Technical White Paper

Dell EMC PowerScale OneFS S3 Overview

Abstract

This document provides technical details about the Dell EMC™ PowerScale™ OneFS™ S3 implementation, including bucket and object operations, authentication, and the authorization process. It also introduces the benefits of OneFS S3 and provides applicable use cases.

June 2020

H18292

Revisions

Date	Description
June 2020	Initial release

Acknowledgments

Author: Lieven Lin

Table of contents

Revisions....2
Acknowledgments....2
Table of contents....3
Executive summary....4
1 Introduction....5
1.1 OneFS S3 overview....5
2 Use cases....8
3 OneFS S3 implementation....9
3.1 Buckets....10
3.1.1 Bucket naming rules....10
3.1.2 Bucket operations....11
3.2 Objects....11
3.2.1 Multipart upload....12
3.2.2 Object operations....12
4 OneFS S3 authentication and authorization....14
4.1 OneFS S3 request authentication....14
4.2 OneFS S3 request authorization....15
4.3 ACL....16
4.3.1 Bucket ACL....16
4.3.2 Object ACL....16
4.3.3 Canned ACL....18
4.4 S3 predefined groups....18
A Technical support and resources....19
A.1 Related resources....19

Executive summary

Dell EMC™ PowerScale™ OneFS™ supports the native S3 protocol. With this capability, clients can access OneFS cluster file-based data as objects. PowerScale combines the benefits of traditional NAS storage and emerging object storage to provide an enhanced data-lake capability and cost-effective unstructured data storage solution. OneFS S3 is designed as the first-class protocol including features for bucket and object operations, security implementation, and management interface.

This document introduces how the S3 API is implemented in OneFS to provide high-performance data access. It introduces the benefits of OneFS S3 and provides applicable use cases. This document also details bucket and object operations, authentication, and the authorization process.

1 Introduction

Data is now a new form of capital. It provides the insights that facilitate your organization digital transformation, and 80% of the information is represented as unstructured data. Organizations in every industry generate an exponentially larger number of unstructured data volumes than ever before, from edge to core to cloud. The way of storing and managing unstructured data is evolving. Its goal is to unlock the value of your data by using both the traditional network attached storage (NAS) system and emerging object storage.

Many organizations run their critical applications on traditional NAS storage, and develop new modern applications using object storage. There are some challenges under the heterogeneous storage platforms for unstructured data:

- Applications running on different storage platforms may need to access a same set of data. In this case, data migration is required between NAS storage and object storage, and the extra copy of the data consumes additional storage capacity.
- There is an inability to access object storage through the NAS protocol, like NFS and SMB. Many object-storage systems provide access to the NAS protocol using a gateway-like architecture which does not perform adequately when combined with the object storage stack.
- In contrast to NAS storage, object storage is not intended for transactional data where operations-per-second and latency are critical.

Designed to address these challenges, PowerScale is ideal for demanding enterprise file workloads and can store, manage, and protect unstructured data with efficiency and massive scalability. It supports multiple NAS protocols natively for applications. Starting with OneFS version 9.0, it provides the capability of accessing data through the Amazon Simple Storage Service (Amazon S3) application programing interface (API) natively. PowerScale implements the S3 API as the first-class protocol along with other NAS protocols on top of its distributed OneFS file system. Whether your application is based on traditional NAS storage or the emerging object storage, the application can access data in a single scale-out storage platform through NAS protocols or the S3 API as needed. This document introduces how S3 API is implemented in OneFS and can provide high-performance data access.

1.1 OneFS S3 overview

The Amazon S3 API was originally developed as the data-access interface of Amazon S3. As applications were developed using the S3 API, it became a common standard for object storage. This document refers to the S3 API for object storage as the S3 protocol. This provides a consistent nomenclature along with other NAS protocols regarding the OneFS file service.

Figure 1 shows the traditional scale-up NAS platform and the emerging object-storage architecture. The traditional scale-up NAS platform is only accessible through file protocols and is not easy to scale as the performance requirement increases. The object storage allows both file and object access, but the file access is achieved through a gateway, with either a software daemon or additional dedicated hardware. This limits file-access performance compared to a traditional NAS platform.

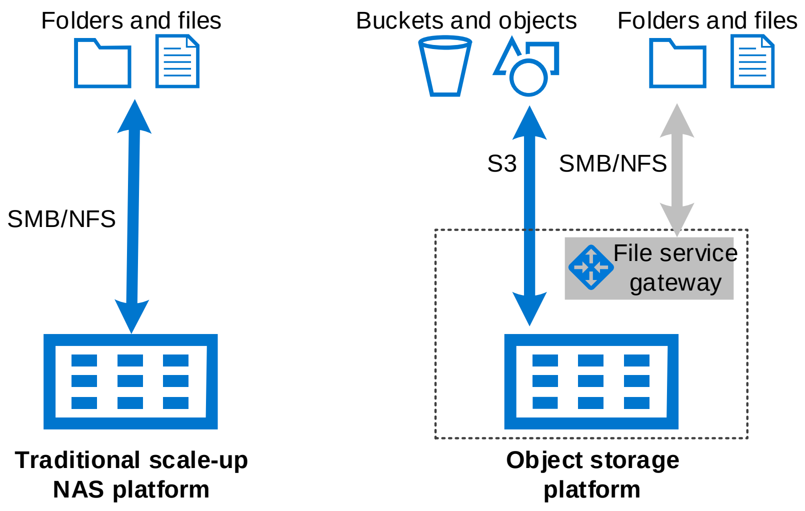


Figure 1 Scale-up NAS platform and object storage platform architecture

This is where PowerScale scale-out storage comes in. With the introduction of the OneFS S3 protocol, PowerScale combines the advantages of both platform types into a single storage system while providing performance for file and object access.

