S3 Data Services

API Reference Guide

Version 5.2.7 | 96-00532-001 | Revision A7

Information in this document is subject to change without notice and does not represent a commitment on the part of DataDirect Networks, Inc. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose other than the purchaser's personal use without the written permission of DataDirect Networks, Inc.

The cited documents are available at: https://www.ddn.com/company/resource-library/

For archived versions of either document, please contact DDN.

December 2022

Changes in this Revision

The following information has been added or updated in this revision of API Reference Guide:

Section	Description
—	This release of S3DS is version 5.2.7. S3DS versioning has changed from 1.4.x to 5.2.x to align with the EXAScaler versioning.
Section 2.1 on page 11	Added specially qualified S3 standard clients

Preface

This guide provides information about EXAScaler Access S3 APIs calls for managing cloud storage resources.

The EXAScaler Access S3 API is compatible with Amazon S3 API. Special attention is given to describe the details on compatibility, including differences, features, and functionalities that are not supported by the EXAScaler Access S3 API.

Related Documents:

- Amazon Simple Storage Service API Reference API Version 2006-03-0: http://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html
- Amazon Simple Storage Service Developer Guide API Version 2006-03-01: http://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html

Audience

This guide is intended for application developers responsible for creating or adapting a software application to store data using EXAScaler Access S3 compatible API. It assumes the reader is familiar with REST API, Amazon S3 API, network protocols, and related technologies.

Glossary

The abbreviations and names used in this document are defined in the following table.

Term	Explanation
account	A virtual allocation of storage associated with a set of credentials.
ASCII	American Standard Code for Information Interchange - A character-encoding scheme. ASCII codes represent text in computers, communications equipment, and other devices that use text.
ACL	Access Control List - A list of permissions attached to a bucket or an object.
bucket	A container for objects stored in EXAScaler Access S3.
CNAME	Canonical Name Record - An entry in a Domain Name System (DNS) table that lets you alias one fully qualified domain name to another.
DNS	Domain Name System – A hierarchical, distributed naming system for computers, services, or any resource connected to the Internet or a private network. A DNS server resolves DNS domain names and host names to their corresponding numeric Internet Protocol (IP) addresses.
Endpoint	A Uniform Resource Locator (URL) that identifies a host and port as the entry point for a web service.
ETag	An entity tag of an object. The entity tag is a hash of the object. The ETag reflects changes to the contents of an object.
grant	A permission for accessing the storage via EXAScaler Access S3.
grantee	An account or one of the predefined groups with specific access rights.
key	A key is a unique identifier for an object within a bucket.
metadata	A set of name-value pairs that describe the object.
object	Objects are the fundamental entities, stored in the EXAScaler Access S3 via S3 API. Objects consist of object's data and metadata.

Term	Explanation
OpenStack	An open-source cloud operating system that controls large pools of compute, storage, and networking resources, all managed and provisioned through a web interface.
part	In a multipart upload request, each part is a contiguous portion of the object's data.
REST	Representational State Transfer – A simple, stateless web services architecture for a distributed hypermedia system, consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements. An alternative to SOAP.
S3	Simple Storage Service – A modular, scalable, tiered object storage service offered by Amazon and accessible using a simple web interface. Includes unlimited object versioning and hierarchical storage management of objects. Units of S3 storage capacity are known as “buckets”.

Table of Contents

List of Tables

1. EXAScaler Access S3 Platform

EXAScaler Access S3 SDS Protocols

EXAScaler Access S3 Storage is an object storage platform that supports the most popular Object Software Defined Storage (SDS) protocols:

- EXAScaler Access S3 – Compatible with Amazon S3

The EXAScaler Access S3 platform with S3 API provides massive scalability and ensures high availability even after hardware failures. In EXAScaler Access S3 API, objects are protected by storing multiple copies of data so that if one node fails, the data can transparently be retrieved from another node. The number of copies kept is configurable and can be easily adjusted to support different customer cases for more protection and better performance.

Like Amazon S3, the EXAScaler Access S3 platform is designed for eventual consistency. In other words, there is a guarantee that the system will eventually become consistent. This design also makes it ideal when performance and scalability are critical, particularly for massive, highly distributed infrastructures with lots of unstructured data serving global sites.

EXAScaler Access S3 API shares common namespace but have different authentication models:

- S3 API uses signature calculated over shared secret and the HTTP request. For more information on AWS authentication model, refer to the *Amazon Simple Storage Service Developer Guide API Version 2006-03-01*.

2. EXAScaler Access S3-Compatible API

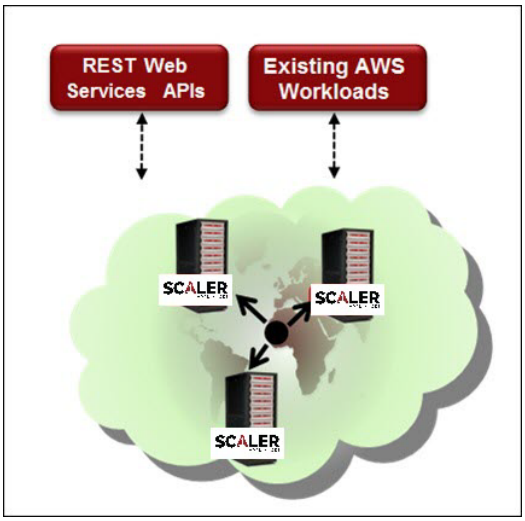
This section provides information on the following topics:

2.1 Overview

The EXAScaler Access S3 API is compatible with the Amazon Web Services (AWS) REST S3 API, providing a smooth transition for the S3 applications when switching from an Amazon public cloud to an S3 private cloud with a simple reconfiguration.

In Figure 1, the EXAScaler Access S3 storage with inter operable API access moves a defined and existing AWS workload to a private cloud. In this private cloud, client applications interact with state-managed methods through Representational State Transfer (REST) APIs to implement various operations, including creating objects and buckets, managing existing buckets and objects, uploading and downloading objects, and assigning permission to access an object or bucket to a given account.

Figure 1. EXAScaler Access S3 Compatibility



EXAScaler Access S3 API provides a well-matched interface for developers to Amazon S3 and allows applications written for these platforms to inter-operate with DDN-enabled storage clouds.

EXAScaler Access S3 API supports the REST interface only (There is no support for deprecated S3 SOAP interface). Customers can access the Cloud interface through HTTP or HTTPS.

EXAScaler Access S3 API supports all S3 standard clients, with the following specially qualified:

- s3curl
- s3cli
- aws cli
- s3cmd

Refer to the *S3 Data Services Administration Guide* for information on the EXAScaler Access S3 platform.

2.2 Endpoint (URL)

EXAScaler Access S3 API is a web-based API. Similar to your Web browser accessing web-pages over the internet, the S3 applications use web addresses (or hostnames) to access S3 buckets and objects.

NOTE:	In the S3 terminology, buckets are basic containers for objects. The S3 object must reside in a single bucket. A bucket cannot be created within another Bucket.

An endpoint is an address (hostname) where S3 data resides. The endpoint corresponds to a cluster of EXAScaler Access S3 servers.

In most cases, the endpoint is the hostname of the load-balancer system, where the load-balancer is configured to pass the connections to S3 MetaNodes. Refer to the *S3 Data Services Administration Guide* for more information on load balancers.

Endpoint Configuration

Many S3 clients, such as *Cyberduck*, *Cloudberry*, and *S3 Browser* work with EXAScaler Access S3 with minimal reconfiguration required.

The S3 client applications come preconfigured with the Amazon S3 endpoint (*s3.amazonaws.com*). To access EXAScaler Access S3, reconfigure S3 applications from *s3.amazonaws.com* to the customer EXAScaler Access S3 endpoint, for example *s3.acme.net*.

NOTE:	In the following examples, we assume that the endpoint is *s3.acme.net*.

For instructions on accessing the EXAScaler Access S3 platform using client applications, refer to the *S3 Data Services Administration Guide*.

2.3 Virtual Hosting of Buckets

Two access patterns work with S3 buckets and objects:

- The *path-style* access method, which makes REST requests using this URL:
  `http://s3.acme.net/bucketname/object`
- The *virtual-hosted-style* access method, which makes REST requests using this URL:
  `http://bucketname.s3.acme.net/object`

Additionally, the EXAScaler Access S3 API supports CNAME method. This method allows you to configure your DNS name as a CNAME alias for EXAScaler Access S3 and to completely customize the URL of your EXAScaler Access S3 resources, for example, **`http://my.bucketname.com/`**. For more information on virtual hosting, see Virtual Hosting of Buckets.

2.4 Bucket Naming Requirements

2.4.1 Virtual-hosted-style Access Method

By default, most S3 applications use a virtual-hosted-style to access buckets and objects: http://bucketname.s3.acme.net/object. Therefore, the bucket naming must conform to Internet and DNS naming standards.

NOTE: If a bucket's hostname does not follow these standards, you will not be able to access this bucket through the Internet.

The Internet standards precisely define the rules for hostnames naming.

The rules for DNS-compliant bucket names are as follows:

- Must be longer than 3, and shorter than 63 characters
- Must start with a number or letter
- Can contain only lowercase English-alphabet [Q: or Latin alphabet] letters (a-z), numbers (0-9), periods (.) and hyphens (-)
- Bucket names cannot contain underscores (_)
- Must not end with a hyphen (-)
- Must not contain character combinations, for example: "..", "-.", or ".-"

2.4.2 Path-style Access Method

As opposed to a virtual-hosted-style access, S3 applications can also use a path-style access method, accessing buckets and objects as **`http://s3.acme.net/bucketname/object`**.

NOTE:	Path-style patterns allow the use of more relaxed rules for bucket naming, which follow the same rules for Amazon's US Standard region.

The rules are as follows:

- Bucket names must be longer than 3, and shorter than 255 characters.
- Bucket may contain both lowercase and uppercase English-alphabet letters (a-z and A-Z), numbers (0-9), periods (.), and hyphens (-) as well as underscore characters (_).
- The bucket name does not need to start with a letter, and there are no restrictions regarding the illegal character combinations.

Refer to Rules for Bucket Naming for more information on rules for US Standard region.

CAUTION:	When using a path-style bucket access, it is possible to create bucket names that do not comply with virtual-hosted-style (for example, using bucket names longer than 63 characters). S3 applications that use only the virtual-hosted-style bucket access may fail to connect to the buckets (and objects in them), because their names do not comply with the DNS hostname rules.

The EXAScaler Access S3 API relaxed bucket naming rules may be turned off through the **`bucket_strict_name_check`** property in the *cloud-adv.service* configuration file, in which case a single strict rule applies for both access styles.

NOTE:	Any slash in the bucket name must be avoided. Including a slash "/" in the bucket name for a PUT Bucket request could have unexpected side effects, for example, an empty object being created under the root bucket name.

2.5 Object Names

Object names can be up to 238 bytes long (218 bytes if object versioning is enabled, 208 bytes if using multipart upload). Object names can contain any combination of letters and numbers.

The HTTP standards, that Web Services rely on, require that the URLs and HTTP headers are sent as US-ASCII characters (Refer to RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax for more information on the generic URI syntax and a process for resolving URI references).

The S3 application may still use object names composed in non-English alphabets and character encodings such as UTF-8, but it will need to be URI-encoded before sending to the EXAScaler Access S3 service (For more details, see Percent-encoding). In most cases, this encoding is handled transparently by the core HTTP communication libraries.

2.6 Authentication

The authentication process verifies the identity of a user, who is trying to access the EXAScaler Access S3. The user credentials (*Access Key* and *Secret Key*) uniquely identify the users of the EXAScaler Access S3 platform and provide access to user data.

2.6.1 Access Key

Access Key is an alphanumeric text string that uniquely identifies the user, who owns the account. Two users cannot have the same *Access Key*.

2.6.2 Secret Key

Secret Key plays the role of a password. It is called secret because it is assumed to be known only by the EXAScaler Access S3 server and the owner. This *Secret Key*, along with the *Access Key*, confirms the identity of the user for every S3 call. You are advised to keep your Secret key in a safe place.

The EXAScaler Access S3 web service uses **AWS Signature Version 2** and **Version 4** protocols for authenticating API requests.

More specifically, the security is based on “shared secret” principle and the *HMAC-SHA1* cryptographic algorithm. The shared secret is the user’s *Secret Key*, which is known to both client and the server.

To authenticate the request, both the S3 client and the EXAScaler Access S3 server independently calculate the HMAC-SHA1 checksum based on the *Secret Key,* which gets combined with the parts of the REST HTTP request (such as date, bucket, and object). If the EXAScaler Access S3 server calculates the same signature as the S3 client did, this proves that the client has used the same Secret Key to compute the signature (server “knows” the S3 client had the same Secret Key). Thus, the user’s request is authenticated.

2.6.3 Authorization

All resources stored in a storage are private by default. In other words, only the resource owner, the user that created it, can access the resource. The resource owner can grant access permissions to another user or a predefined group (*AuthenticatedUsers, AllUsers*) by writing an access policy.

Only authenticated users may perform the following operations:

1. `GET Service` operation to list buckets.
2. `PUT` and `DELETE Bucket` operation to create or delete bucket.
3. Users may store objects in a bucket, if the bucket has the explicit `WRITE` permission given to the person storing the object.

   To be able to store an object into a bucket, a user will need to satisfy at least one of the following requirements:

   a. Authenticated user is the owner of the bucket.

   b. Bucket has WRITE permission granted to the explicit authenticated user.

c. Bucket has WRITE permission given to the *AuthenticatedUsers* group and authenticated user makes the request.

d. Bucket has WRITE permission given to the *AllUsers* group. In this case, even the anonymous or non-authenticated users may upload objects to the bucket.

FULL_CONTROL permission or ownership of the object or bucket automatically grants all permissions (READ, WRITE, READ_ACP and WRITE_ACP).

Refer to the *Amazon Simple Storage Service Developer Guide* for more information on access permissions.

2.6.4 Authentication Header

EXAScaler Access S3 API uses the standard HTTP Authorization header, identical to Amazon, to pass authentication information.

2.6.4.1 AWS Signature Version 2

The Authorization header has the following form:

Authorization: AWS AccessKeyId:Signature

For request authentication, the **AccessKeyId** element identifies the *Access Key* that was used to compute the signature. The signature element is RFC 2104 - HMAC-SHA1 of the selected elements from the request.

For an authenticated S3 REST request, the guest will get an object from *mybucket*:

```
GET /photos/mycat.jpg HTTP/1.1
HOST: mybucket.s3.acme.net
Date: Wed, 01 Mar 2012 12:00:00 GMT

Authorization: AWS AKIAIOSFODNN7EXAMPLE:
bWq2s1WEIj+Ydj0vQ697zp+IXMU=
```

If a guest signs in with a string:

```
GET\n
\n
\n
Wed, 01 Mar 2012 12:00:00 GMT
mybucket/photos/mycat.jpg
```

where `bWq2s1WEIj+Ydj0vQ697zp+IXMU=` is the calculated HMAC-SHA1 signature.

Refer to the *Amazon Simple Storage Service Developer Guide* for more information on the authentication method.

2.6.4.2 AWS Signature Version 4

The Authorization header has the following form:

```
Authorization: AWS4-HMAC-SHA256 Credential=
AKIA7C77S13697R8024L/20170912/us-east-1/s3/aws4_request,SignedHeaders=host;x-amz-content-sh
a256;x-amz-date,Signature=e604aa37bae932f8731033c41e619f63367f510340e32a7025cd44bcb4bcdbfc
```

NOTE: Unlike AWS Signature Version 4 signing keys, which are valid for up to seven days, DDN S3 signing key must be re-calculated for every new Signature Version 4 request made.

2.6.5 Query String Authentication

EXAScaler Access S3 supports Query String Authentication (QSA). QSA provides browser-compatible signed GET requests with a specified expiration date.

AWS Signature Version 2

The following `GET` parameters are required:

- Expires - The number of seconds since the epoch after which the URI is no longer valid.
- Signature - The URL-encoding of the base-64 encoding of the SHA1 of the string to sign.

AWS Signature Version 4

The following `GET` parameters are required. For more information on how to use QSA, please refer to the AWS documentation.

