vCloud Director Object Storage Extension API Programming Guide

VMware vCloud Director Object Storage Extension 1.0

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

About the vCloud Director Object Storage API Programming Guide

The *Programming Guide* provides information about the vCloud Director Object Storage Extension REST APIs, including how to use the API services and resources, how to authenticate and construct REST API calls.

Intended Audience

This guide is intended for administrators and programmers who want to configure and manage vCloud Director Object Storage Extension programmatically using the vCloud Director Object Storage Extension REST API.

The guide focuses on common use cases. For information about all available REST API services, see the *vCloud Director Object Storage Extension API Reference* at https://code.vmware.com/apis/665/vose.

Introduction to the vCloud Director Object Storage Extension REST API

1

vCloud Director Object Storage Extension is a standalone middleware service integrated with vCloud Director that provides a set of APIs that are compatible with the Amazon Web Services Simple Storage Service (AWS S3) API.

For that reason, developing automation solutions and client applications for vCloud Director Object Storage Extension is the same as developing automation solutions and client applications for AWS S3. In addition to the current guide, when designing or developing S3 applications for vCloud Director Object Storage Extension, you can use the rich collection of AWS S3 resources for developers.

AWS S3 resources include getting started and developer guides, a code sample catalog, and software development kits (SDKs) for various development technologies. The following list contains useful references to AWS S3 development resources that can help you develop S3 applications for vCloud Director Object Storage Extension:

- AWS S3 Developer Guide
- AWS S3 API Reference
- AWS Code Sample Catalog
- AWS Developer Center
- AWS SDK for Java
- AWS SDK for Python (Boto3)
- AWS SDK for Go

Getting Started with vCloud Director Object Storage Extension REST API

2

The vCloud Director Object Storage Extension API uses RESTful application development style to provide bucket and object management capabilities to developers.

To make vCloud Director Object Storage Extension REST API service calls, you can use a browser application or an HTTP client to send requests and review responses.

Any client application that can send HTTP requests is an appropriate tool for developing REST applications with the vCloud Director Object Storage Extension S3 API.

Following is a list of tools tested during the development of vCloud Director Object Storage Extension S3 API:

- cURL command-line tool, available at https://curl.haxx.se/.
- Postman application, available at https://www.getpostman.com/.
- AWS Java SDK, available at https://aws.amazon.com/sdk-for-java/.
- AWS Python SDK, available at https://aws.amazon.com/sdk-for-python/.
- AWS Go SDK, available at https://aws.amazon.com/sdk-for-go/.

This chapter includes the following topics:

vCloud Director Object Storage Extension Concepts

The vCloud Director Object Storage Extension API adopts the main concepts used in the AWS S3 API. These concepts are referenced throughout the current document. Understanding these concepts is necessary to work effectively with the vCloud Director Object Storage Extension S3 API.

Bucket

Before you start uploading files to vCloud Director Object Storage Extension, you must create a bucket. You can then upload any number of files to the bucket. A bucket is a logical unit of storage. Buckets are the fundamental containers in vCloud Director Object Storage Extension.

With access control lists, you control the access permissions for buckets.

To aid organize and categorize your buckets, you can add multiple key-value pairs of tags to your buckets. For example, you can create a bucket to store financial reports from the financial department in your organization. You can tag this bucket with the following key-value pairs:

Key	Value
Department	*Finance*
Report	*Monthly*

Bucket names are globally unique and the namespace is shared between all vCloud Director organizations. After a bucket is created, the name of that bucket cannot be used for another bucket in any of the vCloud Director organizations until that bucket is deleted. Bucket names must adhere to the S3 bucket naming requirements. See Amazon S3 Bucket Naming Requirements.

Object

Objects in vCloud Director Object Storage Extension are the files that you upload to your buckets.

You can categorize objects within a bucket by adding key-value pairs of tags. If you are the bucket owner of the bucket that stores an object, you can add properties to the objects by defining metadata in the form of a key-value pair.

Organization administrators can access and manage the objects that all users in the same organization own. Organization users can access and manage the objects that they own and the objects that are shared with them.

You can preview image, text, PDF, audio, and video files directly in the user interface of vCloud Director Object Storage Extension.

Security Credential

vCloud Director Object Storage Extension supports S3-compatible API and the AWS Signature V4 authentication. Security credentials are used for authenticating S3 API requests and consist of an access key and a secret key. vCloud Director Object Storage Extension supports user and application types of security credentials.