Starting with OneFS version 9.0, PowerScale OneFS supports the native S3 protocol. OneFS implements S3 as a first-class protocol along with other protocols, including NFS, SMB, and HDFS. The S3 protocol is implemented over HTTP and secure HTTP (HTTPS). Through OneFS S3, you can access file-based data stored on your OneFS cluster as objects. Since the S3 API is considered to be a protocol, content and metadata can be ingested using S3 and concurrently accessed through other protocols that are configured on the cluster.

Note: The OneFS S3 service is disabled by default. If the service is enabled, it only listens on port 9021 for HTTPS. Port 9020 for HTTP is disabled by default. These ports are configurable through the OneFS WebUI and CLI.

In OneFS 9.0, each S3 bucket is mapped to a directory under an access zone base path, each S3 object is mapped to a file, and the associated object prefix is mapped to directories. A directory for a bucket is created under the access zone base path by default. See the AWS S3 documentation regarding S3 bucket and object definition.

OneFS 9.0 provides the following new S3 features:

- Create, list, update, and delete buckets
- Create, list, read, and delete objects
- Support for both AWS Signature Version 2 and Version 4
- Support both path-style requests and virtual hosted-style requests
- Multipart upload of large content for better performance
- Access ID and secret key management through WebUI, CLI, and PAPI
- Bucket ACL and Object ACL for access control
- Access zone awareness for multitenancy

2 Use cases

By implementing the S3 protocol, OneFS enhances its data-lake capability by supporting both the traditional NAS protocol and object storage protocol. You can unify your file and object data access in a single storage namespace. The S3 protocol on a OneFS cluster provides following benefits:

- Consolidate storage for applications regardless of data-access protocols
- Store data with the S3 protocol and then seamlessly access the data as files with SMB, NFS, HTTPS, FTP, and HDFS
- Store files with SMB, NFS, and other protocols and then access the files as objects through the S3 protocol
- Eliminate data migration when a same set of data is accessed through NAS protocols and the S3 protocol
- Multitenancy support for better storage-as-a-service abilities through S3
- Increased return on investment for the OneFS cluster by supporting object access

As the Figure 2 shows, OneFS seamless interoperates between file-based and object-based data access in a single NAS platform for various workloads.

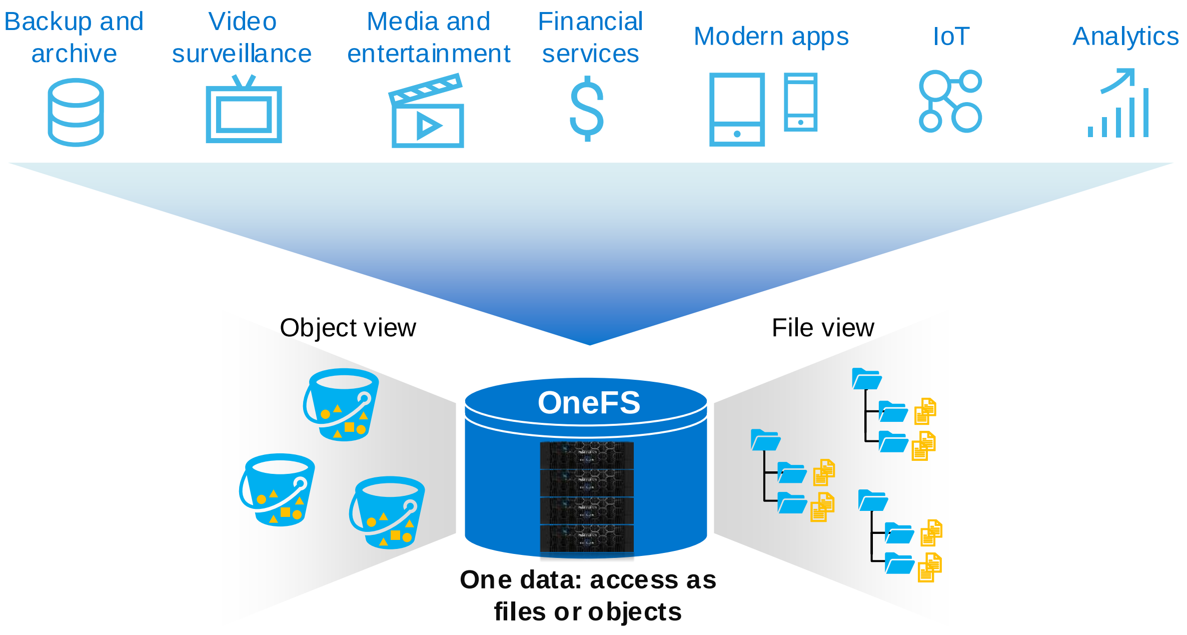


Figure 2 Access data as file and object in a single storage

OneFS can now extend its use cases with the benefits of OneFS S3. The following list includes general use cases for OneFS S3:

- **Backup and archive:** It is possible to make OneFS as an ideal target for S3-compatible data backup and archive software.
- **File service**: Data access for files and data access for objects are easily consolidated in a single scale-out storage. This provides faster service than cloud and more cost-effective service than traditional NAS platforms.
- **Enhanced multiprotocol data access:** The data in a OneFS cluster can be accessed as files or objects.