- X-Amz-Algorithm
- X-Amz-Credential
- X-Amz-Date
- X-Amz-Expires
- X-Amz-SignedHeaders
- X-Amz-Signature

2.6.6 Authentication Possible Error Responses

- AWS Signature Version 2
- AWS Signature Version 4

AWS Signature Version 2

If the signature has expired, one or more required parameters will be missing from the request:

```
403 Forbidden
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>AccessDenied</Code>
  <Message>Access Denied</Message>
  <RequestId>000001481EFF1239</RequestId>
  <HostId>00000000000000000</HostId>
</Error>
```

If a signature does not match, the S3 server sends the following message:

```
403 Forbidden
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>SignatureDoesNotMatch</Code>
  <Message>The request signature we calculated does not match the signature you provided.
Check your key and signing method.
</Message>
  <StringToSignBytes>73, -55, -45, -115, -20, -84, 33, -7, -5, -89, -128, 97, 109, -54,
14, -65, 75, 61, -21, -32</StringToSignBytes>
<SignatureProvided>EkJD6DZtVAHcD1b+/9T1cVs8M3A=_</SignatureProvided>
<StringToSign>GET1409369281/testbucket/testObject.txt</StringToSign>
  <AccessKeyId>AKIA635AOOS4D4Q3PVL4</AccessKeyId>
  <RequestId>000001481F00CB65</RequestId>
  <HostId>00000000000000000</HostId>
</Error>
```

NOTE: If you see the above message, verify the Access Key or Secret Key on both the EXAScaler Access S3 server and the S3 application.

If an incorrect Access Key or Secret Key was not the cause for this error, examine *StringToSign*.

In the following example, the value specified by the **`Request`** parameter represents a date or time earlier than that of the time of the request.

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>AccessDenied</Code>
  <Message>Request has expired</Message>
  <RequestId>REQUEST_ID</RequestId>
  <Expires>YYYY-MM-DDTHH:MM:SSZ</Expires>
  <HostId>HOST_ID</HostId>
  <ServerTime>YYYY-MM-DDTHH:MM:SSZ</ServerTime>
</Error>
```

AWS Signature Version 4

Table 1 below lists the errors that can be raised by Authentication V4 failures.

Table 1. Errors that can be raised by Authentication V4 failures

Error Code	Error Message	Response
403, Forbidden	AccessDenied	403 Forbidden <?xml version="1.0" encoding="UTF-8"?><Error><Code>AccessDenied</Code><Message>AWS authentication requires a valid Date or x-amz-date header</Message><RequestId>74C82612BF89B0DC</RequestId><HostId>0000000000000000</HostId></Error>
400, Bad Request	AuthorizationHeaderMalformed	Response code: 400 <?xml version="1.0" encoding="UTF-8"?> <Error><Code>AuthorizationHeaderMalformed</Code> <Message>The authorization header is malformed; the region 'us-east-3' is wrong; expecting 'us-east-1'</Message> <Region>us-east-1</Region><RequestId>103CF3E05FACCB98</RequestId> <HostId>00000000000000000</HostId></Error>
400, Bad Request	AuthorizationQueryParametersError	400 Bad Request <?xml version="1.0" encoding="UTF-8"?> <Error> <Code>AuthorizationQueryParametersError</Code> <Message>Invalid credential date "20171010". This date is not the same as X-Amz-Date: "20171012".</Message> \ <RequestId>B91B8308B7048CF8</RequestId> <HostId>00000000000000000</HostId> </Error>
403, Forbidden	SignatureDoesNotMatch	403 Forbidden <?xml version="1.0" encoding="UTF-8"?> <Error> <Code>SignatureDoesNotMatch</Code> <Message>The request signature we calculated does not match the signature you provided. Check your key and signing method. </Message> <StringToSignBytes>73, -55, -45, -115, -20, -84, 33, -7, -5, -89, -128, 97, 109, -54, 14, -65, 75, 61, -21, -32</StringToSignBytes> <SignatureProvided>EkJD6DZtVAHcD1b+/9T1cVs8M3A=_</SignatureProvided> <StringToSign>GET1409369281/testbucket/testObject.txt</StringToSign> <AccessKeyId>AKIA635AOOS4D4Q3PVL4</AccessKeyId> <RequestId>000001481F00CB65</RequestId> <HostId>00000000000000000</Hostid>
403, Forbidden	RequestTimeTooSkewed	403 Forbidden <?xml version="1.0" encoding="UTF-8"?><Error><Code>RequestTimeTooSkewed</Code> <Message>The difference between the request time and the server's time is too large.</Message><RequestTime>20171112T121212Z</RequestTime> <ServerTime>2017-11-15T12:05:41Z</ServerTime><MaxAllowedSkewMilliseconds>900000</MaxAllowedSkewMilliseconds> <RequestId>3D83DCB9BF92106C</RequestId><HostId>00000000000000000</HostId></Error>

2.7 REST API

The EXAScaler Access S3 web service is designed to be as close as possible to the AWS S3. This design includes the RESTful API calls, API responses, error replies, and the client-server authentication methods, as well as the format of the Access and Secret keys.

The EXAScaler Access S3 API provided by the EXAScaler Access S3 web service is compatible with the S3 applications designed to work with AWS S3. In most cases, the S3 application will only need to reconfigure the endpoint hostname from Amazon's *s3.amazonaws.com* to the customer's provided endpoint for EXAScaler Access S3 (for example, *s3.acme.net*).

This section lists all basic operations, with examples, that are supported by EXAScaler Access S3 API. Each operation has a brief description and a list of unsupported parameters or elements.

Please refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for more information on AWS S3 calls.

2.8 Common Headers

EXAScaler Access S3 API supports all common headers listed in the *Amazon Simple Storage Service API Reference, API version 2006-03-01* except the following*:*

Common Request Headers Not Supported

- x-amz-security-token
- x-amz-content-sha256

2.9 Operations Not Supported by EXAScaler Access S3 API

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NotImplemented</Code>
  <Message>A header or parameter you provided implies functionality that is not
implemented.</Message>
  <Feature>_feature_name_here_</Feature>
  <RequestId>00000146FA218E78</RequestId>
  <HostId>0000000000000000</HostId>
</Error>
```

EXAScaler Access S3 does not implement all of the AWS S3 API calls. The EXAScaler Access S3 server returns *501 Not Implemented* for the EXAScaler Access S3 API calls (or S3 calls modifications) that are not implemented.

Table 2 lists the features, along with individual bucket calls, that are not supported by the EXAScaler Access S3 and will return the *501 Not Implemented* error code:

Table 3 lists all operations on objects that are not supported by EXAScaler Access S3 API and will also return 501 Not Implemented error code.

Table 2. EXAScaler Access S3 API Bucket Calls - Not Implemented

Calls	Description
Bucket CORS Feature	
DELETE Bucket cors	Deletes the cors configuration information set for the bucket.
GET Bucket cors	Returns the cors configuration information set for the bucket.
PUT Bucket cors	Sets the cors configuration for your bucket.
Bucket Lifecycle Feature	
PUT Bucket Lifecycle	Creates a new lifecycle configuration for the bucket or replaces an existing lifecycle configuration.
GET Bucket Lifecycle	Returns the lifecycle configuration information set on the bucket.
DELETE Bucket Lifecycle	Deletes the lifecycle configuration from the specified bucket.
Bucket Notification Feature	
PUT Bucket notification	Enables notifications of specified events for a bucket.
GET Bucket notification	Returns the notification configuration of a bucket.
Bucket Replication Feature	
PUT Bucket Replication	Adds to or replaces a replication subresource to a bucket.
GET Bucket Replication	Returns the replication subresource of a specified bucket.
DELETE Bucket Replication	Deletes the replication subresource on a specified bucket.
Bucket Tagging Feature	
PUT Bucket Tagging	Adds a set of tags to an existing bucket.
GET Bucket Tagging	Returns the tag set associated with the bucket.
DELETE Bucket Tagging	Removes a tag set from the specified bucket.

Table 2. EXAScaler Access S3 API Bucket Calls - Not Implemented (Continued)

Calls	Description
Bucket Website Feature	
PUT Bucket Website	Sets the configuration of the website that is specified in the website sub-resource.
GET Bucket Website	Returns the website configuration associated with a bucket.
DELETE Bucket Website	Removes the website configuration for a bucket.

Table 3. EXAScaler Access S3 API Object Calls - Not Implemented

Calls	Description
GET Object torrent	Returns torrent files from a bucket.
POST Object Restore	Restores a temporary copy of an archived object.
OPTIONS object	Determines if it possible to send an actual request with the specific origin, HTTP method, and headers.

2.10 EXAScaler Access S3-Compatible API Error Codes

Table 4 lists the EXAScaler Access S3 API error codes.

Table 4. Error Codes

Error	Description	HTTP Status Code
`AccessDenied`	The authentication requires a valid Date or x-amz-date header.	`403 Forbidden`
`BadDigest`	The Content-MD5 you specified did not match what we received.	`400 Bad Request`
`BucketAlreadyExists`	The requested bucket name is not available. The bucket namespace is shared by all users of the system. Please select a different name and try again.	`409 Conflict`
`BucketAlreadyOwnedByYou`	Your previous request to create the named bucket succeeded and you already own it.	`409 Conflict`
`BucketNotEmpty`	The bucket you tried to delete is not empty.	`409 Conflict`
`EntityTooSmall`	Your proposed upload is smaller than the minimum allowed object size.	`400 Bad Request`
`EntityTooLarge`	Your proposed upload exceeds the maximum allowed object size.	`400 Bad Request`
`InternalError`	We encountered an internal error. Please try again.	`500 Internal Server Error`
`InvalidAccessKeyId`	The Access key you provided does not exist in our records.	`403 Forbidden`
`InvalidArgument`	Invalid Argument.	`400 Bad Request`
`InvalidBucketname`	The specified bucket is not valid.	`400 Bad Request`

Table 4. Error Codes (Continued)

Error	Description	HTTP Status Code
InvalidDigest	The Content-MD5 you specified is not valid.	400 Bad Request
InvalidPart	One or more of the specified parts could not be found. The part might not have been uploaded, or the specified entity tag might not have matched the part's entity tag.	400 Bad Request
InvalidPartOrder	The list of parts was not in ascending order. Parts list must specified in order by part number.	400 Bad Request
InvalidRange	The requested range cannot be satisfied.	416 Requested Range Not Satisfiable
InvalidTargetBucketForLogging	The target bucket you specified in PUT bucket logging does not exist or is not owned by you.	400 Bad Request
MalformedPolicy	The request contains invalid policy document. The JSON document must be well formatted and each element must be valid.	400 Bad Request
MetadataTooLarge	Your metadata headers exceed the maximum allowed metadata size.	400 Bad Request
MethodNotAllowed	The specified method is not allowed against this resource.	405 Method Not Allowed
MissingContentLength	You must provide the Content-Length HTTP header.	411 Length Required
MissingSecurityHeader	Your request is missing a required header.	400 Bad Request
NoSuchBucket	The specified bucket does not exist.	404 Not Found
NoSuchBucketPolicy	The requested bucket policy does not exist.	404 Not Found
NoSuchKey	The specified key does not exist.	404 Not Found
NoSuchUpload	The specified multipart upload does not exist. The upload ID might be invalid, or the multipart upload might have been aborted or completed.	404 Not Found
NotImplemented	A header you provided implies functionality that is not implemented.	501 Not Implemented
PermanentRedirect	The bucket you are attempting to access must be addressed using the specified endpoint. Send all future requests to this endpoint.	301 Moved Permanently
PreconditionFailed	At least one of the preconditions you specified did not hold.	412 Precondition Failed
RequestTimeout	Your socket connection to the server was not read from or written to within the timeout period.	400 Bad Request
RequestTimeTooSkewed	The difference between the request time and the server's time is too large.	403 Forbidden
SignatureDoesNotMatch	The request signature we calculated does not match the signature you provided. Check the Access key and signing method. For more information, see REST Authentication and SOAP Authentication.	403 Forbidden
TooManyBuckets	You have attempted to create more buckets than allowed.	400 Bad Request
UnexpectedContent	This request does not support content.	400 Bad Request

3. Operations Supported by EXAScaler Access S3 API

This chapter describes the operations, ACL settings, and bucket policy settings that are supported.

The list of unsupported elements can be found in Elements Not Supported by S3 Calls (Section 3.1 on page 26).

3.1 Elements Not Supported by S3 Calls

Table 5 illustrates the request and response elements that are not supported by all Service, Bucket, and Object operations.

Table 5. Request and Response Elements Not Supported

Calls	Request Elements Not Supported	Response Elements Not Supported
Service Calls		
GET Service	---	• `DisplayName`
Bucket Calls		
GET Bucket	---	• `DisplayName`
GET Bucket ACL		
GET Bucket Object versions		
GET Bucket versioning		
PUT Bucket ACL		
PUT Bucket	• `CreateBucketConfiguration` • `LocationConstraint`	---
Object Calls		
DELETE Object	• `x-amz-mfa`	---
GET Object	• `x-amz-server-side-encryption-customer-algorithm` • `x-amz-server-side-encryption-customer-key` • `x-amz-server-side-encryption-customer-key-MD5`	• `x-amz-expiration` • `x-amz-server-side-encryption` • `x-amz-server-side-encryption-customer-algorithm` • `x-amz-server-side-encryption-customer-key-MD5` • `x-amz-restore` • `x-amz-website-redirect-location`
GET Object ACL	---	• `DisplayName`
HEAD Object	• `x-amz-server-side-encryption-customer-algorithm` • `x-amz-server-side-encryption-customer-key` • `x-amz-server-side-encryption-customer-key-MD5`	• `x-amz-expiration` • `x-amz-missing-meta` • `x-amz-restore` • `x-amz-server-side-encryption`
PUT Object	• `x-amz-storage-class` (This header supports only the Standard value) • `x-amz-website-redirect-location` • `x-amz-server-side encryption`	---
PUT Object ACL	• `DisplayName`	---

Table 5. Request and Response Elements Not Supported (Continued)

Calls	Request Elements Not Supported	Response Elements Not Supported
PUT Object Copy	• x-amz-website-redirect-location • x-amz-server-side-encryption • x-amz-server-side-encryption-customer-algorithm • x-amz-server-side-encryption-customer-key • x-amz-server-side-encryption-customer-key-MD5	---
Initiate Multipart Upload	• x-amz-storageclass • x-amz-website -redirect-location • x-amz-server-side-encryption • x-amz-server-side-encryption-customer-algorithm • x-amz-server-side-encryption-customer-key • x-amz-server-side-encryption-customer-key-MD5	• x-amz-server-side-encryption • x-amz-server-side-encryption-customer-algorithm • x-amz-server-side-encryption-customer-key-MD5
Upload Part	• x-amz-server-side-encryption • x-amz-server-side-encryption-customer-algorithm • x-amz-server-side-encryption-customer-key • x-amz-server-side-encryption-customer-key-MD5	
Upload Part - Copy	• x-amz-server-side-encryption • x-amz-server-side-encryption-customer-algorithm • x-amz-server-side-encryption-customer-key • x-amz-server-side-encryption-customer-key-MD5 • x-amz-copy-source-server-side-encryption-customer-algorithm • x-amz-copy-source-server-side-encryption -customer-key • x-amz-copy-source-server-side-encryption-customer-key-MD5	---
Complete Multipart Upload	---	• x-amz-expiration • x-amz-server-side-encryption • x-amz-server-side-encryption-customer-algorithm
List Parts	---	• DisplayName

3.2 Service Operations

The EXAScaler Access S3 API Service operation is similar to the AWS S3 Service operation. Please refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for more information on the AWS S3 Service operation.

3.2.1 GET Service

Description

This implementation of **`GET`** operation returns a list of buckets that the authenticated sender owns. This operation is done by making a GET request without specifying a bucket or object. This request must be authenticated.

Requests

s3curl Command Request:

```
s3curl.pl --id=ID "http://s3.acme.net"
```

Syntax

```
GET / HTTP/1.1
Host: s3.acme.net
Date: date
Authorization: authorization string
```

Responses

Response Elements

Uses the same response elements as the Amazon S3. Please refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for the list of **`GET Service`** response elements.