With S3 API requests authenticated with user credentials, you can access and manage buckets and objects that you own or that are shared with you.

With S3 API requests authenticated with application credentials, you can access and manage objects at the bucket level.

Users own and manage their security credentials using the vCloud Director Object Storage Extension user interface. See Working with Security Credentials.

Access Control Lists

With access control lists (ACLs) you manage the access to buckets and objects. With ACLs you can share objects and buckets with other users from your vCloud Director organization. vCloud Director Object Storage Extension supports a list of predefined, canned ACLs. You can also create custom ACLs.

Supported S3 API Operations

vCloud Director Object Storage Extension runs on top of a Cloudian HyperStore storage cluster. Both vCloud Director Object Storage Extension and Cloudian HyperStore support most AWS S3 REST API operations.

Table 2-1. Supported S3 API Bucket Operations

Method	URL	Description
HEAD	`https://object-storage-IP-address:443/api/v1/s3/`*bucket-name*	Verifies if a bucket exists and if you have permissions to access it.
GET	`https://object-storage-IP-address:443/api/v1/s3/`*bucket-name*	Returns an XML representation of the objects in a bucket.
PUT	`https://object-storage-IP-address:443/api/v1/s3/`*bucket-name*	Creates a bucket.
DELETE	`https://object-storage-IP-address:443/api/v1/s3/`*bucket-name*	Deletes a bucket.
GET	`https://object-storage-IP-address:443/api/v1/s3/`*bucket-name*`?tagging`	Returns the tag set associated with the bucket.
PUT	`https://object-storage-IP-address:443/api/v1/s3/`*bucket-name*`?tagging`	Adds tags to a bucket.
DELETE	`https://object-storage-IP-address:443/api/v1/s3/`*bucket-name*`?tagging`	Deletes the tag set associated with the bucket.
GET	`https://object-storage-IP-address:443/api/v1/s3/`*bucket-name*`?acl`	Returns the access control list (ACL) configuration of the bucket.
PUT	`https://object-storage-IP-address:443/api/v1/s3/`*bucket-name*`?acl`	Sets the ACL permissions for the bucket.
GET	`https://object-storage-IP-address:443/api/v1/s3/`*bucket-name*`?encryption`	Returns the encryption configuration for a bucket.
PUT	`https://object-storage-IP-address:443/api/v1/s3/`*bucket-name*`?encryption`	Defines an encryption method for a bucket.
DELETE	`https://object-storage-IP-address:443/api/v1/s3/`*bucket-name*`?encryption`	Removes the encryption configuration from a bucket.
GET	`https://object-storage-IP-address:443/api/v1/s3/`*bucket-name*`?policy`	Returns the policy of a bucket.

Table 2-1. Supported S3 API Bucket Operations (continued)

Method	URL	Description
PUT	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name`*`?policy`	Defines a policy for a bucket.
DELETE	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name`*`?policy`	Removes a policy from a bucket.
GET	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name`*`?uploads`	Returns a list of in-progress multipart uploads.

Table 2-2. Supported S3 API Object Operations

Method	URL	Description
HEAD	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name/object-name`*	Verifies if an object exists and if you have permissions to access it.
GET	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name/object-name`*	Retrieves an object.
PUT	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name/object-name`*	Uploads an object to a bucket.
POST	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name/object-name`*	Adds an object to a bucket using HTML forms.
PUT	`https://object-storage-IP-address:443/api/v1/s3/`*`destination-bucket-name/destination-object-name`*	Copies an object from one bucket to another.
DELLETE	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name/object-name`*	Deletes an object.
DELETE	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name?delete`*	Deletes multiple objects.
POST	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name/object-name?uploads`*	Initiates a multipart upload.
GET	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name/object-name?MaxParts=MaxParts&PartNumberMarker=PartNumberMarker&UploadId=UploadId`*	Lists parts that are uploaded for a specific multipart upload.
PUT	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name/object-name?PartNumber=PartNumber&UploadId=UploadId`*	Uploads a part in a multipart upload.
PUT	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name/object-name?PartNumber=PartNumber&UploadId=UploadId`*	Uploads a part in a multipart upload by copying data from an existing object as a data source.
POST	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name/object-name?UploadId=UploadId`*	Completes a multipart upload by assembling previously uploaded parts.
DELETE	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name/object-name?UploadId=UploadId`*	Aborts a multipart upload.