3 OneFS S3 implementation

OneFS implements the S3 protocol on top of the file-service engine like other protocols. Clients that connect to a OneFS cluster with S3 gain access to the single volume of the distributed OneFS file system and take advantage of the entire cluster's performance. To work with OneFS S3, clients connect to the S3 service over HTTP or HTTPS and use standard REST calls such as PUT, GET, and POST to perform bucket and object operations.

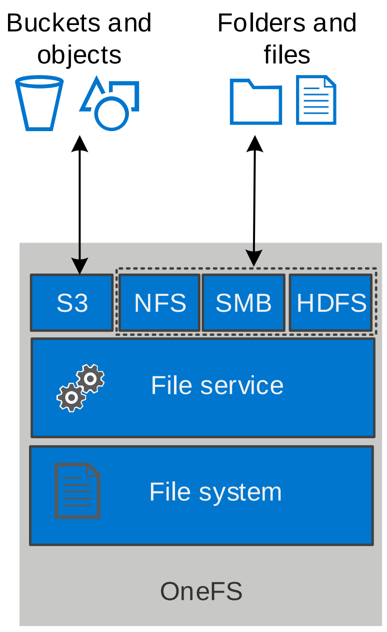


Figure 3 OneFS S3 architecture overview

Making an analogy with an SMB share which is associated with a path, a OneFS S3 bucket is also created based on a specific path within the access zone base path. OneFS S3 maps an object to a file and maps the object prefix to directories correspondingly. For example, assume a file is stored in OneFS with a full path of **/ifs/data/docs/finance/sample.pdf**. To access the file with S3, create a bucket **bkt01** in OneFS and associate the bucket with a path **/ifs/data/docs/**. The object key of **/finance/sample.pdf** is used to represent the file.

OneFS support two types of requests when resolving buckets and objects. See the Amazon S3 documentation Virtual Hosting of Buckets for more details about the following:

- Virtual hosted-style requests: Specify a bucket in a request by using the HTTP Host header
- Path-style requests: Specify a bucket by using the first slash-delimited component of the Request-URI path

With OneFS S3, you can access the file as an object by using GET and PUT operations with the following URLs (for example) which contain the SmartConnect zone name:

- Path-style requests: **https://sc.example.local:9021/bkt01/finance/sample.pdf**
- Virtual hosted-style requests: **https://bkt01.sc.example.local:9021/finance/sample.pdf**

The path-style request is available through both the SmartConnect zone name and IP address of a node.

To use virtual hosted-style request, the following configuration is required:

- A SmartConnect zone name is required.
- The wildcard subdomain option must be enabled for the groupnet. This option is enabled by default.

```
# isi network groupnets modify <groupnet> --allow-wildcard-subdomains=true
```

- Configure the S3 base domain to your SmartConnect zone name using the WebUI or CLI.

```
# isi s3 settings zone modify --base-domain=<smartconnect> --zone=<name>
```

3.1 Buckets

OneFS requires a bucket to map to a specific directory in an access zone. This directory is called the bucket path. If the bucket path is not specified, a default path is used, which is configurable at an access zone level through the WebUI or CLI. When creating a bucket, OneFS creates a directory with a prefix of **.isi_s3_** under the bucket path, and creates 16 other subdirectories named 0 through 15 under the .isi_s3_ directory. An example of this name is **.isi_s3_ 1_1000000010001**. The 16 subdirectories are used to store temporary files for the PUT operation. OneFS automatically balances different temporary files between the directories for better performance. Figure 4 shows the process of putting an object to the OneFS cluster which uses the temporary directory under the bucket.