Response Elements Not Supported

- **`DisplayName`** (Universal Unique Identifier, *UUID*, is displayed instead)

Examples

Sample Request:

```
GET / HTTP/1.1
Host: s3.acme.net
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: authorization string
```

Sample Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult xmlns="http://s3.acme.net/doc/2007-07-07/">
  <Owner>
    <ID>ff2ae2c9a047cec671c0561349d4015961ac9eea</ID>
    <DisplayName>ff2ae2c9a047cec671c0561349d4015961ac9eea
    </DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <Name>quotes</Name>
      <CreationDate>2014-08-28T23:38:24.000Z</CreationDate>
    </Bucket>
    <Bucket>
      <Name>samples</Name>
      <CreationDate>2014-08-28T06:09:39.000Z</CreationDate>
    </Bucket>
  </Buckets>
</ListAllMyBucketsResult>
```

3.3 Bucket Operations

The EXAScaler Access S3 API Bucket basic operations implementation is similar to the AWS S3 Bucket operations. Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for more information on the AWS S3 Bucket operations.

See Table 2 for the list of Bucket operations that are not supported.

NOTE: *Bucket Replication* is NOT supported.

3.3.1 DELETE Bucket

Description

This operation deletes an empty bucket. All objects as well as all object versions and delete markers in the bucket must be deleted before the bucket itself can be deleted.

To use this operation, the user must have **`WRITE`** permission on the bucket.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --delete "http://bucketname.s3.acme.net"
```

Syntax

```
DELETE / HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
```

Request Headers

Uses only the request headers that are common to all operations (see Common Headers).

Responses

Response Headers

Uses the response headers that are common to most responses (see Common Headers).

Examples

Sample Request:

```
DELETE / HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
```

Sample Response:

```
HTTP/1.1 204 No Content
x-amz-id-2: 1C7B1231CB0E3939
x-amz-request-id: 32FE2CEB32F5EE25
Date: Wed, 12 July 2014 17:50:00 GMT
Connection: close
Server: DDN S3
```

3.3.2 DELETE Bucket Policy

Description

The implementation of the DELETE operation deletes the policy subresource from the bucket. Only the bucket owner can delete the policy document.

Requests

s3curl Command Example:

```
s3curl.pl --delete --id=ID "http://bucketname.s3.acme.net?policy"
```

Syntax

```
DELETE /?policy HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
```

Request Headers

Uses only the request headers that are common to all operations (see Common Headers).

Responses

Response Headers

Uses only the response headers that are common to all operations (see Common Headers).

Examples

Sample Request:

```
DELETE /?policy HTTP/1.1
Host: mybucket.s3.acme.net
Date: Mon, 25 Feb 2018 14:54:24 GMT
Authorization: authorization string
```

Sample Response:

```
HTTP/1.1 204 No Content
x-amz-request-id: 1DD5E20CD0F88F96
x-amz-id-2: 1DD6342D9AD6284F
Date: Mon, 25 Feb 2019 14:54:27 GMT
Server: DDN S3
```

3.3.3 GET Bucket

Description

This implementation of **GET** operation lists all objects in a bucket. The **GET Bucket** request requires the caller to have at least **READ** permission on the bucket.

By default, only the first 1000 objects are returned by this call. You can use the *request* parameters such as *prefix*, *marker*, *delimiter*, *max-keys* to control the content and the amount of information returned. For more information, see Request Parameters below.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID "http://bucketname.s3.acme.net"
```

Syntax

```
GET / HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
```

Request Parameters

Request parameters could be used as selection criteria to return a subset of the objects in a bucket. The **GET Bucket** call uses the same request parameters as the Amazon S3. Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01 f*or the list of **GET Bucket** request parameters.

Responses

Response Headers

Uses only the response headers that are common to all operations (see Common Headers).

Response Elements

Uses the same response elements as the Amazon S3. Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for the list of **GET Bucket** response elements.

Response Elements Not Supported

- **DisplayName** (Universal Unique Identifier, *UUID*, is displayed instead)

Examples

Sample Request:

```
GET / HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 01 Mar 2014 12:00:00 GMT
Authorization: authorization string
Content-Type: text/plain
```

Sample Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.acme.net/doc/2007-07-07/">
  <Name>ztest-gd4rqmpf8ey5jqwn03vjn9d-178</Name>
  <Prefix/>
  <Marker/>
  <MaxKeys>1000</MaxKeys>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>mycat.jpg</Key>
    <LastModified>2014-08-28T06:15:11.000Z</LastModified>
    <ETag>"8a0c9ec4967487b18edb6a69d92ec625"</ETag>
    <Size>12</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>79991ba7d1b7cc9e3b5471dbbca0ba58d3a29953</ID>
<DisplayName>79991ba7d1b7cc9e3b5471dbbca0ba58d3a29953</DisplayName>
    </Owner>
  </Contents>
  <Contents>
    <Key>mydog.jpg</Key>
    <LastModified>2014-08-28T06:15:11.000Z</LastModified>
    <ETag>"09e5c9957cdea8a3f6e49050a3f4826c"</ETag>
    <Size>12</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>79991ba7d1b7cc9e3b5471dbbca0ba58d3a29953</ID>
<DisplayName>79991ba7d1b7cc9e3b5471dbbca0ba58d3a29953</DisplayName>
    </Owner>
  </Contents>
</ListBucketResult>
```

3.3.4 GET Bucket ACL

Description

The **GET Bucket ACL** operation uses the **acl** parameter to return the access control list (ACL) of a bucket.

The **GET Bucket ACL** request requires the caller to have at least the **READ_ACP** permission on the bucket.

If the **READ_ACP** permission is granted to the *AllUsers* group, the call will work even for "anonymous" requests (the requests that do not contain authorization headers).

Requests

s3curl Command Example:

```
s3curl.pl --id=ID "http://bucketname.s3.acme.net?acl"
```

Syntax

```
GET /?acl HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
```

Request Headers

Uses only the request headers that are common to all operations (see Common Headers).

Responses

Response Headers

Uses only the response headers that are common to all operations (see Common Headers).

Response Elements

Uses the same response elements as the Amazon S3. Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for the list of **`GET Bucket ACL`** response elements.

Response Elements Not Supported

- **`DisplayName`** (Universal Unique Identifier, *UUID*, is displayed instead)

Examples

Sample Request:

```
GET /?acl HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
```

Sample Response:

```
HTTP/1.1 200 OK
Date: Wed, 12 July 2014 17:50:00 GMT
x-amz-request-id: 000001482031D84B
x-amz-id-2: 00005360851D626F
Content-Type: application/xml;charset=UTF-8
Transfer-Encoding: chunked
Connection: close
Server: DDN S3

<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>ff2ae2c9a047cec671c0561349d4015961ac9eea</ID>
<DisplayName>ff2ae2c9a047cec671c0561349d4015961ac9eea
    </DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema- instance" xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>WRITE</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

3.3.5 GET Bucket Object versions

Description

To list all of the versions of every object in a bucket, use the **versions** subresource in the **GET Bucket** request. The request parameters returns metadata about a subgroup of all the object versions. For more information, see Request Parameters below.

To list metadata about all of the versions of objects, you must have **READ** access to the bucket.

EXAScaler Access S3 can retrieve only a maximum of 1000 objects, and each object version counts fully as an object.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID "http://bucketname.s3.acme.net?versions"
```

Syntax

```
GET /?versions HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
```

Request Headers

Uses only the request headers that are common to all operations (see Common Headers).

Request Parameters

Uses the same request parameters as the Amazon S3. Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for the list of **GET Bucket Object versions** request parameters.

Responses

Response Headers

Uses only the response headers that are common to all operations (see Common Headers).

Response Elements

Uses the same response elements as the Amazon. Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for the list of `GET Bucket Object versions` response elements.

Response Elements Not Supported

- **DisplayName** (Universal Unique Identifier, *UUID*, is displayed instead)

Examples

Sample Request:
This request returns all of the versions of every object in the bucket *mybucket*.

```
GET /?versions HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
```

Sample Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<ListVersionsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>mybucket</Name>
  <Prefix/>
  <KeyMarker/>
  <VersionIdMarker/>
  <MaxKeys>1000</MaxKeys>
  <IsTruncated>false</IsTruncated>
  <DeleteMarker>
    <Key>mycat.jpg</Key>
    <VersionId>OTIyMzM3MDU4MDM1MTYyNTk5MA==</VersionId>
    <IsLatest>true</IsLatest>
    <LastModified>2016-02-26T16:12:29.000Z</LastModified>
    <Owner>
     <ID>3d59ef416f34727b42504220fc9b5027b0d477e6</ID>
<DisplayName>3d59ef416f34727b42504220fc9b5027b0d477e6</DisplayName>
    </Owner>
  </DeleteMarker>
  <Version>
    <Key>mycat.jpg</Key>
    <VersionId>OTIyMzM3MDU4MDM1MTY0MzY5OA==</VersionId>
    <IsLatest>false</IsLatest>
    <LastModified>2016-02-26T16:12:12.000Z</LastModified>
    <ETag>"e2d28c49cda18df30d7c848a731f7ea6"</ETag>
    <Size>16</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>3d59ef416f34727b42504220fc9b5027b0d477e6</ID>
      <DisplayName>3d59ef416f34727b42504220fc9b5027b0d477e6
      </DisplayName>
    </Owner>
  </Version>
  <Version>
    <Key>mydog.jpg</Key>
    <VersionId>OTIyMzM3MDU4MTI5OTIzMDQyMw==</VersionId>
    <IsLatest>true</IsLatest>
    <LastModified>2016-02-26T16:12:12.000Z</LastModified>
    <ETag>"e2d28c49cda18df30d7c848a731f7ea6"</ETag>
    <Size>16</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>3d59ef416f34727b42504220fc9b5027b0d477e6</ID>
<DisplayName>3d59ef416f34727b42504220fc9b5027b0d477e6</DisplayName>
    </Owner>
  </Version>
</ListVersionsResult>
```

Requests Using Request Parameters

The request parameters retrieve a subgroup of object versions. For example, they can return all versions of a specific object, the requested number of versions (*max-keys* parameter) or for listing from a specific key or object version (*key-marker* and *version-id-marker* parameters).

Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for the list of **`GET Bucket Object versions`** request parameters.

Related recourses: Listing Object in a Versioning-Enabled Bucket

3.3.6 GET Bucket location

Description

This implementation of GET returns bucket region name using the location subresource.

Bucket region is the site name that you enter when you connect clusters in a multisite configuration using the *sitectl* utility.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID "http://bucketname.s3.acme.net?location"
```

Syntax

```
GET /?location HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
```

Request Headers

Uses only the request headers that are common to all operations (see Common Headers).

Responses

Response Headers

Uses the response headers that are common to most responses (see Common Headers).

Response Elements

This implementation of the operation returns the XML element, *LocationConstraint*, which specifies the region (site name), where the bucket resides.

Examples

Sample Request:
This request returns the region of *mybucket*.

```
GET /?location HTTP/1.1
Host: mybucket.s3.acme-east.net
Date: Tue, 24 Jan 2017 16:51:26 GMT
Authorization: authorization string
```

Sample Response:
The following sample response body shows that bucket resides in the acme-east region.

```
< HTTP/1.1 200 OK
< x-amz-request-id: 4DBA1C0ED163FE9F
< x-amz-id-2: 4DBA71783685798B
< Date: Tue, 24 Jan 2017 16:51:48 GMT
< Server: DDN S3

<?xml version="1.0" encoding="UTF-8"?>
<LocationConstraint xmlns="http://s3.amazonaws.com/doc/2006-03-01/">acme-east</LocationConstraint>
```

3.3.7 GET Bucket versioning

Description

To return the versioning state of a bucket, use the **`versioning`** subresource in the **`GET Bucket`** request. Only the bucket owner can retrieve the versioning state of a bucket.

The returned value is one of the following versioning states:

- *Enabled* state is returned if you enable the versioning on a bucket.
- *Suspended* state is returned if you suspend the versioning on a bucket.
- If you never enabled (or suspended) versioning on a bucket, no state will be returned in the response body.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID "http://bucketname.s3.acme.net?versioning"
```

Syntax

```
GET /?versioning HTTP/1.1
Host: bucketname.s3.acme.net
Content-Length: length
Date: date
Authorization: authorization string
```

Request Headers

Uses only the request headers that are common to all operations (see Common Headers).

Responses

Response Headers

Uses the response headers that are common to most responses (see Common Headers).

Response Elements

Uses the same response elements as the Amazon. Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for the list of **`GET Bucket versioning`** response elements.

Examples

Sample Request:
This request returns the versioning state of *mybucket*.

```
GET /?versioning HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
```

Sample Response:

The following is a sample of the response body that shows bucket versioning is enabled.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Status>Enabled</Status>
</VersioningConfiguration>
```

The following is a sample of the response body that shows bucket versioning is suspended.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Status>Suspended</Status>
</VersioningConfiguration>
```

If the versioning was never enabled (or suspended), the response is:

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/"/>
```

3.3.8 GET Bucket Policy

Description

The GET Bucket policy operation uses the policy parameter to return the policy document of a bucket. Only the bucket owner can get the bucket policy.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID "http://bucketname.s3.acme.net?policy"
```

Syntax

```
GET /?policy HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
```

Request Headers

Uses only the request headers that are common to all operations (see Common Headers).

Responses

Response Headers

Uses only the response headers that are common to all operations (see Common Headers).

Examples

Sample Request:

```
GET /?policy HTTP/1.1
Host: mybucket.s3.acme.net
Date: Mon, 25 Feb 2019 17:50:00 GMT
Authorization: authorization string
```

Sample Response:

```
HTTP/1.1 200 OK
Date: Mon, 25 Feb 2019 17:50:00 GMT
x-amz-request-id: 000001482031D84B
x-amz-id-2: 00005360851D626F
Content-Type: text/plain; charset=UTF-8
Transfer-Encoding: chunked
Connection: close
Server: DDN S3
{
"Version":"2008-10-17",
"Id":"policy1",
"Statement" : [
    {
        "Effect":"Allow",
        "Sid":"1",
        "Principal" : {
            "DDN":["939d06a45e38c3482e128919ae52ea09924baa1f"]
         },
         "Action":["s3:PutObject"],
         "Resource":"bucket1"
    }
 ]
}
```

3.3.9 HEAD Bucket

Description

The `HEAD Bucket` operation determines if a bucket exists and the user has permission to access the bucket.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --head "http://bucketname.s3.acme.net"
```

Syntax

```
HEAD / HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
```

Request Headers

Uses only the request headers that are common to all operations (see Common Headers).

Responses

Response Headers

Uses the response headers that are common to most responses (see Common Headers).

Examples

Sample Request:

```
HEAD / HTTP/1.1
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
Host: mybucket.s3.acme.net
```

Sample Response:

```
HTTP/1.1 200 OK
x-amz-id-2: 1C7B1231CB0E3939
x-amz-request-id: 32FE2CEB32F5EE25
Date: Wed, 12 July 2014 17:50:00 GMT
Server: DDN S3
```

3.3.10 List Multipart Uploads

Description

This operation lists multipart uploads that have been initiated, but have not yet been completed via the **`Initiate Multipart Upload`** request.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID "http://bucketname.s3.acme.net?uploads"
```

Syntax

```
GET /?uploads HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
```

Request Headers

Uses only the request headers that are common to all operations (see Common Headers).

Request Parameters

Uses the same request parameters as the Amazon S3. Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for the list of **`List Multipart Uploads`** request parameters.

Responses

Response Headers

Uses the response headers that are common to most responses (see Common Headers).

Response Elements

Uses the same response elements as the Amazon. Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for the list of **`List Multipart Uploads`** response elements.

Response Elements Not Supported

- **`DisplayName`** (Universal Unique Identifier, *UUID*, is displayed instead)

Examples

Sample Request:
By specifying the **`max-uploads=3 parameter`**, the example S3 call is requesting a maximum of 3 uploads to be returned.