Table 2-2. Supported S3 API Object Operations (continued)

Method	URL	Description
GET	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name/object-name?tagging`*	Returns the tags associated with an object.
PUT	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name/object-name?tagging`*	Adds tag or set of tags to an object.
DELETE	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name/object-name?tagging`*	Removes the tag set from an object.
GET	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name/object-name?acl`*	Returns the ACL configuration for the object.
PUT	`https://object-storage-IP-address:443/api/v1/s3/`*`bucket-name/object-name?acl`*	Sets an ACL for the object.

Error Responses

The vCloud Director Object Storage Extension S3 REST API supports the same error response format and error codes as the AWS S3 REST API.

For more information about the format and contents of error responses, and for the error codes, see the Error Responses topic in the AWS S3 documentation.

Authenticating vCloud Director Object Storage Extension REST API Requests

vCloud Director Object Storage Extension supports the AWS Signature and the vCloud API session authentication types.

AWS Signature

vCloud Director Object Storage Extension supports AWS Signature Version 4.

To authenticate vCloud Director Object Storage Extension REST API requests using AWS Signature type, you use security credentials. Security credentials are a pair of an access key and a secret key. vCloud Director Object Storage Extension supports user and application types of security credentials. Users own and manage their security credentials.

With S3 API requests authenticated with user credentials, you can manage all objects owned or shared by the owner of the user credentials. With application credentials, you control the S3 API access at the bucket level.

For more information about creating and working with security credentials, see the Working with Security Credentials topic in the *vCloud Director Object Storage Extension User's Guide for Tenant Users*.

In vCloud Director Object Storage Extension, only tenant users can own security credentials. To create and use security credentials, your user account requires the **tenant administrator** role or the **tenant user** role. For more information, see the Roles and Rights in vCloud Director Object Storage Extension topic in the *vCloud Director Object Storage Extension User's Guide for Tenant Users*.

For more information about AWS Signature authentication, see the Authenticating Requests (AWS Signature Version 4) topic in the AWS documentation.

vCloud API Session

To authenticate vCloud Director Object Storage Extension REST API requests, you can also use the vCloud API login mechanism of vCloud Director. For more information, see the Create a vCloud API Session topic in the *vCloud API Programming Guide for Service Providers*.

Common Request and Response Headers

vCloud Director Object Storage Extension supports the same common request and response headers as the AWS S3 REST API.

For more information, see the following topics in the AWS S3 API Reference:

- Common Request Headers
- Common Response Headers

Identity and Access Management

Any enabled vCloud Director organization tenant user can work with the vCloud Director Object Storage Extension S3 REST API.

vCloud Director Object Storage Extension uses the identity providers and configuration of vCloud Director.

For more information, see the Managing Identity Providers topic in the *vCloud Director Service Provider Admin Portal Guide*.

Access management to buckets and objects in vCloud Director Object Storage Extension is the same as in AWS S3. To share an object or a bucket with other users, you modify the access permissions for that object or bucket. For more information, see the Overview of Managing Access topic in the *AWS S3 Developer Guide*.

In the vCloud Director Object Storage Extension S3 REST API, buckets and objects are the entities that you can work with. By default, only the entity owner can access the entity. The user account that creates a bucket and uploads objects to it owns the bucket and the objects in the bucket.

You use bucket policies or access control lists(ACLs) to manage the access permissions for buckets and ACLs to manage the access permissions for objects.

Access Control Lists

Every bucket and every object have an ACL associated with them. An ACL is a list of grants that identifies grantee and permissions granted. You can use ACLs to grant permissions to other users within the same vCloud Director organization, or to make buckets and objects publicly accessible. You use ACLs to grant basic read/write permissions to a grantee.

You can use a set of built-in canned ACLs, or you can create a custom ACL.

The following table describes the permissions that you can use to create a custom ACL.

Permission	When applied to a bucket	When applied to an object
READ	Allows the grantee to list the objects in the bucket.	Allows the grantee to read the object data and its metadata.
WRITE	Allows the grantee to create, overwrite, and delete objects in the bucket.	Not applicable.
READ_ACP	Allows the grantee to read the ACL of the bucket.	Allows the grantee to read the ACL of the object.
WRITE_ACP	Allows the grantee to write the ACL for the bucket.	Allows the grantee to write the ACL for the object.
FULL_CONTROL	Grants **Read** and **Write** permissions for the bucket and **Read** and **Write** permissions for the ACL of the bucket.	Grants **Read** and **Write** permissions for the object and **Read** and **Write** permissions for the ACL of the object.