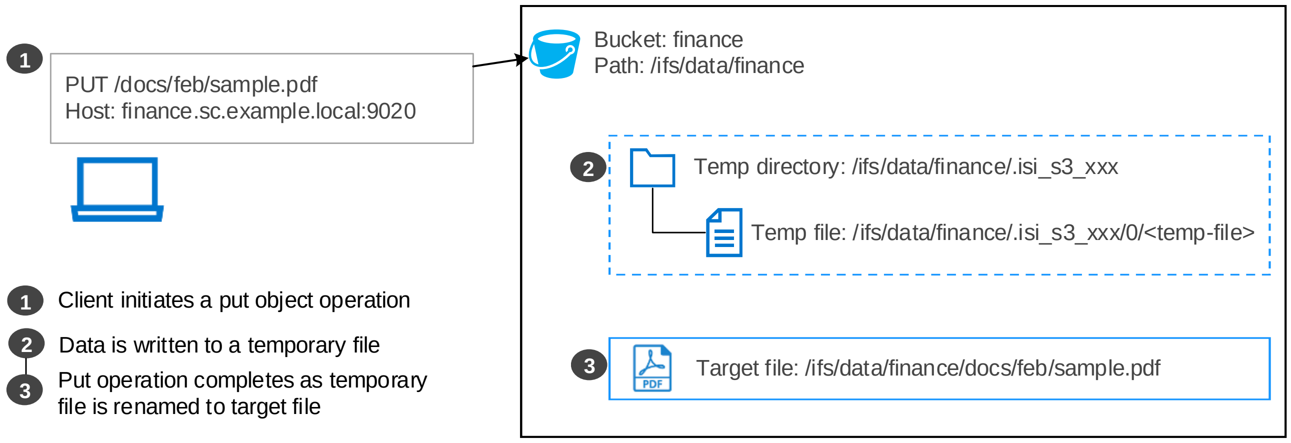

1
PUT /docs/feb/sample.pdf
Host: finance.sc.example.local:9020
Bucket: finance
Path: /ifs/data/finance
2
Temp directory: /ifs/data/finance/.isi_s3_xxx
Temp file: /ifs/data/finance/.isi_s3_xxx/0/<temp-file>
3
Target file: /ifs/data/finance/docs/feb/sample.pdf
1 Client initiates a put object operation
2 Data is written to a temporary file
3 Put operation completes as temporary file is renamed to target file

Figure 4 OneFS S3 put object

3.1.1 Bucket naming rules

OneFS S3 bucket names comply with DNS naming conventions. The following rules are required for naming S3 buckets in OneFS:

- Bucket names must be unique at the OneFS access zone level.
- Bucket names cannot be changed after the bucket is created.
- Bucket names must consist of characters including lowercase letters (a-z), numbers (0-9), or dashes (-).
- Bucket names must start or end with a lowercase letter (a-z) or number (0-9).
- Bucket names must be 3 to 63 characters in length.

3.1.2 Bucket operations

Table 1 shows the supported S3 bucket operations in OneFS 9.0. See the document *Dell EMC PowerScale: OneFS S3 API Guide* on Dell.com/StorageResources for details about each supported API.

Table 1 OneFS S3 bucket operations

API name in AWS S3 API reference	Description
CreateBucket	PUT operation to create a bucket. Anonymous requests are never allowed to create buckets. By creating the bucket, the authenticated user becomes the bucket owner.
ListObjects	List objects in a bucket.
ListObjectsV2	List objects in a bucket.
GetBucketLocation	Returns the location as an empty string.
DeleteBucket	Delete the bucket.
GetBucketAcl	Get the access control list (ACL) of a bucket.
PutBucketAcl	Set the permissions on an existing bucket using ACLs.
HeadBucket	Determine if a bucket exists and if you have permission to access it. The operation returns a **200 OK** if the bucket exists and if you have permissions to access it. Otherwise, the operation might return responses such as 404 Not Found and 403 Forbidden.
ListBuckets	Get a list of all buckets owned by the authenticated user of the request.
ListMultipartUploads	List in-progress multipart uploads. An in-progress multipart upload is a multipart upload that has been initiated using the Initiate Multipart Upload request but has not yet been completed or aborted.

3.2 Objects

The AWS S3 data model is a flat structure—you create a bucket, and the bucket stores objects. There is no hierarchy of subbuckets or subfolders. However, AWS S3 provides a logical hierarchy using object key name prefixes and delimiters to support a concept of folders. For example, instead of naming an object **sample.jpg** in the bucket named **examplebucket**, you can name it **photos/samples/sample.jpg**. The **photos/samples/** becomes the object key name prefix using the slash (/) as delimiter.

OneFS maps objects to files and creates directories as object key name prefixes implicitly. The OneFS file system requires the following rules for object naming :

- The object can only use ASCII or UTF-8 characters.
- Only the slash (/) is supported as an object-key-name delimiter. The slash cannot be part of the object key name; it is automatically treated as a delimiter.
- Objects cannot contain a prefix that conflicts with an existing object key name in its path. For example, creating the object **/document** and **/document/sample.docx** is not allowed within the same bucket.
- You cannot use a period (.) or double period (..) as object key name or part of object key name prefix. For example, you can create the object **/.sample.jpg** but not the object **/./sample.jpg**.
- You cannot use **.snapshot** as the object key name or part of an object key name prefix; this is reserved for OneFS SnapshotIQ.

3.2.1 Multipart upload

The multipart upload allows users to upload new large files or make a copy of an existing file in parts for better uploading performance. After receiving the CreateMultipartUpload request from client, OneFS processes the multipart upload by writing each part to a temporary directory. It completes the multipart upload after all the parts are uploaded successfully.

Parts are uploaded to the temporary directory **.isi_s3_parts_UploadId**, and the temporary directory is created under the target directory. Upon receiving the complete multipart-upload request, OneFS concatenates the temporary files to the target file. You can access the file as object in your bucket. Figure 5 shows the details about the temporary directory and files during a multipart-upload operation.