```
GET /?uploads&max-uploads=3 HTTP/1.1
Host: mybucket.s3.acme.net
Date: Mon, 1 Nov 2010 20:34:56 GMT
Authorization: authorization string
```

Sample Request:

```
HTTP/1.1 200 OK
 Date: Fri, 29 Aug 2014 06:05:40 GMT
 x-amz-request-id: 00000148205E8CEC
 x-amz-id-2: 0000560AAC4FB6E0
 Content-Type: application/xml;charset=UTF-8
 Transfer-Encoding: chunked
 Connection: close
 Server: DDN S3

<?xml version="1.0" encoding="UTF-8"?>
<ListMultipartUploadsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Bucket>mpart</Bucket>
  <KeyMarker/>
  <UploadIdMarker/>
  <NextKeyMarker>my-movie.avi</NextKeyMarker>
<NextUploadIdMarker>00000148205d44e700001a04974e4341</NextUploadIdMarker>
  <MaxUploads>3</MaxUploads>
  <IsTruncated>false</IsTruncated>
  <Upload>
    <Key>mpObject1</Key>
    <UploadId>00000148205c888f000019f95d6f5abc</UploadId>
    <Initiator>
      <ID>ff2ae2c9a047cec671c0561349d4015961ac9eea</ID>
      <DisplayName>ff2ae2c9a047cec671c0561349d4015961ac9eea </DisplayName>
    </Initiator>
    <Owner>
      <ID>ff2ae2c9a047cec671c0561349d4015961ac9eea</ID>
      <DisplayName>ff2ae2c9a047cec671c0561349d4015961ac9eea </DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
    <Initiated>2014-08-29T06:03:28.000Z</Initiated>
  </Upload>
  <Upload>
    <Key>my-divisor</Key>
    <UploadId>00000148205cfcbc000055f2d1f37ae3</UploadId>
    <Initiator>
      <ID>ff2ae2c9a047cec671c0561349d4015961ac9eea</ID>
      <DisplayName>ff2ae2c9a047cec671c0561349d4015961ac9eea </DisplayName>
    </Initiator>
    <Owner>
      <ID>ff2ae2c9a047cec671c0561349d4015961ac9eea</ID>
      <DisplayName>ff2ae2c9a047cec671c0561349d4015961ac9eea </DisplayName>
    </Owner>
   <StorageClass>STANDARD</StorageClass>
   <Initiated>2014-05-20T15:45:54.000Z</Initiated>
 </Upload>
</ListMultipartUploadsResult>
```

3.3.11 PUT Bucket

Description

The **`PUT Bucket`** operation creates a bucket.

To create a bucket, you must have a valid *Access Key* and *Secret Key* to authenticate requests. For more information on bucket naming, see Bucket Naming Requirements (Section 2.4 on page 13). All bucket names must be unique across the whole S3 namespace.

NOTE:	*Bucket Regions* is an unsupported feature, and if specified together with the PUT Bucket request, it will cause the request to be refused.

All bucket creation requests require authentication. When creating buckets, you can optionally specify the permissions and grants for the users and groups. By default, only the bucket owner will be able to list the objects in the bucket or put the objects into the bucket. For more details about existing permissions, see Using ACLs (Section 3.5.2 on page 84).

NOTE:	Amazon allows you to create only 100 buckets per user. By default, EXAScaler Access S3 allows you to create 2000 buckets per user credential (configurable option).

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --createBucket -- "http://bucketname.s3.acme.net"
```

Syntax

```
PUT / HTTP/1.1
Host: bucketname.s3.acme.net
Content-Length: length
Date: date
Authorization: authorization string
```

Request Headers

Uses only the request headers that are common to all operations (see Common Headers).

Similar to the Amazon S3 API, the EXAScaler Access S3 API supports a set of predefined ACLs, known as canned ACLs. Each canned ACL has a predefined set of grantees and permissions. For more information about the permissions, see Using ACLs (Section 3.5.2 on page 84).

Request Elements Not Supported

- CreateBucketConfiguration
- LocationConstraint

Responses

Response Elements:

This implementation of the operation does not return response elements.

Examples

Sample Request:

```
PUT / HTTP/1.1
Host: mybucket.s3.acme.net
```

```
Content-Length: 0
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
```

Sample Request:

```
HTTP/1.1 200 OK
x-amz-id-2: 1C7B1231CB0E3939
x-amz-request-id: 236A8905248E5A01
Date: Wed, 12 July 2014 17:50:00 GMT
Content-Length: 0
Connection: close
Server: DDN S3
```

3.3.11.1 Associate Your Directory in the File System with Your S3 Bucket

When you create the bucket, specify a header as: **`x-ddn-specified-directory`** and give your file system path where you want to store all the objects of this bucket.

The path must exist, and the user must have write-permission in the given directory. The system validates user's permission depending on File system type configured and by using Filesystem user Id associated with the S3 credentials.

This given bucket specific file system path is directly used in the system for creating any path of object file in the file system. So, it should be an independent accessible path with having proper file separators. This path should be accessible to S3 nodes and for that this path should be under any mounted file system among multiple file systems which are defined in S3 Data Services configuration.

For more information on Creating S3 Buckets on Existing EXAScaler File System Directories, See *S3 Data Services Administration Guide*.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --createBucket -- "http://bucketname.s3.acme.net" -H
'x-ddn-specified-directory: /mnt/lustre/user1_dir'
```

Or

```
./s3curl.pl --id=ddnID --createBucket -- -v "http://10.44.2.162/bucket1" -H
'x-ddn-specified-directory: /mnt/lustre/user1_dir'
```

Sample Request:

```
PUT / HTTP/1.1
Host: mybucket.s3.acme.net
x-ddn-specified-directory: /mnt/lustre/user1_dir
Content-Length: length
Date: date
Authorization: authorization string
```

Sample Response:

x-ddn-specified-directory, this header will not be included in the GET/PUT Bucket and HEAD Bucket responses.

```
HTTP/1.1 200 OK
Date: date
Server: DDN S3
```

Error Conditions:

In following conditions you will get the 400 Invalid Argument with an error message:

- Directory does not exist or path is invalid or not accessible.
- Requester does not have write permission.

Error Message:

```
HTTP/1.1 400 Bad Request
< x-amz-request-id: 22B6BDD50BF6CF85
< x-amz-id-2: 22CB756AF5FD5857
< Date: Tue, 24 Mar 2020 09:54:16 GMT
< Server: DDN S3
< Content-Type: application/xml
< Connection: close
<
* Closing connection #0
<?xml version="1.0" encoding="UTF-8"?><Error><Code>InvalidArgument</Code><Message>Invalid bucket
specific Filesystem path(x-ddn-specified-directory). User does not have write permission or invalid
path</Message><ArgumentName>x-ddn-specified-directory</ArgumentName><ArgumentValue>/mnt/lustre/walter8
8</ArgumentValue><RequestId>22B6BDD50BF6CF85</RequestId><HostId>0000000000000000</HostId></Error>
```

3.3.12 PUT BucketSync

Description

The BucketSync scans the file system and brings the S3DS namespace into sync with the file system. BucketSync is implemented by a custom DDN extension to the Put Bucket API. Only one BucketSync operation can be active for a bucket at a time.

By default, all three synchronization operations (write, update, delete) are executed when BucketSync is called. The user can specify a subset of these operations to be executed in one of the following ways:

- Specify the sync operations to perform in the request body
- Specify the sync operations to perform using request header

Requests

s3curl Command Example:

```
s3curl.pl --id=ID -put=emptyPayload -- "http://bucketname.s3.acme.net?sync" " -H
"x-ddn-bucket-sync-ops:WRITE,UPDATE,DELETE"
```

Syntax

The following request shows the syntax for sending the BucketSync in the request body:

```
PUT /?sync
HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
<?xml version="1.0" encoding="UTF-8"?>
<operation>
    <operationTypes>WRITE,UPDATE,DELETE</operationTypes>
</operation>
```

Request Headers

Uses all the request headers that are common to all operations (see Common Headers). This operation uses only one request header (optional) in addition to the request headers common to most requests.

Optional operation types header could be included in the request:

- **`x-ddn-bucket-sync-ops`**
  Any combination of BucketSync operation types (WRITE/UPDATE/DELETE) in a comma separated string or any single BucketSync operation type can be specified in this header which user wants to perform in this BucketSync request.

Responses

Response Headers

Uses the response headers that are common to most responses (see Common Headers).

Response Elements

Name	Description
OperationType	Type of operation, which will always be “BucketSync”
OperationId	One ID is always assigned to each BucketSync operation
OperationStatus	Current status of the BucketSync operation
OperationMessage	Some details of the BucketSync operation
LastUpdatedDate	Last updated timestamp of this operation
CreatedDate	Creation time of this BucketSync operations

Examples

Sample Request:

In this request bucket "mybucket" would be sync with File System content directory associated with this bucket. The payload is passed with all the three BucektSync operations so it would start syncing for all these operations of BucketSync.

```
PUT /?sync HTTP/1.1
Host: mybucket.s3.acme.net
Date: Fri, 29 Aug 2014 05:50:47 +0000
Authorization: authorization string Content-Length: 118
<?xml version="1.0" encoding="UTF-8"?>
<operation>
<operationTypes>WRITE,UPDATE,DELETE</operationTypes>
</operation>
```

Sample Response:

```
Date: Tue, 16 Jun 2020 09:11:25 +0000
 Authorization: AWS AKIA49EBBXA4A39536YL:kBPXalPSuscdAsWjbV8Yi2X6t8Q=
 Content-Length: 1
 HTTP/1.1 200 OK
 x-amz-request-id: 3F422296BC65C645
 x-amz-id-2: 3F70A09422A5883A
 Date: Tue, 16 Jun 2020 09:11:26 GMT
 Server: DDN S3
 Content-Type: application/xml;charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
<!--Bucket Syncup Started, Please check the status of your operation using the operation Id-->
<Operation xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <OperationType>BUCKETSYNC</OperationType>
  <OperationId>BUCKETSYNC_ mybucket</OperationId>
  <OperationStatus>IN_PROGRESS</OperationStatus>
  <OperationMessage>Syncup started by: 44f1319400b95b3030820000b6c4389fe80363bc at: Tue Jun 16 09:11:26
UTC 2020 with Operations: [UPDATE, DELETE, WRITE]</OperationMessage>
  <LastUpdatedDate>Tue Jun 16 09:11:26 UTC 2020</LastUpdatedDate>
  <CreatedDate>Tue Jun 16 09:11:26 UTC 2020</CreatedDate>
</Operation>
```

For more information on Namespace Synchronization and BucketSync, please refer to the *S3 Data Services Administration Guide* (System Administration - S3DS Namespace Synchronization).

3.3.13 PUT Bucket ACL

Description

The PUT Bucket ACL operation sets the permissions on an existing bucket using the specified access control lists (ACL). See ACL Settings Supported by EXAScaler Access S3 (Section 3.5 on page 82) for more information on the ACLs.

Set the permissions in one of the following ways:

- Specify the ACL in the request body
- Specify permissions using request headers

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --createBucket -- "http://bucketname.s3.acme.net?acl" -H "acl-permission"
```

Syntax

The following request shows the syntax for sending the ACL in the request body:

```
PUT /?acl HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string

<AccessControlPolicy>
   <Owner>
      <ID>ID</ID>
      <DisplayName>DisplayName</DisplayName>
   </Owner>
   <AccessControlList>
   <Grant>
       <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema- instance" xsi:type="CanonicalUser">
          <ID>ID</ID>
          <DisplayName>DisplayName</DisplayName>
       </Grantee>
       <permission>Permission</Permission>
   </Grant>
   </AccessControlList>
</AccessControlPolicy>
```

Request Headers

Uses only the request headers that are common to all operations (see Common Headers).
Similar to Amazon, EXAScaler Access S3 API supports a set of predefined ACLs, known as canned ACLs. Each canned ACL has a predefined set of grantees and permissions. See Using ACLs (Section 3.5.2 on page 84) for more information on canned ACLs.

Request Elements Not Supported

- **`DisplayName`** (Universal Unique Identifier, *UUID*, is displayed instead)

Responses

Response Headers

Uses the response headers that are common to most responses (see Common Headers).

Examples

Sample Request:

In this example, the request specifies the ACL in the body. In addition to granting full control to the bucket owner, the XML specifies the following grants:

- Grant an S3 credential, identified by **`CanonicalUserID`**, with the **`FULL_CONTROL`** permission.
- **`CanonicalUserIDs`** are automatically generated, when the administrator generates new user credentials. To list **`CannonicalUserIDs`**, run the **`accesskeyctl -l`** command, the output displays them in the UUID column. Refer to the *S3 Data Services Administration Guide* for more information about managing S3 credentials.
- Grant an S3 credentials, identified by AllUsers group with the **`READ`** permission on the bucket.
- Grant an S3 credentials, identified by **`CanonicalUserID`**, with the **`WRITE_ACP`** permission.

```
PUT /?acl HTTP/1.1
Host: mybucket.s3.acme.net
Date: Fri, 29 Aug 2014 05:50:47 +0000
Authorization: authorization string
Content-Length: 1182

<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.acme.net/doc/2007-07-07/">
  <Owner>
    <ID>ff2ae2c9a047cec671c0561349d4015961ac9eea</ID>
<DisplayName>ff2ae2c9a047cec671c0561349d4015961ac9eea</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="CanonicalUser">
        <ID>2aa8f890c06887bb0c79ddcd76c437a063e87ccd</ID>
        <DisplayName>2aa8f890c06887bb0c79ddcd76c437a063e87ccd</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"   xsi:type="CanonicalUser">
        <ID>3dee8826ce0447c23a4911dee7f896e4f1bdaf3b</ID>
        <DisplayName>3dee8826ce0447c23a4911dee7f896e4f1bdaf3b</DisplayName>
      </Grantee>
      <Permission>READ_ACP</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

The following request uses ACL-specific request headers to grant the **`READ`** permission on the *AllUsers* group:

```
PUT /?acl HTTP/1.1
Host: mybucket.s3.acme.net
x-amz-date: Sun, 29 Apr 2012 22:00:57 GMT
x-amz-grant-read: uri="http://acs.acme.net/groups/global/AllUsers"
Authorization: authorization string
```

Sample Response:

```
HTTP/1.1 200 OK
x-amz-id-2: 1C7B1231CB0E3939
x-amz-request-id: C651BC9B4E1BD401
Date: Wed, 12 July 2014 17:50:00 GMT
Content-Length: 0
Server: DDN S3
```

3.3.14 PUT Bucket versioning

Description

To set the versioning state of an existing bucket, use the *versioning* subresource in the PUT Bucket request. Only the bucket owner can set the versioning state.

Set the versioning state to one of the following values:

- Enabled - Enables versioning for the objects in the bucket. Objects uploaded to the bucket get a unique version ID.
- Suspended - Disables versioning for the objects in the bucket. Objects uploaded to the bucket get the version ID null.

A bucket can only be in enabled or suspended state, after the versioning is enabled on it. You cannot disable versioning on a bucket.

If you have never set the versioning state on a bucket, it has no versioning state.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --put <versioning-file> \ --http://bucketname.s3.acme.net?versioning
```

Example contents of *versioning-file*:

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Status>VersioningState</Status>
</VersioningConfiguration>
```

Syntax

```
PUT /?versioning HTTP/1.1
Host: bucketname.s3.acme.net
Content-Length: length
Date: date
Authorization: authorization string

<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Status>VersioningState</Status>
</VersioningConfiguration>
```

Request Elements

- Status - Sets the versioning state of the bucket
- VersioningConfiguration - Container for setting the versioning state

Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for the list of **`PUT Bucket versioning`** request elements.

Request Elements Not Supported

- **`MfaDelete`**

Responses

Response Headers
Uses the response headers that are common to most responses (see Common Headers).

Examples

Sample Request:
This request sets the versioning state to *Enabled* for the specified bucket.

```
PUT /?versioning HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 124

<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

The following request suspends versioning for the specified bucket:

```
PUT /?versioning HTTP/1.1
Host: bucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 124

<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Suspended</Status>
</VersioningConfiguration>
```

Sample Response:

```
HTTP/1.1 200 OK
x-amz-id-2: 1C7B1231CB0E3939
x-amz-request-id: 1C7ABE9FA870ADFE
Date: Wed, 12 July 2014 17:50:00 GMT
```

3.3.15 PUT Bucket Policy

Description

The PUT Bucket Policy operation sets the permissions on an existing bucket using the specified Policy. For more information on the Bucket Policy, See Bucket Policy Settings Supported by EXAScaler Access S3 (Section 3.6 on page 86). Set the permissions by specifying the Statement(s) in the JSON request body. Only the Bucket owner can PUT the bucket policy.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --put=policy1.json -- -v -- "http://bucketname.s3.acme.net?policy"
```

Syntax

The following request shows the syntax for sending the Policy in the request body:

```
PUT /?policy HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
```

Request Headers

Uses only the request headers that are common to all operations (see Common Headers).

Responses

Response Headers

Uses the response headers that are common to most responses (see Common Headers).

Examples

Sample Request:

In this example, the request specifies the policy in the JSON body. In addition to granting PUT and GET object permission to a DDN EXAScaler-S3 user by specifying it's canonical UserId as "939d06a45e38c3482e128919ae52ea09924baa1f".

