataset is.
- **Node** - The node number in the cluster for which we are measuring.
- **Proto** - The protocol that we are measuring the statistics for
- **Class** - indicates the class of an operation as shows in Table 9.
- **Op** - The actual protocol operation name.

```
f800eth-1# isi statistics protocol list --protocols=nfs3,nfss4 --sort=TimeAvg --degraded
```

Ops	In	Out	TimeAvg	TimeStdDev	Node	Proto	Class	Op
319.7	46.0k	46.0k	18890.4	4753.0	1	nfs3	delete	rmdir
322.7	46.5k	46.5k	17596.2	3461.0	3	nfs3	delete	rmdir
321.8	46.3k	46.3k	17418.4	3412.2	2	nfs3	delete	rmdir
1.3k	192.9k	139.7k	16590.1	5795.6	1	nfs3	delete	remove
1.3k	186.9k	183.8k	15158.3	4761.4	2	nfs3	delete	remove
1.3k	192.8k	189.6k	15111.7	4651.0	3	nfs3	delete	remove
153.0	22.0k	22.0k	14257.6	1918.2	4	nfs3	delete	rmdir
320.5	52.6k	46.1k	13719.9	6680.5	1	nfs3	namespace_write	setattr
319.0	52.3k	45.9k	12552.8	6458.4	2	nfs3	namespace_write	setattr
316.2	51.9k	45.5k	12262.6	6143.8	3	nfs3	namespace_write	setattr
626.3	91.7k	90.2k	11840.7	4316.8	4	nfs3	delete	remove
152.0	24.9k	21.9k	9824.8	5590.4	4	nfs3	namespace_write	setattr
9.0k	347.2M	1.4M	7094.2	5316.4	1	nfs3	write	write
1.3k	234.3k	357.1k	6818.8	4466.1	1	nfs3	create	create
318.9	54.8k	86.7k	6739.3	4990.9	1	nfs3	create	mkdir
9.7k	419.6M	1.5M	6320.4	4450.9	2	nfs3	write	write
9.6k	400.7M	1.5M	6097.8	4170.2	3	nfs3	write	write
322.5	55.5k	87.7k	5504.8	2805.8	3	nfs3	create	mkdir
1.3k	227.6k	347.0k	5421.8	2863.3	2	nfs3	create	create
1.3k	234.4k	357.3k	5393.8	2586.4	3	nfs3	create	create
321.6	55.3k	87.5k	5271.5	2509.7	2	nfs3	create	mkdir
1.2k	178.2k	188.1k	4790.9	4332.1	1	nfs3	other	commit
1.2k	175.5k	185.2k	4183.3	3757.2	2	nfs3	other	commit
1.2k	177.4k	187.3k	3968.3	3075.9	3	nfs3	other	commit
4.9k	203.3M	781.0k	3551.6	3069.7	4	nfs3	write	write
8.3k	1.2M	404.4M	3381.3	3304.3	1	nfs3	read	read
624.8	111.5k	169.9k	2885.8	1510.9	4	nfs3	create	create
153.0	26.3k	41.6k	2812.8	1474.7	4	nfs3	create	mkdir
1.3k	192.5k	152.5k	2754.7	3278.3	1	nfs3	namespace_read	lookup
2.7k	362.5k	319.9k	2596.4	3047.1	1	nfs3	namespace_read	access
13.8k	1.8M	1.5M	2581.2	3184.9	1	nfs3	namespace_read	getattr
9.9k	1.4M	489.2M	2505.2	2335.8	2	nfs3	read	read
9.8k	1.4M	485.1M	2471.5	2100.6	3	nfs3	read	read
668.3	96.2k	101.6k	2454.2	2080.9	4	nfs3	other	commit
1.3k	192.5k	152.5k	1809.7	1959.8	3	nfs3	namespace_read	lookup
1.3k	186.9k	148.1k	1808.8	1904.2	2	nfs3	namespace_read	lookup
4.3k	588.3k	519.1k	1738.5	1961.3	3	nfs3	namespace_read	access
3.6k	488.0k	430.6k	1682.4	1808.2	2	nfs3	namespace_read	access
14.5k	1.9M	1.6M	1674.3	1889.8	2	nfs3	namespace_read	getattr
14.3k	1.9M	1.6M	1664.8	1910.3	3	nfs3	namespace_read	getattr
4.9k	710.2k	240.9M	888.0	1160.6	4	nfs3	read	read
624.4	91.4k	72.4k	167.4	222.6	4	nfs3	namespace_read	lookup
1.5k	199.2k	175.7k	133.7	227.2	4	nfs3	namespace_read	access
7.3k	961.9k	816.2k	83.7	229.0	4	nfs3	namespace_read	getattr

Total: 44

Figure 19 isi statistics protocol result for NFS

We can see the latency of the protocol to clarify if it is high. A high latency in the protocol maybe an indication of a potential problem in the cluster. As the high latency in the NFS protocol level would be caused by the problem of the lower level, such as network, disk, or file system that is holding up the protocol operations. The Table 11 shows the common expectations about the protocol latency times. You can convert the TimeAvg to milliseconds to compare with the expected time in Table 11.