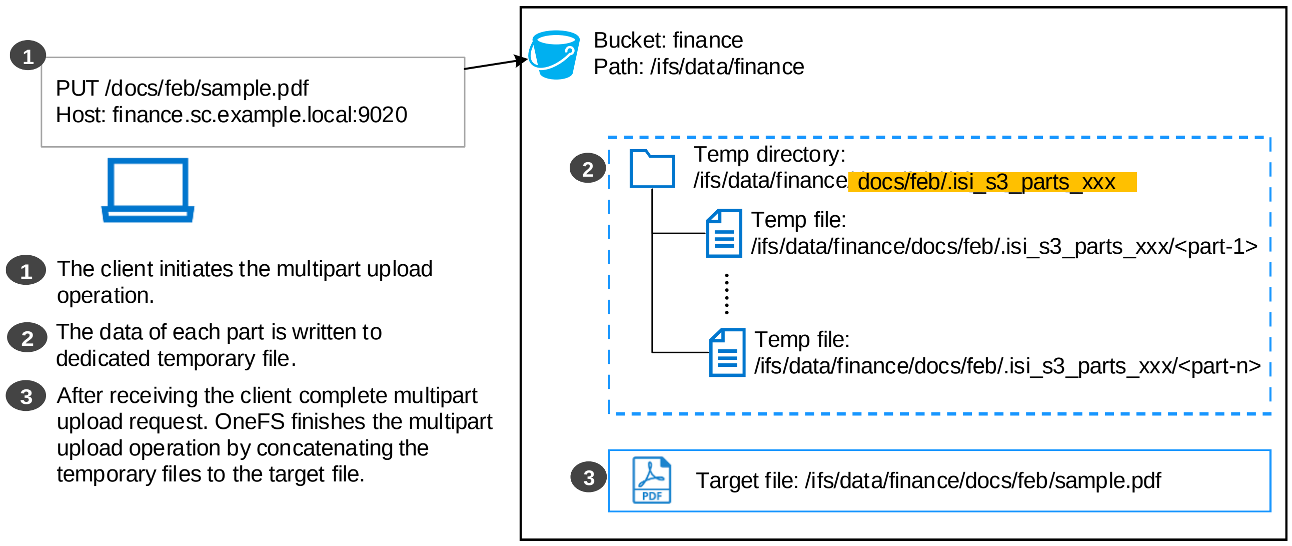

1
PUT /docs/feb/sample.pdf
Host: finance.sc.example.local:9020
Bucket: finance
Path: /ifs/data/finance
2
Temp directory:
/ifs/data/finance/docs/feb/.isi_s3_parts_xxx
Temp file:
/ifs/data/finance/docs/feb/.isi_s3_parts_xxx/<part-1>
Temp file:
/ifs/data/finance/docs/feb/.isi_s3_parts_xxx/<part-n>
3
Target file: /ifs/data/finance/docs/feb/sample.pdf
1 The client initiates the multipart upload operation.
2 The data of each part is written to dedicated temporary file.
3 After receiving the client complete multipart upload request. OneFS finishes the multipart upload operation by concatenating the temporary files to the target file.

Figure 5 OneFS S3 multipart upload process

3.2.2 Object operations

Table 2 shows the supported S3 object operations in OneFS 9.0. See the document *Dell EMC PowerScale: OneFS S3 API Guide* on Dell.com/StorageResources for details about each supported API.

Table 2 OneFS S3 object operations

API name in AWS S3 API reference	**Note**
DeleteObject	Delete a single object from a bucket. Deleting multiple objects from a bucket using a single request is not supported.
GetObject	Retrieve an object content.
GetObjectAcl	Get the ACL of an object.
HeadObject	HEAD operation retrieves metadata from an object without returning the object itself. This operation is useful if you are only interested in an object's metadata. The operation returns a **200 OK** if the object exists and if you have permission to access it. Otherwise, the operation might return responses such as 404 Not Found and 403 Forbidden.
PutObject	Add an object to a bucket.