```
PUT /?policy HTTP/1.1
Host: mybucket.s3.acme.net
x-amz-date: Mon, 25 Feb 2019 17:50:00 GMT
Authorization: authorization string
Content-Length: 244
Expect: 100-continue
{
"Version":"2008-10-17",
"Id":"policy1",
"Statement" : [
    {
        "Effect":"Allow",
        "Sid":"statement1_Allow",
        "Principal" : {
            "DDN":["939d06a45e38c3482e128919ae52ea09924baa1f"]
         },
         "Action":["s3:PutObject","s3:GetObject"],
         "Resource":"bucket1"
    }
 ]
}
```

Sample Response:

```
HTTP/1.1 100 Continue
HTTP/1.1 204 No Content
Date: Mon, 20 May 2019 10:12:49 GMT
Server: DDN S3
x-amz-request-id: 23EBD90CD4BABF07
x-amz-id-2: 23F695A9CA835A0E
```

3.4 Object Operations

The EXAScaler Access S3 API supports the majority of the AWS S3 object calls. See Table 2 for information on the unsupported Object operations.

This section provides a brief description of each operation. Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for more details about AWS S3 Object operations.

3.4.1 DELETE Object

Description

This implementation of **DELETE** operation removes the **null** version (if there is one) of an object. When versioning is *enabled* or *suspended*, a simple **DELETE** call cannot delete an object, a delete marker will be inserted in the bucket instead. This delete marker becomes the current version of the object. If the object deleted is a delete marker, EXAScaler Access S3 sets the response header, **x-amz-delete-marker**, to **true**.

Versioning

To permanently delete a specific version of an object, specify a version ID in the *versionId* subresource.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --delete "http://bucketname.s3.acme.net/objectname"
```

Syntax

```
DELETE /ObjectName HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Content-Length: length
Authorization: authorization string
```

Request Headers Not Supported

- **x-amz-mfa**

Responses

Response Headers

The **DELETE Object** call uses the same response headers (*x-amz-delete-marker* and *x-amz-version-id*) as Amazon. Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for the list of **DELETE Object** response headers.

Examples

Sample Request:

```
DELETE /my-second-image.jpg HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
```

Sample Response:

```
HTTP/1.1 204 NoContent
x-amz-id-2: 1C7B1231CB0E3939
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: OTIyMzM3MDU4MDM1MTYyNTk5MA==
Date: Wed, 12 July 2014 17:50:00 GMT
Content-Length: 0
Connection: close
Server: DDN S3
```

Sample Request Deleting a Specified Version of an Object:
The following request deletes the specified version of the object, my-second-image.jpg.

```
DELETE /my-second-image.jpg?versionId=OTIyMzM3MDU4MDM1MTYyNTk5MA== HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 0
```

Sample Response:

```
HTTP/1.1 204 NoContent
x-amz-id-2: 1C7B1231CB0E3939
x-amz-request-id: 0A49CE4060975EAC
Date: Wed, 12 July 2014 17:50:00 GMT
Content-Length: 0
Connection: close
Server: DDN S3
```

Sample Response if the Object Deleted is a Delete Marker:

```
HTTP/1.1 204 NoContent
x-amz-id-2: 1C7B1231CB0E3939
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: OTIyMzM3MDU4MDM1MTY0MzY5OA==
x-amz-delete-marker: true
Date: 12 July 2014 17:50:00 GMT
Content-Length: 0
Connection: close
Server: DDN S3
```

Related resources:

- Deleting Objects from Versioning-Suspended Buckets
- Deleting Object Versions

3.4.2 DELETE Multiple Objects

Description

The Delete Multiple Objects operation is a request, which deletes multiple objects from a bucket using a single HTTP request. It requires object keys in the request xml body accompanied with mandatory *Content-MD5* and *Content-Length* headers.

You need to provide the object key names, and optionally, version IDs in the XML (if the goal is to delete a specific version of the object from a versioning-enabled bucket).

The *Multi-Object Delete* operation supports two modes for the response, verbose and quiet. By default, the operation uses verbose mode in which the response includes the result of deletion of each key in the request. In quiet mode, the response includes only keys where the delete operation encounters an error.

For a successful deletion, the operation does not return any information about the delete in the response body. In cases where the object specified in the request is not found, EXAScaler Access S3 returns the result as deleted.

NOTE: The *Multi-Object* Delete operation requires a single query string parameter called **`delete`** to distinguish it from other bucket **`POST`** operations.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --post dmo.xml --calculateContentMd5 --
"http://bucketname.s3.acme.net?delete"
```

where *dmo.xml* is:

```
<?xml version="1.0" encoding="UTF-8"?>
<Delete>
   <Object>
        <Key>key_1</Key>
   </Object>
   <Object>
        <Key>key_2</Key>
   </Object>
</Delete>
```

Syntax

```
POST /?delete HTTP/1.1
Host: bucketname.s3.acme.net
Authorization: authorization string
Content-Length: Size
Content-MD5: MD5

<?xml version="1.0" encoding="UTF-8"?>
<Delete>
  <Object>
    <Key>Key</Key>
  </Object>
  <Object>
    <Key>Key</Key>
  </Object>
  ...
</Delete>
```

Request Headers

This operation uses the following request headers in addition to the request headers common to most requests:

- Content-MD5 - The base64-encoded 128-bit MD5 digest of the data. This header is required as a message integrity check to verify that the request body was not corrupted in transit.
- Content-Length - Length of the body according to RFC 2616.

Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for the list of **`DELETE Multiple Objects`** request headers.

Request Headers Not Supported

- x-amz-mfa

Request Elements

Uses the same request elements as the Amazon S3. Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for the list of **`DELETE Multiple Objects`** request elements.

Responses

Response Headers

Uses the response headers that are common to most responses (see Common Headers).

Response Elements

Uses the same response elements as the Amazon S3. Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for the list of **`DELETE Multiple Objects`** response elements.

Examples

Sample Request:

The following *Multi-Object Delete* request deletes two objects from bucket *mybucke*t.

```
POST /?delete HTTP/1.1
Host: mybucket.s3.acme.net
Authorization: authorization string
Content-Length: 125
Content-MD5: p5/WA/oEr30qrEEl21PAqw==

<?xml version="1.0" encoding="UTF-8"?>
<Delete>
   <Object>
        <Key>sample1.txt</Key>
   </Object>
   <Object>
        <Key>sample2.txt</Key>
   </Object>
</Delete>
```

Sample Response:

In this example, the response includes a **`DeleteResult`** element that contains a Deleted element for the item, which EXAScaler Access S3 successfully deleted and an Error element, which EXAScaler Access S3 did not remove because the user did not have the required permission.

```
HTTP/1.1 200 OK
x-amz-id-2: 1C7B1231CB0E3939
x-amz-request-id: A437B3B641629AEE
Date: Wed, 12 July 2014 17:50:00 GMT
Content-Type: application/xml
Server: DDN S3
Content-Length: 251

<?xml version="1.0" encoding="UTF-8"?>
<DeleteResult xmlns="http://mybucket.s3.acme.net/doc/2007-07-07/">
  <Deleted>
    <Key>sample1.txt</Key>
  </Deleted>
  <Error>
    <Key>sample2.txt</Key>
    <Code>AccessDenied</Code>
    <Message>Access Denied</Message>
  </Error>
</DeleteResult>
```

Deleting Objects from Versioning-Enabled Bucket:

If you delete an item from a versioning-enabled bucket, all versions of that object remain in the bucket; however, a delete marker is inserted. In this example, the response indicates that a delete marker was added.

```
HTTP/1.1 200 OK
x-amz-id-2: 1C7B1231CB0E3939
x-amz-request-id: A437B3B641629AEE
Date: Fri, 02 Dec 2011 01:53:42 GMT
Content-Type: application/xml
Server: DDN S3
Content-Length: 251

<?xml version="1.0" encoding="UTF-8"?>
<DeleteResult xmlns="http://mybucket.s3.acme.net/doc/2007-07-07/">
  <Deleted>
    <Key>sample1.txt</Key>
    <VersionID>OTIyMzM3MDU4MDEwNjc3MzAxMA==</VersionID>
  </Deleted>
</DeleteResult>
```

If the goal is to delete a specific version of the object from a versioning-enabled bucket, version ID should be present in the request:

```
POST /?delete HTTP/1.1
Host: mybucket.s3.acme.net
Authorization: authorization string
Content-Length: 125
Content-MD5: p5/WA/oEr30qrEEl21PAqw==

<?xml version="1.0" encoding="UTF-8"?>
<Delete>
   <Object>
     <Key>sample1.txt</Key>
     <VersionId>OTIyMzM3MDU4MDEwNjc3MzAxMA==</VersionId>
   </Object>
</Delete>
```

3.4.3 GET Object

Description

Use this implementation of **GET** call to download or retrieve objects from EXAScaler Access S3. The **GET Object** request requires the caller to have at least **READ** permission on the object.

Versioning

By default, the **GET** operation returns the current version of an object. Use the *versionId* subresource to request a different version of an object.

If the current version of the object is a delete marker, EXAScaler Access S3 behaves as if the object was deleted and returns **404** error.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID "http://bucketname.s3.acme.net/objectname"
```

Syntax:

```
GET /ObjectName HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
Range:bytes=byte_range
```

Request Headers Not Supported

EXAScaler Access S3 API currently does not support the server-side encryption feature; therefore, the following headers cannot be used with this operation:

- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key
- x-amz-server-side-encryption-customer-key-MD5

Responses

Response Headers Not Supported

- x-amz-expiration
- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key-MD5
- x-amz-restore
- x-amz-website-redirect-location

Examples

Sample Request:

```
GET /my-first-image.jpg HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
Range: bytes=10-20
```

Sample Response:

```
HTTP/1.1 206 Partial Content
Date: Wed, 12 July 2014 17:50:00 GMT
x-amz-request-id: 00000148206D3FCC
```

```
x-amz-id-2: 000056EA7F48644B
ETag: "86506b43dbfb7422326698d036cb308a"
Content-Type: text/plain
Last-Modified: Wed, 12 July 2014 17:50:00 GMT
Accept-Ranges: bytes
Content-Range: bytes 10-20/1182
Content-Length: 11
Connection: close
Server: DDN S3

[11 bytes of object data]
```

In cases where the **`Range`** header is not defined, the response status will be **`200 OK`** and the full content of object will be returned.

Sample Request Getting a Specified Version of an Object:
The following request returns the specified version of an object.

```
GET /myObject?versionId=OTIyMzM3MDU4MDEwNjc3MzAxMA== HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
```

Sample Response to a Versioned Object GET Request:

```
HTTP/1.1 200 OK
x-amz-id-2: eftixk72aD6Ap54OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC148832E5
Date: Wed, 12 July 2014 17:50:00 GMT
Last-Modified: Wed, 12 July 2014 14:00:00 GMT
x-amz-version-id: OTIyMzM3MDU4MDEwNjc3MzAxMA==
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 434234
Content-Type: text/plain
Connection: close
Server: DDN S3
[434234 bytes of object data]
```

Limitation

Although the HTTP RFC 2616 specification allows multiple byte-ranges, EXAScaler Access S3 API as well as AWS REST S3 API supports only a single range of bytes (For example, **`Range:bytes=100-200`**).

In cases where a *Range* with multiple bytes (for example, **`Range:bytes=100-200,300-400`**) is requested from the S3 servers, the EXAScaler Access S3 (or AWS) servers will not recognize the Range header and will return the full object.

NOTE:	When a multiple byte-range request on an object, with size smaller than 8 kilobytes, is passed via Apache (for example, when client uses SSL connection via Apache), Apache will cache that object and produce the correct multiple byte-range content rather than returning the full object.

3.4.4 HEAD Object

Description

The **`HEAD Object`** operation retrieves metadata from an object without returning the data of the object. The user must have the **`READ`** permission to access to the object.

Versioning

The **`HEAD`** operation retrieves metadata from the current version of an object. To retrieve metadata from a different version, use the **`versionId`** subresource.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --head "http://bucketname.s3.acme.net/objectname"
```

Syntax:

```
HEAD /ObjectName HTTP/1.1
Host: bucketname.s3.acme.net
Authorization: authorization string
Date: date
```

Request Headers

Similar to Amazon S3, this operation uses common and additional request headers (see Common Headers).

Request Headers Not Supported

- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key
- x-amz-server-side-encryption-customer-key-MD5

Responses

Response Headers Not Supported

- x-amz-expiration
- x-amz-missing-meta
- x-amz-restore
- x-amz-server-side-encryption

Response Elements

This operation does not return any response elements.

Examples

Sample Request:

```
HEAD /my-image.jpg HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
```

Sample Response:

```
HTTP/1.1 200 OK
x-amz-id-2: ef8yU9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC143432E5
Date: Wed, 12 July 2014 17:50:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 434234
```

```
Content-Type: text/plain
Connection: close
Server: DDN S3
```

Getting Metadata from a Specified Object Version Sample Request:
The following request returns the metadata of the specified version of an object my-cat.jpg.

```
HEAD /my-cat.jpg?versionId= OTIyMzM3MDU3ODMwNzc3Nzk5NQ== HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
```

3.4.5 GET Object ACL

Description

This implementation of the **GET** operation retrieves the ACL (the ownership and permissions information) on the given object. The caller must have **READ_ACL** permission on the object (or be the owner of the object).

Versioning

By default, this call returns ACL information about the current version of an object. To return ACL information about a different version, use the **versionId** subresource.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID "http://bucketname.s3.acme.net/objectname?acl"
```

Syntax

```
GET /ObjectName?acl HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
```

Request Headers

Uses only the request headers that are common to all operations (see Common Headers).

Responses

Response Headers

Uses the response headers that are common to most responses (see Common Headers).

Response Elements

Uses the same response elements as the Amazon. Refer to the *Amazon Simple Storage Service API Reference, API version 2006-03-01* for the list of **GET Object ACL** response elements.

Response Elements Not Supported

- **DisplayName** (Universal Unique Identifier, *UUID*, is displayed instead)

Examples

Sample Request:

```
GET /my-image.jpg?acl HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
```

Sample Response:

```
HTTP/1.1 200 OK
Date: Wed, 12 July 2014 17:50:00 GMT
x-amz-request-id: 0000014820747248
x-amz-id-2: 00005758C74F804A
Content-Type: application/xml;charset=UTF-8
Transfer-Encoding: chunked
Connection: close
Server: DDN S3

<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>ff2ae2c9a047cec671c0561349d4015961ac9eea</ID>
    <DisplayName>ff2ae2c9a047cec671c0561349d4015961ac9eea   </DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
     <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-    instance" xsi:type="CanonicalUser">
      <ID>ff2ae2c9a047cec671c0561349d4015961ac9eea</ID>
      <DisplayName>ff2ae2c9a047cec671c0561349d4015961ac9eea   </DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Sample Request of getting the ACL of the Specific Version of an Object:
The following request returns the ACL of the specified version of the object, my-image.jpg.

```
GET /my-image.jpg?versionId=OTIyMzM3MDU4MDEwNjc3MzAxMA==&acl HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
```

Sample Response of getting the ACL of the Specific Version of an Object:

```
HTTP/1.1 200 OK
x-amz-id-2: 1C7B1231CB0E3939
x-amz-request-id: 318BC8BC148832E5
Date: Wed, 12 July 2014 17:50:00 GMT
Last-Modified: Wed, 12 July 2014 13:30:00 GMT
x-amz-version-id: OTIyMzM3MDU4MDEwNjc3MzAxMA==
Content-Length: 124
Content-Type: text/plain
Connection: close
Server: DDN S3

<AccessControlPolicy>
  <Owner>
    <ID>764d4b4c17e52f7a647f01f45ecc17303502b664 </ID>
    <DisplayName>764d4b4c17e52f7a647f01f45ecc17303502b664 </DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="CanonicalUser">
        <ID>764d4b4c17e52f7a647f01f45ecc17303502b664</ID>
<DisplayName>764d4b4c17e52f7a647f01f45ecc17303502b664</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

3.4.6 POST Object

NOTE: EXAScaler Access S3 v5.2.7 does not support Authentication Signature v4 for POST.

Description

The **`POST Object`** operation is another form of **`PUT`** object, but the **`POST Object`** operation adds an object to a specified bucket via HTML forms, where the parameters are passed as a form field.

The user must have the **`WRITE`** permission to access the bucket to add an object to it.

NOTE: EXAScaler Access S3 does not support policies with the POST Object call. As a result, the POST Object calls will be treated as anonymous (non-authenticated) S3 calls and can create objects in the buckets that have publicly writeable buckets.