You can apply a canned ACL to both an object and a bucket. The following table describes the set of canned ACLs and the associated permissions.

Canned ACL	Description
Private	Only the owner can access the bucket or the object.
Public-Read	Grants **Read** permissions to all users.
Public-Read-Write	Grants **Read** and **Write** permissions to all users.
Authenticated-Read	Grants **Read** permissions to all authenticated users.

Bucket Policies

With vCloud Director Object Storage Extension S3 REST API, you can modify the access permissions of a bucket by adding a bucket policy to the bucket. Bucket policies supplement the capabilities of granting access permissions with ACLs. With bucket policies, you can, for example, grant access permissions with added conditions for a bucket to multiple vCloud Director organization users. You can also restrict the access to a bucket for a specific IP address or a specific HTTP referrer.

Making vCloud Director Object Storage Extension API Requests

Any client application that can send HTTP requests is an appropriate tool for developing REST applications with the vCloud Director Object Storage Extension S3 API.

Making API Requests Using Postman

You can use the Postman API Client to issue vCloud Director Object Storage Extension S3 API requests and view responses.

Make sure that you use AWS Signature type of authentication.

For more information, go to https://www.getpostman.com/.

Making API Requests Using cURL

If you use cURL to make vCloud Director Object Storage Extension S3 API requests, you must calculate the signature for each request. For more information about authenticating S3 API requests and signature calculations, see https://docs.aws.amazon.com/AmazonS3/latest/API/sig-v4-authenticating-requests.html.

Following is an example of an S3 API request to list the contents of a bucket using cURL.

```
curl -X GET https://vcloud-object-storace.example.com:8443/api/v1/s3 -H 'Authorization: AWS4-HMAC-
SHA256 Credential={{Access_Key}}/20190820/{{Region}}/s3/aws4_request, SignedHeaders=content-
type;host;x-amz-content-sha256;x-amz-date,
Signature=2916a7fe150115bdb3c93ac3f87d26de6eca59f64a7e15fd698fb61b7a1cefd7' -H 'Content-Type:
application/x-www-form-urlencoded' -H 'X-Amz-Content-Sha256:
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855' -H 'X-Amz-Date: 20190820T013058Z'
```

Making API Requests Using AWS SDK for Java

If you are developing an application with Java, to integrate the vCloud Director Object Storage Extension S3 API, use the AWS SDK for Java. See https://aws.amazon.com/sdk-for-java/.

To integrate your Java client with vCloud Director Object Storage Extension, include the following code sample in your project:

```
package com.vmware.voss.client;

import com.amazonaws.SDKGlobalConfiguration;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import org.junit.Test;

import static com.amazonaws.SDKGlobalConfiguration.DISABLE_CERT_CHECKING_SYSTEM_PROPERTY;

public class AWSSDKTest {
    @Test
    public void testWithAWSSDK() {

        String accessKey = "security-credentials-access-key";
        String secretKey = "security-credentials-secret-key";
        String endpoint = "https://vcloud-object-storace.example.com:8443/api/v1/s3";
        AwsClientBuilder.EndpointConfiguration endpointConfig = new
AwsClientBuilder.EndpointConfiguration(endpoint, null);
        AWSCredentials s3Credential = new BasicAWSCredentials(accessKey, secretKey);
        AmazonS3 client = AmazonS3ClientBuilder.standard()
                .withPathStyleAccessEnabled(true)
                .withCredentials(new AWSStaticCredentialsProvider(s3Credential))
                .withEndpointConfiguration(endpointConfig)
                .build();
        client.listBuckets();
    }

}
```

Making API Requests Using AWS SDK for Python

For developers using Python, to integrate the vCloud Director Object Storage Extension S3 API, use the AWS SDK for Python (Boto3). See https://aws.amazon.com/sdk-for-python/.