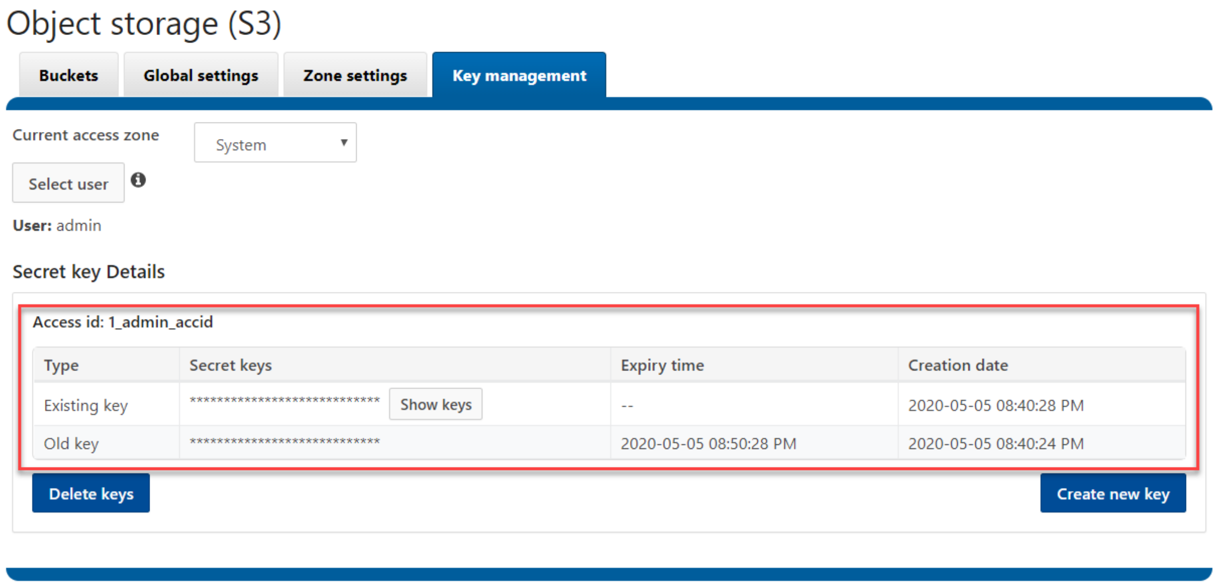
API name in AWS S3 API reference	**Note**
CopyObject	Create a copy of an object that is already stored in OneFS. You can treat it as server-side-copy which reduces the network traffic between the clients and OneFS.
PutObjectAcl	Set the ACL permissions for an object that exists in a bucket.
CreateMultipartUpload	Initiate a multipart upload and return an upload ID. This upload ID is used to associate with all the parts in the specific multipart upload. You can specify this upload ID in each of your subsequent upload part requests. You also include this upload ID in the final request to either complete or abort the multipart upload request.
UploadPart	Upload a part in a multipart upload. Each part must be at least 5 MiB, except the last part. The maximum size of each part is 5 GiB.
UploadPartCopy	Upload a part by copying data from an existing object in OneFS as the data source. Each part must be at least 5 MiB in size, except the last part. The maximum size of each part is 5 GiB.
CompleteMultipartUpload	Complete a multipart upload by assembling previously uploaded parts.
ListParts	List the parts that have been uploaded for a specific multipart upload.
AbortMultipartUpload	Abort a multipart upload. After a multipart upload is aborted, no additional parts can be uploaded using that upload ID. The storage consumed by any previously uploaded parts is freed. However, if any uploads of parts are in progress, those uploads might or might not succeed. As a result, it might be necessary to abort a given multipart upload multiple times to free all storage consumed by all parts.

4 OneFS S3 authentication and authorization

The S3 protocol enables accessing OneFS file-system data as objects. Every interaction with OneFS S3 is either authenticated or anonymous. Authentication verifies the identity of the requester trying to access OneFS data. After OneFS authenticates the requesting user or anonymous user, authorization is required to control which permissions the requester has for the required data. This section introduces how OneFS authenticates and authorizes each S3 request.

4.1 OneFS S3 request authentication

OneFS S3 authenticates requests with a signing mechanism, either AWS Signature Version 2 or AWS Signature Version 4. OneFS does not support the chuncked upload process. Authenticated requests must include a signature value that authenticates the request sender. Each of these requires the user to have an access ID and a secret key. The user must get these credentials through the OneFS WebUI or CLI before performing signed S3 requests. Figure 6 shows a result of generating the access ID and secret key for the admin user through the WebUI.



Current access zone: System

User: admin

Secret key Details

Access id: 1_admin_accid

Type	Secret keys	Expiry time	Creation date
Existing key	***************************	--	2020-05-05 08:40:28 PM
Old key	***************************	2020-05-05 08:50:28 PM	2020-05-05 08:40:24 PM

Figure 6 Generate access ID and secret key using the WebUI

The access ID indicates who the user is. OneFS generates one access ID per user. For example, OneFS may generate access ID **1_joe_accid** for user **joe**. The prefix number represents the user's access zone ID. Each access ID would have a latest secret key without an expiry time set and an old secret key that has an expiry time set.

The secret key is used to generate the signature value along with several request-header values. After receiving the signed request, OneFS uses the access ID to retrieve a copy of the secret key internally, recompute the signature value of the request, and compare it against the received signature. If they match, the requester is authenticated, and any header value that was used in the signature is now verified to be untampered as well.

OneFS treats unauthenticated requests as anonymous requests made by the **nobody** user (UID 65534). If an object is uploaded to a bucket through an unauthenticated request, the **nobody** user owns the object and has

FULL_CONTROL to the object. To prevent an anonymous request from modifying OneFS buckets and objects, we recommend not to use ACLs that give the **nobody** user write access to your bucket.

For example, use the following OneFS CLI to grant or revoke write permissions on the bucket for the **nobody** user.

```
# isi s3 buckets modify <bucket> --add-ace="name=nobody,type=user,perm=write" --
zone=<name>

# isi s3 buckets modify <bucket> --remove-ace="name=nobody,type=user,perm=write"
--zone=<name>
```

4.2 OneFS S3 request authorization

OneFS supports the bucket ACL to control whether a user has permission for a bucket. When receiving a S3 request for a bucket operation, OneFS parses the user access ID from request header and evaluates the request according to the target bucket ACL. Figure 7 shows the authentication and authorization-evaluation-for-bucket operation.