Versioning

EXAScaler Access S3 returns this ID in the response using the **`x-amz-version-id`** response header. When versioning is suspended, the version ID is always **`null`**.

Requests

curl Command Example:

```
curl -v -F "key=key_name" -F "success_action_status=200" -F "file=file_location" -F
"submit=Upload" http://bucketname.s3.acme.net
```

Syntax

```
POST / HTTP/1.1
Host: bucketname.s3.acme.net
User-Agent: browser_data
Accept: file_types
Accept-Encoding: encoding
Accept-Charset: character_set
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: length

--9431149156168
Content-Disposition: form-data; name="key"

acl
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

success_redirect
--9431149156168
Content-Disposition: form-data; name="Content-Type"

content_type
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

uuid
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

metadata
```

```
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg

file_content
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to DDN S3
--9431149156168--
```

Form Fields Not Supported

EXAScaler Access S3 API POST operation does not support the following form fields:

- AccessKeyId
- policy
- x-amz-storage-class
- x-amz-security token
- x-amz-server side-encryption
- x-amz-website-redirect-location
- x-amz-server-side-encryption

Responses

Response Headers

Supports only **`x-amz-version-id`** header in addition to the request headers common to all operations (see Common Headers).

Response Headers Not Supported

- x-amz-expiration
- success_action_redirect, redirect
- x-amz-server-side-encryption
- x-amz-server-side-encryption-aws-kms-key-id
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key-MD5

Examples

Sample Request:

```
POST / HTTP/1.1
User-Agent: curl/7.19.7 ...
Host: mybucket.s3.acme.net
Accept: */*
Content-Length: 525262
Expect: 100-continue
Content-Type: multipart/form-data; boundary=----------------------------d36998ab3142
```

Sample Response:

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: 1C7B1231CB0E3939
x-amz-request-id: 0A49CE4060975EAC
Date: Wed, 12 July 2014 17:50:00 GMT
ETag: "1b2cf535f27731c974343645a3985328"
Content-Length: 0
Connection: close
Server: DDN S3
```

Sample Response when versioning is enabled on a bucket:

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: 1C7B1231CB0E3939
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: OTIyMzM3MDU3ODAyNzc5NTkyMw==
Date: Wed, 12 July 2014 17:50:00 GMT
ETag: "1b2cf535f27731c974343645a3985328"
Content-Length: 0
Connection: close
Server: DDN S3
```

3.4.7 PUT Object

Description

This implementation of **`PUT`** call adds an object to a bucket. This requires the WRITE permission on the bucket into which the object is to be uploaded.

Versioning

Similar to Amazon S3, EXAScaler Access S3 generates a unique version ID for the object being stored in a versioning-enabled bucket. The response returns this version ID using the **`x-amz-version-id`** response header. For versioning-suspended buckets, EXAScaler Access S3 uses **`null`** as the version ID for the object stored.

Reduced Redundancy Storage

EXAScaler Access S3 supports the *Standard* storage class only. The *Reduced Redundancy* storage class is not supported.

Access Permissions

The EXAScaler Access S3 API grants you the **`FULL_CONTROL`** permission by default when you upload an object. Alternatively, use the **`x-amz-acl`** (canned ACL) or **`x-amz-grant-{permission}`** request headers. For more details, see Using ACLs (Section 3.5.2 on page 84).

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --put=filename "http://bucketname.s3.acme.net/objectname"
```

Syntax

```
PUT /ObjectName HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
```

Request Headers

Uses the same request headers as Amazon in addition to the request headers common to all operations (see Common Headers).

Request Headers Not Supported

- x-amz-storage-class (This header supports only the Standard storage class)
- x-amz-website-redirect-location
- x-amz-server–side encryption

Responses

Response Headers

Supports only **`x-amz-version-id`** header in addition to the request headers common to all operations (see Common Headers).

Request Headers Not Supported

- x-amz-expiration
- x-amz-server-side
- `-encryption`
- x-amz-server-side-encryption-aws-kms-key-id
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key-MD5

Examples

Sample Request:

```
PUT /my-image.jpg HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 11434
Expect: 100-continue

[11434 bytes of object data]
```

Sample Response:

```
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
x-amz-id-2: riYPLdmOdAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiIOIxl+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
Date: Wed, 12 July 2014 17:50:00 GMT
ETag: "1b2cf535f27731c974343645a3985328"
Content-Length: 0
Connection: close
Server: DDN S3
```

Sample Response for Versioning-Enabled Bucket:
If the bucket has versioning-enabled, the response includes the **`x-amz-version-id`** header with a unique version ID.

```
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
Date: Wed, 12 July 2014 17:50:00 GMT
Server: DDN S3
x-amz-request-id: 773C7CC6982CFA15
x-amz-id-2: 773D553DFC410B95
ETag: "57c77aa974e02d407aa04ed51616ef9d"
x-amz-version-id: OTIyMzM3MDU3ODMwNzc3Nzk5NQ==
Content-Length: 0
Connection: close
Content-Type: text/plain; charset=UTF-8
```

Sample Response for Versioning-Suspended Bucket:

```
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
Date: Wed, 12 July 2014 17:50:00 GMT
Server: DDN S3
x-amz-request-id: 2A745C32983383AD
```

```
x-amz-id-2: 2A75350DBD4EBFE5
ETag: "57c77aa974e02d407aa04ed51616ef9d"
Content-Length: 0
Connection: close
Content-Type: text/plain; charset=UTF-8
```

Sample Response when User Quota was Reached:
If the user quota limit has been reached, the response contains the following error message.

```
HTTP/1.1 400 Bad Request
Date: Wed, 12 July 2014 17:50:00 GMT
Server: DDN S3
x-amz-request-id: 416775C4A9C489B4
x-amz-id-2: 4167DD989E575E8B
Content-Type: application/xml
Connection: close
Transfer-Encoding: chunked

<?xml version="1.0" encoding="UTF-8"?<Error>
  <Code>UUIDQuotaExceeded</Code>
  <Message>Your proposed upload exceeds the User quota limits.</Message>
  <BucketName>mybucket</BucketName>
  <QuotaLimit>User Quota limit is N bytes</QuotaLimit>
  <RequestId>416775C4A9C489B4</RequestId>
  <HostId>0000000000000000</HostId>
</Error>
```

Refer to the *S3 Data Services Administration Guide* for more information on user quotas.

3.4.8 PUT Object ACL

Description

This implementation of **`PUT`** uses the **`acl`** subresource to set the access control list (ACL) permissions for an object, existing in a bucket. To set the ACL of an object, the user must have the **`WRITE_ACP`** permission (or be the owner of the object).

Versioning

`PUT Object ACL` sets the ACL of the current version of an object. To set the ACL of a different version, use the **`versionId`** subresource.

Requests

s3curl Command Example:

`s3curl.pl --id=ID --put acl-file "http://bucketname.s3.acme.net/objectname?acl"`

Example contents of *acl-file*:

```
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>764d4b4c17e52f7a647f01f45ecc17303502b664</ID>
<DisplayName>764d4b4c17e52f7a647f01f45ecc17303502b664</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
        URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Syntax:

```
PUT /ObjectName?acl HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string

<AccessControlPolicy>
  <Owner>
    <ID>ID</ID>
    <DisplayName>ID</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xsi:type="CanonicalUser">
        <ID>ID</ID>
      <DisplayName>ID</DisplayName>
    </Grantee>
    <Permission>Permission</Permission>
    </Grant>
    ...
  </AccessControlList>
</AccessControlPolicy>
```

Request Headers

Uses canned and granted ACLs headers in addition to the request headers that are common to all operations:

- Canned ACL or permission
- Permission headers for each grantee

Each canned ACL has a predefined set of grantees and permissions. See Using ACLs (Section 3.5.2 on page 84) for more information on canned and granted ACLs.

Request Elements Not Supported

- **`DisplayName`** (Universal Unique Identifier, *UUID*, is displayed instead)

Grantee Values

Specifies the grantee, who has granted the access rights, for using the request elements. You can use the **`CanonicalUserID`**. For more details about the **`CanonicalUserID`**, see the example in PUT Bucket ACL (Section 3.3.13 on page 48).

Responses

Response Headers

This implementation of the operation can include the **`x-amz-version-id`** response header in addition to the response headers common to all responses (see Common Headers).

Examples

Sample Request:

```
PUT /my-image.jpg?acl HTTP/1.1
Host: mybucket.s3.acme.net
Date:  Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
Content-Length: 124

<AccessControlPolicy>
  <Owner>
    <ID>764d4b4c17e52f7a647f01f45ecc17303502b664</ID>
<DisplayName>764d4b4c17e52f7a647f01f45ecc17303502b664
    </DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
       <ID>8fc37d9a65e7e44c19dec403fb70710ea403af37</ID>
<DisplayName>8fc37d9a65e7e44c19dec403fb70710ea403af37
       </DisplayName>
    </Grantee>
    <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Sample Response:

```
HTTP/1.1 200 OK
x-amz-id-2: 1C7B1231CB0E3939
x-amz-version-id: 3/L4kqtJlcpXrof3vjVBH40Nr8X8gdRQBpUMLUo
Date: Wed, 12 July 2014 17:50:00 GMT
Last-Modified: Wed, 12 July 2014 13:00:00 GMT
Content-Length: 0
Connection: close
Server: DDN S3
```

3.4.9 PUT Object Copy

Description

This implementation of the **PUT** operation is a combination of **GET** and **PUT** calls. The **Put Object Copy** call copies the source object and creates it as a new object on the specified location. The user is required to have a READ access to the source object and **WRITE** access to the destination bucket.

Copy an object from one bucket to another by making a **PUT** request with the **x-amz-copy-source** header.

Access Permissions

EXAScaler Access S3 grants you **FULL_CONTROL** permission by default when you upload an object. Alternatively, you can apply the **x-amz-acl** (Canned ACL) or **x-amz-grant-{permission}** request headers, where **{permission}** is one of the **READ**, **WRITE**, **READ_ACP**, **WRITE_ACP** or **FULL_CONTROL** permissions. See Using ACLs (Section 3.5.2 on page 84) for more information on access permissions.

Versioning

The **x-amz-copy-source** header identifies the current version of an object to copy. To copy a different version, use the **versionId** subresource. If the current version is a delete marker, EXAScaler Access S3 behaves as if the object was deleted.

If versioning is enabled on the bucket, objects being copied receive a unique version ID, otherwise the version ID is always **null**.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --copySrc=source-bucket-name/source-object-name
"http://bucketname.s3.acme.net/objectname"
```

Syntax

```
PUT /destinationObject HTTP/1.1
Host: bucketname.s3.acme.net
x-amz-copy-source: /sourceBucket/sourceObject
x-amz-metadata-directive: metadata_directive
x-amz-copy-source-if-match: etag
x-amz-copy-source-if-none-match: etag
x-amz-copy-source-if-unmodified-since: time_stamp
x-amz-copy-source-if-modified-since: time_stamp
<request metadata>
Authorization: authorization string
Date: date
```

Request Headers Not Supported

- x-amz-website-redirect-location
- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key
- x-amz-server-side-encryption-customer-key-MD5

Response

Response Headers

Supports only **x-amz-copy-source-version-id** and **x-amz-version-id** headers in addition to the request headers common to all operations (see Common Headers).

Response Headers Not Supported

- x-amz-expiration
- x-amz-server-side-encryption
- x-amz-server-side-encryption-aws-kms-key-id
- x-amz-server-side-encryption-customer-key-MD5
- x-amz-server-side-encryption-customer-algorithm

Examples

Sample Request:

```
PUT /my-second-image.jpg HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
x-amz-copy-source: /bucket/my-image.jpg
Authorization: authorization string
```

Sample Response:

```
HTTP/1.1 200 OK
x-amz-id-2: 1C7B1231CB0E3939
x-amz-request-id: 318BC8BC148832E5
Date: Wed, 12 July 2014 17:50:00 GMT
Connection: close
Server: DDN S3

<CopyObjectResult>
  <LastModified>20-10-28T22:32:00</LastModified>
  <ETag>"9b2cf535f27731c974343645a3985328"</ETag>
</CopyObjectResult>
```

Sample Request of Copying Specified Object Version:
The following request copies the specified version of the key my-cat.jpg into the bucket mybucket and gives it the key my-new-cat.jpg.

```
PUT /my-new-cat.jpg HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
x-amz-copy-source: /mybucket/my-new-cat.jpg?versionId= OTIyMzM3MDU3ODAyNzc5NTkyMw==
Authorization: authorization string
```

Sample Response of Copying Object into Versioning Enabled Bucket:
This response shows that an object was copied into versioning-enabled bucket mybucket.

```
HTTP/1.1 200 OK
x-amz-id-2: 48ACF6EDE7B3EF3A
x-amz-request-id: 318BC8BC148832E5
x-amz-version-id: OTIyMzM3MDU3ODAyNzc5NTkyMw==
x-amz-copy-source-version-id: OTIyMzM3JDF3OTDyNzc5IUkyFw==
Date: Wed, 12 July 2014 17:50:00 GMT
Connection: close
Server: DDN S3

<?xml version="1.0" encoding="UTF-8"?>
<CopyObjectResult>
   <LastModified>2014-12-22T22:32:00</LastModified>
   <ETag>"9b2cf535f27731c974343645a3985328"</ETag>
</CopyObjectResult>
```

Sample Response of Copying Object into Versioning Suspended Bucket:
The following response shows that an object was copied into versioning-suspended bucket mybucket. The response does not contain the x-amz-version-id header.

```
HTTP/1.1 200 OK
x-amz-id-2: 48ACF6EDE7B3EF3A
x-amz-request-id: 318BC8BC148832E5
x-amz-copy-source-version-id: OTIyMzM3JDF3OTDyNzc5IUkyFw==
Date: Wed, 12 July 2014 17:50:00 GMT
Connection: close
Server: DDN S3

<?xml version="1.0" encoding="UTF-8"?>
<CopyObjectResult>
   <LastModified>2014-12-22T22:32:00</LastModified>
   <ETag>"9b2cf535f27731c974343645a3985328"</ETag>
</CopyObjectResult>
```

3.4.10 Initiate Multipart Upload

Multipart Uploading

EXAScaler Access S3 also supports multipart uploading. The multipart uploading allows S3 clients to upload large object as a set of parts, which are then composed into a single S3 object on the server-side. This is the only way S3 clients can store huge (multi-terabyte) objects.

Upload these parts in any order, independently without worrying about transmission errors. If transmission of any part fails, S3 clients can retransmit only the failed parts without affecting the other parts. The multipart uploading is a three-step process:

1. Initiate upload.
2. Add parts with part numbers.
3. Complete upload: collect parts and construct an object.
4. Concatenating parts in ascending order based on the part number. ETag normally represents the MD5 checksum for the stored objects data, and may be used to validate if the data-transfer happened correctly. However, this does not apply to the completed multipart objects, where the ETag is computed as MD5 checksum computation of part's ETag entries.

NOTE: Upload the parts in any order. In addition, upload the multiple parts simultaneously in parallel.

Description

This operation initiates a multipart upload and returns an upload ID, which is used to associate all the parts in the specific multipart upload.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --post -- http://bucketname.s3.acme.net/multipart-upload?uploads
```

Syntax

```
POST /ObjectName?uploads HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
```

Request Headers Not Supported

- x-amz-storageclass

- x-amz-website -redirect-location
- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption -customer-key
- x-amz-server-side-encryption-customer-key-MD5

Responses

Response Headers

Uses the response headers that are common to most responses (see Common Headers).

Response Headers Not Supported

- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key-MD5

Examples

Sample Request:

```
POST /my-movie.avi?uploads HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
```

Sample Response:

```
HTTP/1.1 200 OK
Date: Wed, 12 July 2014 17:50:00 GMT
x-amz-request-id: 00000148205D44DF
x-amz-id-2: 00001A0496D5EFF5
Content-Type: application/xml;charset=UTF-8
Transfer-Encoding: chunked
Connection: close
Server: DDN S3

<?xml version="1.0" encoding="UTF-8"?>
<InitiateMultipartUploadResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Bucket>multipart</Bucket>
  <Key>my-movie.avi</Key>
  <UploadId>00000148205d44e700001a04974e4341</UploadId>
</InitiateMultipartUploadResult>
```

3.4.11 Upload Part

Description

This operation uploads a part in a multipart upload. Each part must be at least 5 MB in size, except the last part. There is no size limit on the last part of your multipart upload.

NOTE:	You must initiate a multipart upload before uploading any part. EXAScaler Access S3 returns a unique identifier, the upload ID, that you must include in your upload part request.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --put=filename
"http://bucketname.s3.acme.net/multipart-upload?partNumber=part-number&uploadId=upload-id"
```

Syntax

```
PUT /ObjectName?partNumber=PartNumber&uploadId=UploadId HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Content-Length: size
Authorization: authorization string
```

Request Headers Not Supported

- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key
- x-amz-server-side-encryption-customer-key-MD5

Responses

Response Headers

Uses the response headers that are common to all operations (see Common Headers).