Table 11 Common expected protocol latency time

Namespace metadata		Read	Write
< 10 ms	Good	Dependent on I/O size	Dependent on I/O size
10 ms ~ 20 ms	Normal		
> 20 ms	Bad		
> 50 ms	Investigate		

Isi statistics psstat

The sub-command `isi statistics psstat` output can help you analysis the approximate mix of read, write, and metadata operations. Figure 20 is an example output by running the following command with `--protocol=nfs3` option to get the NFSv3 statistics.

```
f800eth-1# isi statistics psstat --protocol=nfs3
```

NFS3 Operations Per Second					
access	12343.48/s	commit	4253.26/s	create	4185.22/s
fsinfo	0.00/s	getattr	46988.80/s	link	0.00/s
lookup	4179.97/s	mkdir	1036.72/s	mknod	0.00/s
noop	0.00/s	null	0.00/s	pathconf	0.00/s
read	33071.55/s	readdir	0.00/s	readdirplus	0.00/s
readlink	0.00/s	remove	4187.72/s	rename	0.00/s
rmdir	1034.97/s	setattr	1039.14/s	statfs	0.00/s
symlink	0.00/s	write	33992.27/s		
Total	146313.11/s				

CPU Utilization		OneFS Stats	
user	11.7%	In	1.17 GB/s
system	69.3%	Out	2.31 GB/s
idle	19.0%	Total	3.47 GB/s

Network Input		Network Output		Disk I/O	
MB/s	1283.65	MB/s	1456.99	Disk	305321.80 iops
Pkt/s	243751.60	Pkt/s	325552.77	Read	1.62 GB/s
Errors/s	23.77	Errors/s	0.00	Write	4.12 GB/s

Figure 20 isi statistics psstat result for NFS

You can find each operation rates for NFSv3, in this example, subtract the 33071 ops/s for read and 33992 ops/s for write from the total 146313 ops/s, which leaves 79250 ops/s for metadata operations. So the NFSv3 read, write and metadata ratio approximately: read 22.6%, write 23.2%, metadata 54.2%.

8.2 Packet capture tool and analysis

It is useful to capture NFS packet during the troubleshooting. You can figure out the communication details between the server and client. On Linux environment, you can use `tcpdump` tool to capture the traffic on the server or client. For the usage of `tcpdump`, you can refer to the man page of [tcpdump](#). For OneFS cluster, it is more convenient to use `isi_netlogger` command tool to capture the traffic in the cluster on more than one node. You can get the usage through the help output using `isi_netlogger -h`. Below is an example scenario about how to use these tools to analysis the NFS traffic and get the information as needed.

Assuming that an application has an issue to read/write a file on the cluster, we need to verify and find out if the actual file in the network traffic is same as the file that the application reported. So that we need to find the filename according to the filehandle in the captured packet. We use `isi_netlogger` to capture the packet and use Wireshark to analyze the .pcap file.

1. Running the following command captures the packets on the interface mlxen2 (IP 172.16.200.41) of OneFS cluster node number 1. And only capture the traffic to and from the application host 172.16.200.160.

```
isi_netlogger -l mlxen2 -n 1 host 172.16.200.160
```

2. The captured file will be stored at `/ifs/netlog` by OneFS. You can download the file to your local machine, and open it using Wireshark which is a feature-rich network packet analysis tool. Find the operation which accesses the file on OneFS cluster, as shows in Figure 21.

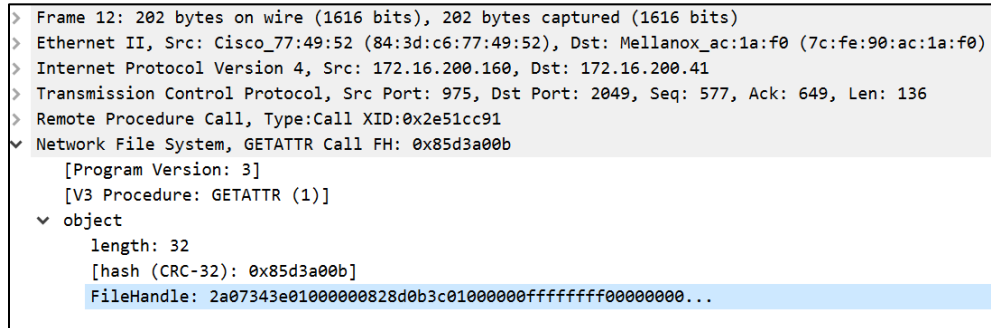


Figure 21 The NFSv3 operation filehandle

Filehandle in OneFS NFSv3 is a 32 bytes structure contains the following parts:

- File system ID (4 bytes): the exported file system identifier.
- Export (4 bytes): the unique export identifier.
- File ID (8 bytes): the file's logical inode number (LIN).
- Snap ID (4 bytes): the snapshot identifier of the root of the mount.
- Portal (4 bytes): the snapshot portal depth.
- Root LIN (8 bytes): the LIN of the root of the export.

Figure 22 is the filehandle broken into each section, the LIN in the packet capture is represented in little-endian format rather than a [big-endian format](#). So the actual file's LIN in OneFS is 00:00:00:01:3c:0b:8d:82 (13c0b8d82).

2a:07:34:3e:01:00:00:00:82:8d:0b:3c:01:00:00:00:ff:ff:ff:ff:00:00:00:00:02:00:00:00:00:00:00:00
fsid export lin snapid portal_depth root_lin

Figure 22 NFSv3 filehandle in OneFS

3. Find the file path on the OneFS cluster for the obtained LIN using command `isi get -L 13c0b8d82`, the output is similar to the following.

```
# isi get -L 13c0b8d82
A valid path for LIN 0x13c0b8d82 is /ifs/test.txt
```

A Technical support and resources

[Dell.com/support](https://dell.com/support) is focused on meeting customer needs with proven services and support.

A.1 Related resources

- [PowerScale Info Hub](#)
- [Dell PowerScale OneFS: A Technical Overview](#)
- [PowerScale OneFS Web Administration Guide](#)
- [PowerScale OneFS CLI Administration Guide](#)
- [Current PowerScale Software Releases](#)
- [OneFS Security Configuration Guide](#)