To integrate your Python application with vCloud Director Object Storage Extension, add the following sample to your code:

```
import boto3
import botocore

import logging
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s %(levelname)-8s %(message)s',
datefmt='%a, %d %b % %H:%M:%S')

ENDPOINT="https://vcloud-object-storace.example.com:8443/api/v1/s3"

client = boto3.client('s3', aws_access_key_id="security-credentials-access-key",
aws_secret_access_key="security-credentials-secret-key", endpoint_url=ENDPOINT)

client.list_buckets()
```

Making API Requests Using AWS SDK for Go

If you are developing an application with Go, to integrate the vCloud Director Object Storage Extension S3 API, use the AWS SDK for Go. See https://aws.amazon.com/sdk-for-go/.

To integrate your Go application with vCloud Director Object Storage Extension, add the following sample to your code:

```
sess := session.Must(session.NewSessionWithOptions(session.Options{
        SharedConfigState: session.SharedConfigEnable,
    }))

    creds := credentials.NewSharedCredentials(“./credentials”, “test-account”)
    DisableSSL := true
    awsConfig := aws.Config{
        Region:      aws.String(“us-west-1”),
        Credentials: creds,
        Endpoint:    "https://vcloud-object-storace.example.com:8443/api/v1/s3",
        //disable SSL only for test purposes
        DisableSSL:  &DisableSSL,
    }

    sess, err = session.NewSession(&awsConfig)

    sess.Config.HTTPClient.Transport = &http.Transport{
        TLSClientConfig: &tls.Config{InsecureSkipVerify: true},
    }

    s3ForcePathStyle := true
    svc := s3.New(sess, &aws.Config{
        S3ForcePathStyle: &s3ForcePathStyle,
    })
```

Common Workflows 3

This chapter contains step-by-step procedures that cover the common workflows for working with vCloud Director Object Storage Extension S3 API.

This chapter includes the following topics:

Manage Objects

Following the steps in the current example, you list all buckets that you own, create a bucket, upload an object to it, and add tags to the object.

Procedure

1 List all buckets owned by the authenticated user.

```
GET https://vcloud-object-storage.example.com:8443/api/v1/s3
```

The response body contains a list of all buckets that the authenticated user owns.

2 Create a bucket.

```
PUT https://vcloud-object-storage.example.com:8443/api/v1/s3/bucket-name
```

For more information, see https://docs.aws.amazon.com/AmazonS3/latest/API/API_CreateBucket.html.

The system returns a `200 OK` message upon a successful creation of the bucket.

3 Verify that the bucket is available and accessible.

```
HEAD https://vcloud-object-storage.example.com:8443/api/v1/s3/bucket-name
```

If the bucket is available and accessible for the authenticated user, the system returns a `200 OK` message.

4 Upload an object.

```
PUT https://vcloud-object-storage.example.com:8443/api/v1/s3/bucket-name/object-name
```

For more information, see https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutObject.html.

5 Add tags to the object.

Object tags are a key-value pairs that help you categorize objects and buckets.

```
PUT https://vcloud-object-storage.example.com:8443/api/v1/s3/bucket-name/object-name?tagging
```

Following is an example of a valid tagging request body in JSON and XML formats:

```
{
  "tagSets": [
    {
      "tags": [
        {
          "key": "string",
          "value": "string"
        }
      ]
    }
  ]
}
```

```
<Tagging>
  <TagSet>
     <Tag>
       <Key>string</Key>
       <Value>string</Value>
     </Tag>
  </TagSet>
</Tagging>
```

For more information, see https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutObjectTagging.html.

The system returns a `200 OK` message upon a successful completion of the operation.

Encrypt an Object

With the vCloud Director Object Storage Extension S3 API, you can encrypt individual objects for security purposes.

A tenant administrator can force a server-side encryption at vCloud Director organization level using the vCloud Director Object Storage Extension user interface. Encrypting objects using the vCloud Director Object Storage Extension API overrides the encryption configuration that is set using the vCloud Director Object Storage Extension user interface.

This procedure demonstrates how to encrypt an object using SSE-C type of encryption.

This type of encryption requires you to manage your encryption algorithms and master keys. The objects are encrypted as vCloud Director Object Storage Extension writes the data to disks in the data center and decrypts the data when you access it.

When you add an object, you provide the encryption key as part of the request. vCloud Director Object Storage Extension uses the encryption key to apply AES-256 encryption to your data and removes the encryption key from memory.

When you encrypt an object, you encrypt only the object data, not the object metadata.

When you want to retrieve your data, you provide the encryption key as part of your request. vCloud Director Object Storage Extension verifies that the encryption key matches the key used for the object upload. If the keys match, vCloud Director Object Storage Extension decrypts the object and returns the data to you.