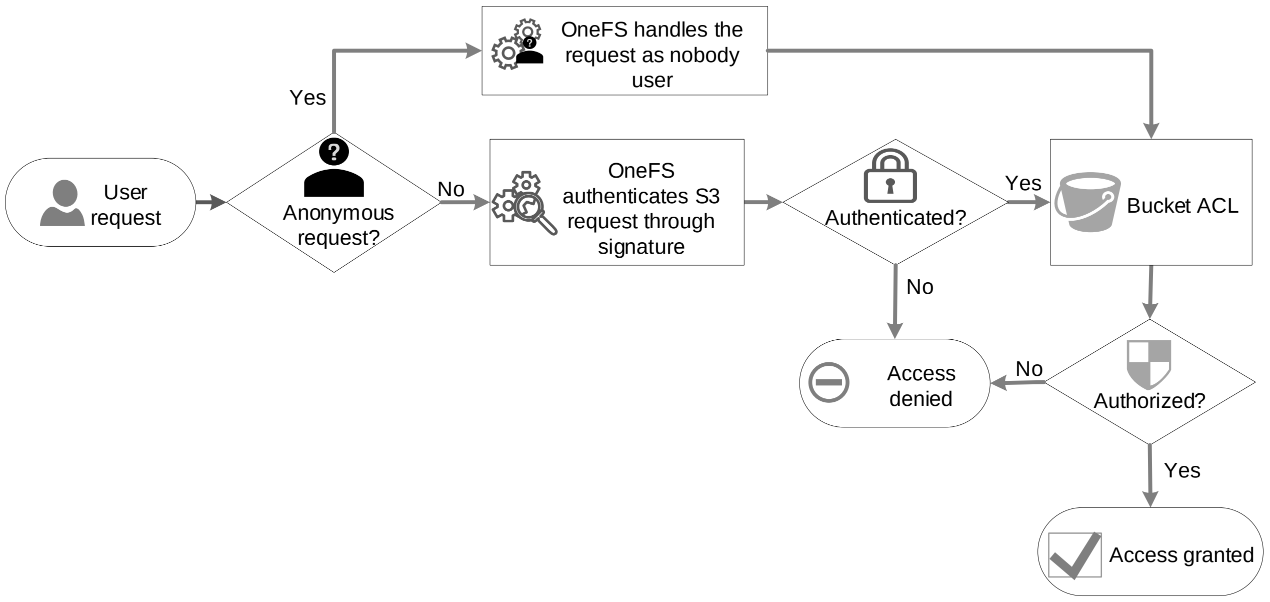


Figure 7 Authentication and authorization for bucket operations

To access OneFS S3 objects, the S3 request must be authorized in both the bucket and object level. When receiving an S3 request for an object operation, OneFS parses the username from the request header. If the request is to PUT or DELETE an object, OneFS evaluates the request according to the target bucket ACL. If authorized, OneFS evaluates the request against the OneFS file system ACL. Otherwise, OneFS evaluates the request against OneFS ACL directly. The Object ACL is derived from OneFS file system ACL and for representation only. See section 4.3.2 for more details about object the ACL and OneFS ACL.

Figure 8 shows the authentication and authorization evaluation for the object operation.

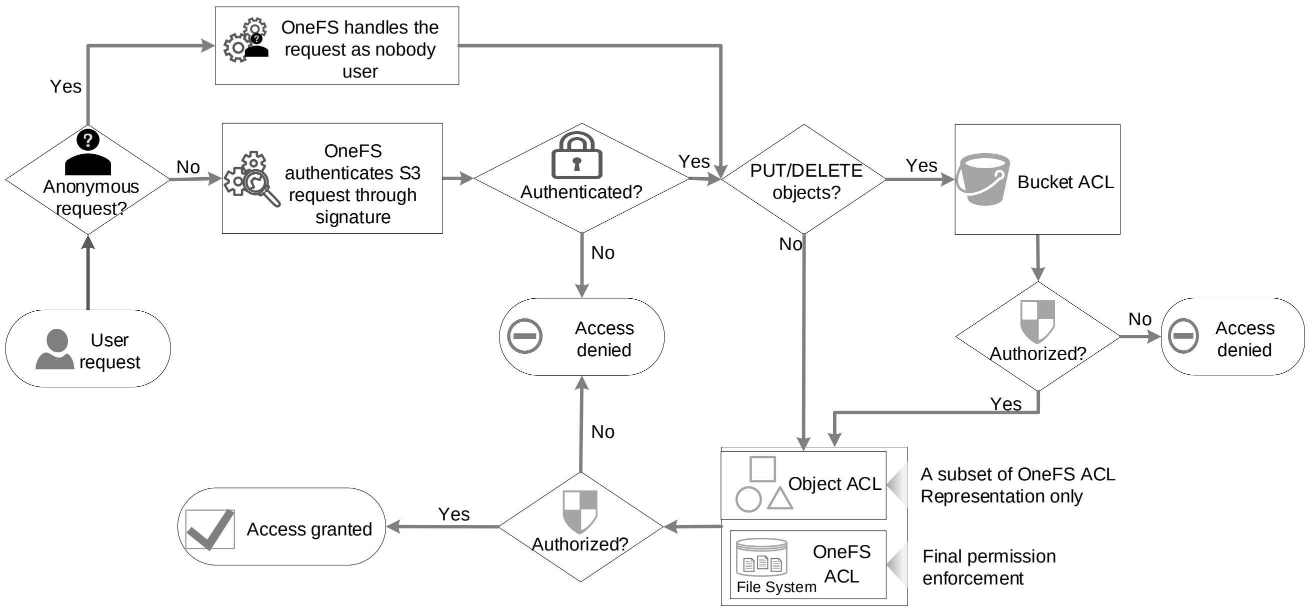


Figure 8 Authentication and authorization for object operations

4.3 ACL

OneFS implements both the bucket ACL and object ACL to control user access. Every bucket and object has an associated ACL. The ACL defines which OneFS users or groups are granted access and the type of access.