Response Headers Not Supported

- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key-MD5

Examples

Sample Request:

```
PUT /my-movie.m2ts?partNumber=1&uploadId=VCVsb2FkIElEIGZvciBlbZZpbm
cncyBteS1tb3ZpZS5tMnRzIHVwbG9hZR HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Content-Length: 10485760
Content-MD5: pUNXr/BjKK5G2UKvaRRrOA==
Authorization: authorization string
***part data omitted***
```

Sample Response:

```
HTTP/1.1 200 OK
x-amz-id-2: Vvag1LuByRx9e9pO4ZVKnJ2Qz7/C1NPcfTWAtRPfTaOFg==
x-amz-request-id: 656c76696e6727732072657175657374
Date: Wed, 12 July 2014 17:50:00 GMT
ETag: "b54357faf0632cce46e942fa68356b38"
Content-Length: 0
Connection: keep-alive
Server: DDN S3
```

3.4.12 Upload Part - Copy

Description

The **`Upload Part-Copy`** operation uploads a part by copying data from an existing object as data source.

NOTE:	You must initiate a multipart upload before uploading any part. EXAScaler Access S3 returns a unique identifier, the upload ID, that you must include in your upload part request.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --copySrc=source-bucket-name/source-object-name
"http://bucketname.s3.acme.net/multipart-upload?partNumber=part-number&uploadId=upload-id"
```

Syntax

```
PUT /ObjectName?partNumber=PartNumber&uploadId=UploadId HTTP/1.1
Host: bucketname.s3.acme.net
x-amz-copy-source: /source_bucket/sourceObject
x-amz-copy-source-range:bytes=first-last
x-amz-copy-source-if-match: etag
x-amz-copy-source-if-none-match: etag
x-amz-copy-source-if-unmodified-since: time_stamp
x-amz-copy-source-if-modified-since: time_stamp
Date: date
Authorization: authorization string
```

Request Headers

Similar to AWS S3, this operation uses common and additional request headers (see Common Headers).
For more information about additional headers for this operations, see the Amazon Simple Storage Service API Reference Guide, and search for Upload Part – Copy.

Request Headers Not Supported

- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key
- x-amz-server-side-encryption-customer-key-MD5
- x-amz-copy-source-server-side-encryption-customer-algorithm
- x-amz-copy-source-server-side-encryption -customer-key
- x-amz-copy-source-server-side-encryption-customer-key-MD5

Responses

Response Headers

This implementation of the operation can include **`x-amz-copy-source-version-id`** response header in addition to the response headers common to all responses (see Common Headers).

Response Headers Not Supported

- x-amz-server-side-encryption
- x-amz-server-side-encryption-aws-kms-key-id
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key-MD5

Examples

Sample Request:
The following **`PUT`** request uploads a part (part number 2) in a multipart upload. The request specifies a byte range from an existing object as the source of this upload. The request includes the upload ID that you get in response to your **`Initiate Multipart Upload`** request.

```
PUT /newobject?partNumber=2&uploadId=VCVsb2FkIElEIGZvciBlbZZpbm
cncyBteS1tb3ZpZS5tMnRzIHVwbG9hZR HTTP/1.1
Host: mybucket.s3.acme.net
Date: Mon, 11 Apr 2011 20:34:56 GMT
x-amz-copy-source: /source-bucket/sourceobject
x-amz-copy-source-range:bytes=500-6291456
Authorization: authorization string
```

Sample Response:
The response includes the **`ETag`** value. Retain this value to use when you send th**e `Complete Multipart Upload`** request.

```
HTTP/1.1 200 OK
x-amz-id-2: Vvag1LuByRx9e6j5Onimru9pO4ZVKnJ2Qz7/C1NPcfTWAtRPfTaOFg==
x-amz-request-id: 656c76696e6727732072657175657374
Date: Mon, 11 Apr 2011 20:34:56 GMT
Server: DDN S3

<CopyPartResult>
    <LastModified>2009-10-28T22:32:00</LastModified>
    <ETag>"9b2cf535f27731c974343645a3985328"</ETag>
</CopyPartResult>
```

3.4.13 Complete Multipart Upload

Description

This operation completes a multipart upload by assembling previously uploaded parts. You first initiate the multipart upload and then upload all parts using the Upload Part operation.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --post --
"http://bucketname.s3.acme.net/multipart-upload?uploadId=upload-id" --data "\
 <CompleteMultipartUpload>\
  <Part>\
    <PartNumber>1</PartNumber>\
    <ETag>"ETag"</ETag>\
  </Part>\
</CompleteMultipartUpload>"
```

Syntax

```
POST /ObjectName?uploadId=UploadId HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Content-Length: size
Authorization: authorization string

<CompleteMultipartUpload>
  <Part>
    <PartNumber>PartNumber</PartNumber>
    <ETag>ETag</ETag>
  </Part>
</CompleteMultipartUpload>
```

Request Headers

Uses the request headers that are common to most responses (see Common Headers).

Responses

Response Headers

Supports only **`x-amz-version-id`** header in addition to the request headers common to all operations (see Common Headers).

Response Headers Not Supported

- x-amz-expiration
- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm

Examples

Sample Request:
The following Complete Multipart Upload request specifies three parts in the **`<CompleteMultipartUpload>`** element.

```
POST /example-object?uploadId=AAAsb2FkIElEIGZvciBlbHZpbmcncyWeeS1tb3ZpZS5tMnRzIR
RwbG9hZA HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Content-Length: 391
Authorization: authorization string

<CompleteMultipartUpload>
  <Part>
    <PartNumber>1</PartNumber>
    <ETag>"a54357aff0632cce46d942af68356b38"</ETag>
  </Part>
  <Part>
    <PartNumber>2</PartNumber>
    <ETag>"0c78aef83f66abc1fa1e8477f296d394"</ETag>
  </Part>
  <Part>
    <PartNumber>3</PartNumber>
    <ETag>"acbd18db4cc2f85cedef654fccc4a4d8"</ETag>
  </Part>
</CompleteMultipartUpload>
```

Sample Response:
The following response indicates that an object was successfully assembled.

```
HTTP/1.1 200 OK
x-amz-id-2: Uuag1LuByRx9e6j5Onimru9pO4ZVKnJ2Qz7/C1NPcfTWAtRPfTaOFg==
x-amz-request-id: 656c76696e6727732072657175657374
Date: Mon, 1 Nov 2010 20:34:56 GMT
Connection: close
Server: DDN S3

<?xml version="1.0" encoding="UTF-8"?>
<CompleteMultipartUploadResult xmlns="http://s3.amazonaws.com
/doc/2006-03-01/">
  <Location>http://Example-Bucket.s3.acme.net/Example-Object</Location>
  <Bucket>Example-Bucket</Bucket>
  <Key>Example-Object</Key>
  <ETag>"3858f62230ac3c915f300c664312c11f-9"</ETag>
</CompleteMultipartUploadResult>
```

3.4.14 Abort Multipart Upload

Description

This operation aborts a multipart upload. After a multipart upload is aborted, no additional parts can be uploaded using that upload ID.

By default, the bucket owner and the initiator of the multipart upload are allowed to perform this action.

Requests

s3curl Command Example:

```
s3curl.pl --id=ID --delete
"http://bucketname.s3.acme.net/multipart-upload?uploadId=upload-id"
```

Syntax

```
DELETE /ObjectName?uploadId=UploadId HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
```

Request Headers

Uses only the request headers that are common to all operations (see Common Headers).

Responses

Response Headers

Uses only the response headers that are common to all operations (see Common Headers).

Examples

Sample Request:
The following request aborts a multipart upload identified by its upload ID.

```
DELETE /example-object?uploadId=VXBsb2FkIElEIGZvciBlbHZpbmcncyBteS1tb3ZpZS5tM
nRzIHVwbG9hZ HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
```

Sample Response:

```
HTTP/1.1 204 No Content
x-amz-id-2: 7350913B1BA3A205
x-amz-request-id: 996c76696e672773207265717565737 4
Date: Wed, 12 July 2014 17:50:00 GMT
Content-Length: 0
Connection: keep-alive
Server: DDN S3
```

3.4.15 List Parts

Description

This operation lists the parts that have been uploaded for a specific multipart upload. This operation must include the upload ID, which you obtain from the initiate multipart upload request. The default number of parts returned is 1,000 parts.

By default, the bucket owner has permission to list parts for any multipart upload to the bucket. The initiator of the multipart upload has the permission to list parts of the specific multipart upload.

Requests

s3curl Command Example

```
s3curl.pl --id=ID "http://bucketname.s3.acme.net/multipart-upload?uploadId=upload-id"
```

Syntax

```
GET /ObjectName?uploadId=UploadId HTTP/1.1
Host: bucketname.s3.acme.net
Date: date
Authorization: authorization string
```

Request Headers

Uses only the request headers that are common to all operations (see Common Headers).

Responses

Response Headers

Uses the response headers that are common to most responses (see Common Headers).

Response Elements

Uses the same response elements as the Amazon S3 API. Refer to the *Amazon Simple Storage Service API Reference Guide* for information on the response elements.

Response Elements Not Supported

- DisplayName

Examples

Sample Request:
In this example, the request lists the first two parts that follow part number 1, you will get parts 2 and 3 in the response.

```
GET
/example-object?max-parts=2&part-number-marker=1&uploadId=XXBsb2FkIElEIGZvciBlbHZpbmcncyVcdS1tb3ZpZS5t
MnRzEEEwbG9hZA& HTTP/1.1
Host: mybucket.s3.acme.net
Date: Wed, 12 July 2014 17:50:00 GMT
Authorization: authorization string
```

Sample Response:
This response shows that if more parts exist, the result is a truncated result and therefore the response will return an **`IsTruncated`** element with the value true. The response will also return the **`NextPartNumberMarker`** element with the value 3, which should be used for the value of the part-number-marker request query string parameter in the next **`List Parts`** request.

```
HTTP/1.1 200 OK
x-amz-id-2: 7350913B1BA3A205
x-amz-request-id: 656c76696e6727732072657175657374
Date: Wed, 12 July 2014 17:50:00 GMT
Content-Length: 985
Connection: keep-alive
Server: DDN S3

<?xml version="1.0" encoding="UTF-8"?>
<ListPartsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Bucket>example-bucket</Bucket>
  <Key>example-object</Key>
  <UploadId>XXBsb2pbmcncyVcdS1tb3ZpZS5tMnRzEEEwbG9hZA</UploadId>
    <Initiator>
      <ID>ff2ae2c9a047cec671c0561349d4015961ac9eea </ID>
<DisplayName>ff2ae2c9a047cec671c0561349d4015961ac9eea     </DisplayName>
    </Initiator>
   <Owner>
      <ID> ff2ae2c9a047cec671c0561349d4015961ac9eea </ID>
<DisplayName>ff2ae2c9a047cec671c0561349d4015961ac9eea     </DisplayName>
   </Owner>
   <StorageClass>STANDARD</StorageClass>
   <PartNumberMarker>1</PartNumberMarker>
   <NextPartNumberMarker>3</NextPartNumberMarker>
   <MaxParts>2</MaxParts>
   <IsTruncated>true</IsTruncated>
    <Part>
      <PartNumber>2</PartNumber>
    <LastModified>2010-11-10T20:48:34.000Z</LastModified>
    <ETag>"7778aef83f66abc1fa1e8477f296d394"</ETag>
    <Size>10485760</Size>
    </Part>
    <Part>
    <PartNumber>3</PartNumber>
    <LastModified>2010-11-10T20:48:33.000Z</LastModified>
    <ETag>"aaaa18db4cc2f85cedef654fccc4a4x8"</ETag>
      <Size>10485760</Size>
    </Part>
</ListPartsResult>
```

3.5 ACL Settings Supported by EXAScaler Access S3

The following section describes the S3 security model in general and provides precise details about which security and ACL settings are supported by EXAScaler Access S3 API. The behavior differences between EXAScaler Access S3 and Amazon S3 are also described.

3.5.1 ACL Settings

When creating a bucket, you can grant permissions to individual accounts or predefined groups. Every S3 object and bucket has security and ownership metadata associated with it (even if ACL was never specified with **`PUT Object`** or **`PUT Bucket`**, the security-setting will still be there, with the content signifying the object ownership is private).

The API Access Control List (ACL) is a way to represent the security settings (such as ownership or permissions) with the XML-formatted description.

The EXAScaler Access S3 ACL defines the access to buckets and objects. An ACL attached to each bucket and object, as internal metadata, contains the access information that defines precisely the type of access to the object/bucket the users or groups have. The users/groups are sometimes called Grantees, and the allowed operations are called Permissions (for example **`READ`**, **`WRITE`**, **`FULL_CONTROL`**).

When a request is received against a resource, EXAScaler Access S3 API checks the corresponding ACL to verify if the requester has the required access permissions.

The EXAScaler Access S3 API implementation of ACL support varies from the Amazon S3 format. For instance, the EXAScaler Access S3 API does not support resolving grantees via e-mail addresses; therefore, the **`AmazonCustomerByEmail`** grantees will not work. Additionally, the **`DisplayName`** will not be preserved or displayed as an e-mail address (instead, it will be displayed as the **`CanonicalUserID`**).

Applying ACL to Existing Bucket or ACL Parameter

The **`PUT Object`** or **`PUT Bucket`** request, with an ACL parameter and the XML description of the ACL grants, can change the permissions of the existing objects and buckets.

Resetting the permissions using the **`PUT Object ACL`** or **`PUT Bucket ACL`** completely replaces the existing **`ACL`** with a new one. You must have **`WRITE_ACL`** to do this. You must have **`READ_ACL`** permission to read the ACL of an existing bucket or object.

The ACL information is represented as an XML document. The ACL syntax is the same as the Amazon's, with exceptions that the e-mail grantee is not supported, and the **`DisplayName`** is not returned.