Prerequisites

- Verify that you have an SSE-C encryption key. For more information about the encryption key specifics, see https://docs.aws.amazon.com/AmazonS3/latest/dev/ServerSideEncryptionCustomerKeys.html.
- Verify that you calculated a base64-encoded 128-bit MD5 digest of the encryption key.

Procedure

1 Upload and encrypt an object using your own encryption key.

  Add the following three headers to the vCloud Director Object Storage Extension S3 API request:

Header	Description
`x-amz-server-side-encryption-customer-algorithm`	Specifies the encryption algorithm. For the SSE-C encryption type, enter `AES256`.
`x-amz-server-side-encryption-customer-key`	Specifies the encryption key. Use this header to provide the 256-bit, base64-encoded encryption key. This key is used to encrypt and decrypt your data.
`x-amz-server-side-encryption-customer-key-MD5`	Use this header to enter the base64-encoded 128-bit MD5 digest of the encryption key.

```
PUT https://vcloud-object-storage.example.com:8443/api/v1/s3/bucket-name/object-name
```

  Upon a successful upload and encryption, the system returns a `200 OK` message.

2 Access the encrypted object.

  To access encrypted objects, you provide the encryption key and use the same headers that you used when uploading and encrypting the object.

  You can access the object using one of the following methods:

  - To retrieve an object, use the `GET` method.
  - To verify if the object is available and accessible for the authenticated user, use the `HEAD` method.

- To copy and object, use the PUT method. See https://docs.aws.amazon.com/AmazonS3/latest/API/API_CopyObject.html.

If you do not provide the encryption key, the system returns a 401 Unauthorized message.

Manage a Multipart Upload

With the multipart upload API operation, you can upload large objects in parts. You can also use the multipart upload API to copy an existing object.

Multipart upload consists of the following three substeps:

1. You initiate the upload.
2. You upload parts of the object.
3. After you upload all parts of the object, you complete the upload.

After the successful completion of the multipart upload, vCloud Director Object Storage Extension builds the object using all uploaded parts and you can access the object just as you access any other objects in your buckets.

You can list all multipart uploads that are in progress and you can get a list of all the parts that you uploaded for a specific multipart upload.

When you initiate a multipart upload, vCloud Director Object Storage Extension returns an upload ID that is a unique identifier for the multipart upload. You use the ID to upload parts, to complete the upload, or to abort it.

Procedure

1 List all multipart uploads in the bucket.

```
GET https://vcloud-object-storage.example.com:8443/api/v1/s3/bucket-name?uploads
```

The system returns details about all multipart uploads in the bucket.

2 Initiate a multipart upload.

```
POST https://vcloud-object-storage.example.com:8443/api/v1/s3/bucket-name/object-name?uploads
```

For more information about the operation and the request body, see https://docs.aws.amazon.com/AmazonS3/latest/API/API_CreateMultipartUpload.html.

Upon a successful creation of the multipart upload, the system returns an upload ID. Note the ID, as you need it to upload parts and to complete the upload.

3 Upload parts.

You must include the multipart upload ID and the part number in the upload part reqruest.

Part numbers can be any number from 1 to 10,000, inclusive. A part number uniquely identifies a part and also defines its position within the object being created. If you upload a new part using the same part number that was used with a previous part, the previously uploaded part is overwritten. Each part must be at least 5 MB, except the last part. There is no size limit on the last part of your multipart upload.

```
PUT https://vcloud-object-storage.example.com:8443/api/v1/s3/bucket-name/object-name?uploadId,
partNumber=number-of-the-part-you-upload
```

For more information, see https://docs.aws.amazon.com/AmazonS3/latest/API/API_UploadPart.html.

Upon a successful upload, the system returns a part number and an `ETag` value. Note the two values for each uploaded part, as you need them for completing the multipart upload.

4 List the previously uploaded parts.

```
GET https://vcloud-object-storage.example.com:8443/api/v1/s3/bucket-name/object-name?encoding-
type=encoding-method,max-parts=max-number-of-parts,part-number-marker=number,uploadId
```

By default, the request a maximum of 1000 uploaded parts. To change the default value, use the `max-parts` argument.

To select the part after which the listing begins, use the `part-number-marker`.

For more information, see https://docs.aws.amazon.com/AmazonS3/latest/API/API_ListParts.html.