4.3.1 Bucket ACL

The following list includes the set of permissions that OneFS S3 supports in a bucket ACL. The bucket owner always has FULL_CONTROL on the bucket.

- **READ:** Allows users to list the objects in the bucket.
- **WRITE:** Allows users to create, overwrite, and delete objects in the bucket.
- **READ_ACP:** Allows users to read the bucket ACL.
- **WRITE_ACP:** Allows users to write the ACL for the applicable bucket.
- **FULL_CONTROL:** Allows users the READ, WRITE, READ_ACP, and WRITE_ACP permissions on the bucket.

4.3.2 Object ACL

The following list includes the set of permissions that OneFS S3 supports in an object ACL.

- **READ:** Allows users to read the object data and its metadata.
- **WRITE:** Not applicable but can be set on an object. When updating an object, S3 overwrites the entire object instead of modifying existing data. Overwriting an object requires the WRITE permission at the bucket level. It also requires the WRITE permission on the directory containing the object at the OneFS level, where the directory is the largest prefix of the object (for example, if the object is /a/b/c.txt, the directory is /a/b/). When this permission is set on an object, OneFS assigns the corresponding OneFS ACL permission on disk so that the object data can be modified through other protocols.

- **READ_ACP:** Allows users to read the object ACL.
- **WRITE_ACP:** Allows users to write the ACL for the applicable object.
- **FULL_CONTROL:** Allows grantee the READ, WRITE, READ_ACP, and WRITE_ACP permissions on the object. Also includes additional execute and std_write_owner OneFS permissions to provide consistent multiprotocol access.

OneFS always handles permission enforcement based on the OneFS ACL. The object ACL is for representation only. The object ACL only defines ALLOW ACL, whereas OneFS ACL also defines DENY ACL. When viewing the object ACL through S3, it is possible that only a subset of the actual ACL is shown.

To reconcile the S3 object ACL and OneFS ACL, OneFS provides the following object ACL policy along with the mapping relationship shown as Table 3:

- **Replace:** When S3 modifies a OneFS ACL, OneFS removes the existing ACL (including both ALLOW and DENY settings) and replaces it with the new S3 ACL. This is the default setting.
- **Deny:** OneFS ACL modification by S3 is not allowed. The S3 client receives a permission-denied error when trying to modify ACL.

Use the following command to modify object ACL policy on a bucket:

```
# isi s3 buckets modify <bucket> --object-acl-policy=<replace/deny> --
zone=<name>
```

The object ACL permissions are mapped to the following OneFS ACL permissions respectively in Table 3. See the white paper *Access Control Lists on Dell EMC PowerScale OneFS on* Dell.com/StorageResources for more details about OneFS ACL permissions.

Table 3 Object ACL and OneFS ACL mapping

Object ACL permission	OneFS ACL permission
READ	file_read, file_read_attr, file_read_ext_attr, std_synchronize
WRITE	file_write, append, file_write_attr, file_write_ext_attr, std_synchronize
READ_ACP	std_read_dac
WRITE ACP	std_write_dac
FULL_CONTROL	file_gen_all (this permission includes execute, std_write_owner permission, and all the above permissions)

4.3.3 Canned ACL

S3 has a set of predefined ACLs, known as canned ACLs. Each canned ACL has predefined grantees and permissions. Table 4 lists the canned ACLs supported by OneFS, and the associated predefined grants.

Table 4 OneFS S3 supported canned ACL

Canned ACL	Applies to	Grantees and permissions added to ACL
private	Bucket and object	Owner gets FULL_CONTROL. No one else has access rights (default).
public-read	Bucket and object	Owner gets FULL_CONTROL. The AllUsers group gets READ access.
public-read-write	Bucket and object	Owner gets FULL_CONTROL. The AllUsers group gets READ and WRITE access. We do not recommend granting this on a bucket.
authenticated-read	Bucket and object	Owner gets FULL_CONTROL. The AuthenticatedUsers group gets READ access.
bucket-owner-read	Object	Object owner gets FULL_CONTROL. Bucket owner gets READ access.
bucket-owner-full-control	Object	Both the object owner and the bucket owner get FULL_CONTROL over the object.

4.4 S3 predefined groups

S3 has the following predefined groups in Table 5. OneFS maps some groups with OneFS predefined groups.

Table 5 S3 predefined groups

S3 predefined Group	OneFS predefined Group	Note
Authenticated Users group	Authenticated Users	Only authenticated request is allowed in this group
All Users group	Everyone	Both authenticated request and unauthenticated request are allowed in this group
Log Delivery group	Not supported	OneFS does not support this group

A Technical support and resources

Dell.com/support is focused on meeting customer needs with proven services and support.

Storage technical documents and videos provide expertise that helps to ensure customer success on Dell Technologies storage platforms.

A.1 Related resources

See the following resources on Dell.com/StorageResources:

- Access Control Lists on Dell EMC PowerScale OneFS
- Dell EMC PowerScale OneFS S3 API Guide
- Dell EMC PowerScale OneFS: A Technical Overview

Also see the following Amazon documentation, Amazon S3 API Reference.