5 Complete multipart upload.

After you successfully upload all parts of an object, use this vCloud Director Object Storage Extension S3 API request to build the object using all parts. vCloud Director Object Storage Extension concatenates all parts in ascending order by part number to create a new object.

You must provide a complete list of the uploaded parts in the request. The operation only concatenates parts that are present in the list. For each part in the list, you must provide the part number and the `ETag` value, returned after that part was uploaded.

This operation can take several minutes to complete, so vCloud Director Object Storage Extension periodically sends whitespace characters to keep the connection from timing out. Since the operation might fail after the initial `200 OK` response, regularly check the response body to verify if the request succeeded.

If you use the multipart upload API in your application, make sure that your application is prepared to retry failed requests.

```
POST https://vcloud-object-storage.example.com:8443/api/v1/s3/bucket-name/object-name?uploadId
```

For more information, see https://docs.aws.amazon.com/AmazonS3/latest/API/API_CompleteMultipartUpload.html.

Share an Object and a Bucket

To share an object, you use access control lists(ACLs). To set the ACL of an object, you must have the **WRITE_ACP** permission for the object.

Depending on your application architecture and needs, you can set the ACLs using the request headers or the body of the request.

When setting access permissions, you can use predefined ACL and specify the type with the `x-amz-acl` header, or explicitly specify the permissions you grant with the following headers:

- `x-amz-grant-read`
- `x-amz-grant-read-acp`
- `x-amz-grant-write-acp`
- `x-amz-grant-full-control`

To grant permissions, you specify the grantee of the permission as a type-value pair. Following are the available grantee types:

- To use the user name of an organization user, use the `id` type.
- To grant permissions to a predefined group, use the `uri` type.

For more information about working with ACLs, see https://docs.aws.amazon.com/AmazonS3/latest/dev/acl-overview.html.

Following the steps in the current procedure, you assign the **Read of Object**, the **Write of Object**, and the **Write of Bucket** permissions to an organization user. Then assign **Public Read** permissions for the object.

Prerequisites

- Verify that you created an object in a bucket. For example, *report_dec.xlsx*, in the *Reports* bucket.
- Verify that you have **WRITE_ACP** permissions for the *report_dec.xlsx* object and for the *Reports* bucket.

Procedure

1 Assign **Read of Object** permissions to an organization user.

```
PUT https://vcloud-object-storage.example.com:8443/api/v1/s3/Reports/report_dec.xlsx
```

If you use the request headers, use the `x-amz-grant-read` header and the user ID of the grantee.

If you use the request body, see the following example of a valid request body in JSON and XML formats:

```
{
  "grants": [
    {
      "grantee": {
        "id": "ID-of-Jane-Doe"
      },
      "permission": "READ"
    }
  ],
  "owner": {
    "displayName": "Jane Doe"
  }
}
```

```
<AccessControlPolicy>
  <Owner>
    <ID>ID-of-Jane-Doe</ID>
    <DisplayName>Jane Doe</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee>
        <ID>ID-of-John-Doe</ID>
        <DisplayName>John Doe</DisplayName>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

The grantee can now download the object.

2 To allow the organization user to upload updated versions of the object to the bucket, assign **Write of Bucket** permissions for the bucket.

```
PUT https://vcloud-object-storage.example.com:8443/api/v1/s3/Reports
```

If you use the request headers, use the `x-amz-grant-write` header and the user ID of the grantee.

If you use the request body, see the following example of a valid request body in JSON and XML formats:

```
{
  "grants": [
    {
      "grantee": {
        "id": "ID-of-John-Doe"
      },
      "permission": "WRITE"
    }
  ],
  "owner": {
    "displayName": "Jane Doe"
  }
}
```

```
<AccessControlPolicy>
  <Owner>
    <ID>ID-of-Jane-Doe</ID>
    <DisplayName>Jane Doe</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee>
        <ID>ID-of-John-Doe</ID>
        <DisplayName>John Doe</DisplayName>
      </Grantee>
      <Permission>WRITE</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

The grantee can now upload updated versions of the object to the bucket.

3 To enable sharing the report to internal and external users, make the object publicly readable.

```
PUT https://vcloud-object-storage.example.com:8443/api/v1/s3/Reports/report_dec.xlsx
```

Use the `x-amz-acl` header and the `public-read` value.