ACL Example

The following sample ACL on a bucket defines the owner of the resource and a set of grants. The bucket owner has **`FULL_CONTROL`** of the resource. This ACL shows how permissions are granted on a resource to one account, identified by CanonicalUserID, and to two of the predefined groups: AllUsers and AuthenticatedUsers.

```
<?xml version="1.0" encoding="UTF-8"?>

<AccessControlPolicy xmlns="http://s3.acme.net/doc/2006-03-01/">
  <Owner>
    <ID>CANONICAL_USER_ID</ID>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"   xsi:type="CanonicalUser">
        <ID>CANONICAL_USER_ID</ID>
      </Grantee>            <Permission>READ|WRITE|READ_ACL|WRITE_ACL|FULL_CONTROL</Permission>
    </Grant>
    <Grant>
    <Grantee xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
<Permission>READ|WRITE|READ_ACL|WRITE_ACL|FULL_CONTROL
      </Permission>
    </Grant>
    <Grant>
      <Grantee xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance  xsi:type="Group">
<URI>http://acs.amazonaws.com/groups/global/AuthenticatedUsers</URI>
      </Grantee>            <Permission>READ|WRITE|READ_ACL|WRITE_ACL|FULL_CONTROL
      </Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

3.5.2 Using ACLs

The canned ACLs are shorthand-predefined grants, normally passed in the header with the creation of Bucket, Object, or Multipart. There are two ways to grant these permissions:

- Specify a Canned ACL
- Specify the Access Permission Explicitly

Specify a Canned ACL

Table 6 lists the request headers used when specifying the canned ACL:

Table 6. Canned ACL Header

Name	Description
`x-amz-acl`	The canned ACL to apply to the bucket you are creating. For more information, go to Canned ACL in the Amazon Simple Storage Service Developer Guide. Type: String Valid Values: private | public-read | public-read-write | authenticated-read | bucket-owner-read | bucket-owner-full-control

Specify the Access Permission Explicitly

Grantees are the "entities" that can be granted as a certain access privilege on the EXAScaler Access S3 objects or buckets. In other words, it describes "who" gets the permission (grant).

To explicitly grant access permissions to specific EXAScaler Access S3 accounts or groups, use the headers described in Table 7. Each of these headers maps to specific permissions S3 supports in the ACL.

Table 7. Access Permission Headers

Name	Description
`x-amz-grant-read`	Allows grantee to list the objects in the bucket
`x-amz-grant-write`	Allows grantee to create, overwrite, and delete any object in the bucket
`x-amz-grant-read-acp`	Allows grantee to read the bucket ACL. Type: String
`x-amz-grant-write-acp`	Allows grantee to write the ACL for the applicable bucket
`x-amz-grant-full-control`	Allows grantee the READ, WRITE, READ_ACP, and WRITE_ACP permissions on the bucket

Grantee

A grantee can be a EXAScaler Access S3 credential or one of the predefined Cloud groups.

Grantee could be:

- Canonical Grantee
- Group Grantees

NOTE: The EXAScaler Access S3 API does not support the EmailAddress type of Grantee.

Group type of grantee normally represents one of the special EXAScaler Access S3 groups:

- AllUsersGroup
- AuthenticatedUsersGroup

NOTE: The EXAScaler Access S3 API does not support the LogDelivery Grantee.

Specify each grantee as type=value pair, where the type can be one of the followings:

- ID — if the value specified is the CanonicalUserID of an EXAScaler Access S3 account:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
  <ID>852b113e7a2f25102679df27bb0ae12b3f85be6</ID>
  <DisplayName>SomeDisplayName</DisplayName>
</Grantee>
```

- URI - if granting permission to a predefined EXAScaler Access S3 group (AllUsers in the example below):

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
  <URI> http://acs.amazonaws.com/groups/global/AllUsers</URI>
  <DisplayName>SomeDisplayName</DisplayName>
</Grantee>
```

Mapping of ACL from S3 Object to File System ACL

The EXAScaler Access S3 maps the S3 object ACL to File System ACL. This only supports the read and write ACLs assigned to predefined groups for now. If you do not want to manipulate the File System ACLs from S3 interface, then you can also disable this functionality.

For more information on ACL mappings, please refer to the *S3 Data Services Administration Guide* (S3 Data Services Features - S3 ACL to File System ACL Mapping).

3.5.3 Human Readable DisplayName ACL Information

The EXAScaler Access S3 API does not currently support storing some user information such as user names; therefore, the ACL settings retrieved from EXAScaler Access S3 API cannot set the DisplayName attribute correctly.

When the user asks for the ACL setting on a Bucket, the "human readable" **<DisplayName>** field is not returned:

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>4ed78f918c30570ca62ab0edec0d590fda7352dd</ID>
<DisplayName>4ed78f918c30570ca62ab0edec0d590fda7352dd</DisplayName>
  </Owner>
<AccessControlList>
  <Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"     xsi:type="CanonicalUser">
      <ID>4ed78f918c30570ca62ab0edec0d590fda7352dd</ID>
<DisplayName>4ed78f918c30570ca62ab0edec0d590fda7352dd
      </DisplayName>
    </Grantee>
    <Permission>FULL_CONTROL</Permission>
  </Grant>
</AccessControlList>
</AccessControlPolicy>
```

NOTE: The **<DisplayName>** fields have no other use in the EXAScaler Access S3 API, except to make the ACL more readable or usable by humans (no EXAScaler Access S3 API function takes **<DisplayName>** as an input parameter). In most calls that use the **<DisplayName>** as one of the XML elements, the **<DisplayName>** element can be safely skipped, or it can be included in XML with any value.

3.6 Bucket Policy Settings Supported by EXAScaler Access S3

The following section describes the S3 security model in general and provides precise details about which security and Bucket Policy settings are supported by EXAScaler Access S3 API. The behavior differences between EXAScaler Access S3 and Amazon S3 are also described.

3.6.1 Bucket Policy

You can add a bucket policy to grant other accounts access permissions for the objects in it. Bucket policies supplement, and in many cases, replace ACL-based access policies. Each bucket metadata can persist one policy document or bucket policy object with it.

Once a request comes to PUT Bucket Policy, the requested policy document get parsed and persisted with the bucket metadata. ACL applies to the individual object or bucket while Bucket Policy applies to all the objects under the bucket.

The EXAScaler Access S3 API implementation of bucket policy varies from the Amazon S3 format. For instance, the EXAScaler Access S3 API does not support IAM user (as principal) in the policy and supports condition element with a limited scope. EXAScaler Access S3 API, however, allows a bucket owner to specify any default user group as principal in the policy. In this release, the bucket policy feature covers the list bucket object operation as well as all the object operations. The PUT, GET, and DELETE bucket policies can only be performed by the bucket owner.

3.6.2 Apply Bucket Policy to an Existing Bucket

The PUT Bucket Policy request, with a policy parameter and the JSON description of the policy permissions, can change the permissions of the existing objects in the bucket. Resetting the permissions using the PUT Bucket Policy completely replaces the existing Policy with a new one. Requester must be the owner of the bucket to do this and only the bucket owner can GET or Delete any bucket policy.

The policy is represented as JSON document. The policy syntax is the same as the Amazon's, with the following exceptions:

- EXAScaler Access S3 API supports all the actions representing object operations plus the ListBucket action. You can grant or deny any permission on the object operations. The ListBucket action grants permission to list some or all of the objects in a bucket.
- Table 8 shows the operations that come under the bucket policy implementation of EXAScaler Access S3 API, along with the mapping of Actions (policy permission) to EXAScaler Access S3 operations.

Table 8. Operations and Policy permission mapping in Bucket Policy implementation

Action Names	S3 object Operations
`s3:ListBucket`	LIST BUCKET OBJECTS
`s3:PutObject`	PUT OBJECT, PUT OBJECT -COPY, INITIATE MULTI PART, UPLOAD PART, COMPLETE MULTI PART UPLOAD, POST OBJECT
`s3:GetObject`	GET OBJECT, HEAD OBJECT (versioned/non-versioned)

Table 8. Operations and Policy permission mapping in Bucket Policy implementation (Continued)

Action Names	S3 object Operations
`s3:DeleteObject`	DELETE OBJECT (versioned/non-versioned)
`s3:AbortMultipartUpload`	ABORT MULTI PART UPLOAD
`s3:ListMultipartUploadParts`	LIST MULTIPART UPLOAD
`s3:PutObjectAcl`	PUT OBJECT ACL (versioned/non-versioned)
`s3:GetObjectAcl`	GET OBJECT ACL (versioned/non-versioned)
`s3:* or * (ALL OBJECT OPERATIONS)`	FULL OBJECT CONTROL

- EXAScaler Access S3 API does not categorize the versioned and non versioned operations. Specifying any bucket policy permission will, by default, include the versioned and non versioned operations considered under that permission.
- EXAScaler Access S3 API does not have the IAM policy so only owner can PUT/GET/DELETE the bucket policy.
- EXAScaler Access S3 API supports condition element only with IP address condition operators.
- The owner (of bucket as well as object) cannot be denied for access to it's own object.
- EXAScaler Access S3 have only canonical user in our S3 so the principal would be the array of DDN canonical user Ids (UUID).
- Currently, there is no limit in our bucket policy document size and no quota validation on Put_Bucket_Policy.
- EXAScaler Access S3 API supports the default user group in bucket policy, also the public access can be specified by an asterisk (*) in principal element of bucket policy JSON document (same as AWS S3).
- In EXAScaler Access S3, there are three default group of users:

  a. AnonymousUser – Unsigned request that omits the Authentication header

  b. AllUsersGroup – Access permission to this group allows anyone to access the resource

  c. AuthenticatedUsersGroup – Access permission to this group allows any EXAScaler Access S3 account to access the resource

- EXAScaler Access S3 maintains the UUID for all the groups that can be accessed using the command line interface: **`s3ds accesskeys list`**

```
[root@s3ds1 ~]# s3ds accesskeys list
+---------------------------+---------+-------+------------------------------------------+-----
| accesskey                 | enabled | fsuid | secretkey                                | tag
+---------------------------+---------+-------+------------------------------------------+-----
| AnonymousUser             | True    | None  | RESERVED                                 | (re
| S3AuthenticatedUsersGroup | True    | None  | RESERVED                                 | (re
| S3AllUsersGroup           | True    | None  | RESERVED                                 | (re
| AKIAHZ2W16YSMO2J93UX      | True    | None  | lhmoVJNX5EvL45/9Auo4JahQt5+QXhytwZyb/ac2 | jsm
| AKIAZ91FNW517L84LS9C      | True    | None  | SZgoQN6bEF75aiFXXRVJpUcQTJtDriEhAO+JQ5SV | mjo
| AKIA9UDTB650PU2495IB      | True    | None  | /m2TlyhU0bdsENYNMvblPYlj4MJVD3e665Lug8Um | vle
+---------------------------+---------+-------+------------------------------------------+-----
```

- You can specify the UUID of these groups in the Principal element of JSON group.
- For public access, specify asterisk (*) or UUID of allUserGroup.
- After the implementation of bucket policy in DDN S3, the “complete multipart” and “upload multipart object-part” can only be performed by the multipart upload initiator. The “abort multipart” and “list multipart” operations can be performed by other users who may not be the initiator (if the bucket policy allows other users).
- The bucket owner can never be denied for any operation in a bucket policy (even if there is any Deny statement in the bucket policy for the bucket owner). The bucket owner can only change the bucket policy and only object operations are supported in this release so bucket owner denies itself is pointless.
- The resource element should be the bucket name only. The principal element can be specified by the key “DDN”, for example:

```
"Principal": {
   "DDN": [
     "e9af24fd2c28639b2b395ed9e27297a134132445"
   ]
 },
```

or

```
"Principal": {
  "DDN": "e9af24fd2c28639b2b395ed9e27297a134132445"
},
```

3.6.3 Bucket Policy Example

In the following example, the permission for the actions “s3:PutObject” and “s3:GetObject” are granted to the user “939d06a45e38c3482e128919ae52ea09924baa1f”. The mapping of actions (permission set) to operations is shown in Table 8.

```
{
 "Id": "policy1",
 "Version": "2008-10-17",
 "Statement": [
   {
     "Sid": "statement1_Allow",
     "Effect": "Allow",
     "Principal": {
       "DDN": [
         "939d06a45e38c3482e128919ae52ea09924baa1f"
       ]
     },
     "Action": [
       "s3:PutObject", "s3:GetObject"
     ],
     "Resource": "bucket1"
    }
   ]
 }
```

3.6.4 Bucket Policy JSON Document Language Overview

- ID – The Policy ID is a place holder. (Alphanumeric String)
- Version –The language Version is only a place holder for EXAScaler Access S3 API in the current release. For a valid bucket policy in EXAScaler Access S3 API either you do not specify it or specify a Version as "Version": "2008-10-17".
- Statement – Each statement defines an access rule for specific users and actions on the resource (bucket).
- SID – The Statement ID is a place holder. (Alphanumeric String)
- Effect – What the effect will be when the user requests the specific action—this can be either Allow or Deny. If you do not explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do in order to make sure that a user cannot access it, even if a different policy grants access.
- Principal – The account or user who is allowed access to the actions and resources in the statement. In a bucket policy, the principal is the authenticated user's canonical ID or any default user group's canonical Id who is the recipient of this permission.

- Actions – For each object resource, EXAScaler Access S3 supports a set of operations. You identify resource operations that you will allow (or deny) by using action sets (see Table 8). For example, the s3:GetObject permission allows the user permission to the EXAScaler Access S3 GET and HEAD object operation.
- Resources – Buckets for which you can allow or deny permissions. In a policy, EXAScaler Access S3 allows only the bucket name.
- Condition – Condition element decides when a policy is in effect. The Condition element is optional. In the condition block, you can specify any expression where you can use condition operators (for example, IpAddress) and condition key (for example, aws:SourceIp) and value (for example, IP ranges). These specified condition keys and values in the condition blocks would be matched against keys and values in the S3 request context.
  Syntax of a condition block:

```
"Condition" : {
   "CONDITION_OPERATOR" : { "CONDITION_KEY" : "CONDITION_VALUE" } ,
   "CONDITION_OPERATOR" : { "CONDITION_KEY" : "CONDITION_VALUE" }
   }
```

  OR you can use an array to put condition values:

```
"Condition" : {
   "CONDITION_OPERATOR" : { "CONDITION_KEY" :
["CONDITION_VALUE_1","CONDITION_VALUE_2".. . "CONDITION_VALUE_5"] }
   }
```

- Condition Operators – You can use the condition operators in the condition block, depending on the condition key you have specified. The condition operator is used in the condition element to match the condition key and value against the values in the S3 request context. The EXAScaler Access S3 supports only the IP address condition operators.
- IP Address Condition Operators – You can use the IP address condition operators (see table below) to construct condition elements which can restrict access based on comparing a condition key "aws:SourceIp" to an IPv4 address or range of IP addresses. You can use these with only the condition key "aws:SourceIp". The value specified with this key must be in the standard CIDR format (for example, 10.12.14.0/24).

Condition Operator	Description
IpAddress	The specified IP address or range
NotIpAddress	All IP addresses except the specified IP address or range

3.6.5 Condition Element in EXAScaler Access S3

Starting with S3 Data Services v1.3, EXAScaler Access S3 supports condition element in bucket policy:

- This condition element is supported with limited functionality.
- If required, you can add bucket policy with IP restrictions to each statement in the bucket policy.
- Condition key "aws:SourceIp" must be used to specify the IP address ranges in the condition block when condition operators "IpAddress" and/or "NotIpAddress" are used.
- Only IPv4 is supported and IP ranges must be in standard CIDR format (for example, 10.12.14.0/24). If you specify an IP address without the associated routing prefix, EXAScaler Access S3 uses the default prefix value of /32.

The following describes the condition keys and condition operators supported by EXAScaler Access S3:

- Condition Operators (IP Address Operators)

  - IpAddress — Use this operator when you want to allow the users to perform S3 operations only from a specified IP address or range.
  - NotIpAddress — Use this operator when you want to allow the users to perform S3 operations from all IP, except from the specified IP address or range.
- Condition Keys
  aws:SourceIp — This condition key works with IP address operators. You can use this key to compare the IP address that you have specified in the condition block of your bucket policy with the requester's IP address.

The condition block in EXAScaler Access S3 bucket policy works the same way as AWS S3, except for the following:

- Only two condition operators are supported—(1) IpAddress (2) NotIpAddress.
- Only condition key "aws:SourceIp" is supported in the current release.
- Only IPv4 is supported in the current release.
- IPv6 is not supported—Specifying an IPv6 address range in the condition block for the IPAddress or NotIpAddress will not work.
- Strict validation of condition key "aws:SourceIp"—You must specify the condition key as "aws:SourceIp" and this key is case-sensitive. Other values will NOT be accepted.
- EXAScaler Access S3 skips bucket policy authorization for the bucket owner—Any IP restriction applied using the condition block in the statement would not be able to deny the bucket owner's access.
- For IP restrictions to work in the condition block, EXAScaler Access S3 checks the HTTP header 'X-FORWARDED-FOR' to fetch the client IP from request context. If the client is using any proxy, the proxy server must supply the original client IP with 'X-FORWARDED-FOR' header.

Examples of Bucket Policy with Condition

1. Simple allow for a specific IP address range

The following bucket policy will allow the UUID “2ce26dd4ed36a5d9698ffe22ee55c77948284433” for “S3:GetObject” permission in resource “bucket1”, only if the request has been originated from IP address 10.12.13.139 or in the IP address range of 10.12.14.0 to 10.12.14.255.

```
{
  "Id": " bucket1_Policy ",
  "Version": "2008-10-17",
  "Statement": [
   {
      "Sid": "Allow_Ip",
      "Effect": "Allow",
      "Principal": {
        "DDN": ["2ce26dd4ed36a5d9698ffe22ee55c77948284433"]
      },
      "Action": [ "S3:GetObject"],
      "Resource": "bucket1",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": ["10.12.13.139/32", "10.12.14.0/32"]
        }
      }
    }
  ]
}
```

2. Simple deny for a specific IP address range

The following bucket policy will deny the UUID “2ce26dd4ed36a5d9698ffe22ee55c77948284433” for “S3:GetObject” permission in resource “bucket1”, only if the request has been originated from the IP address range of 10.12.14.0 to 10.12.14.255.

```
{
  "Id": " bucket1_Policy ",
  "Version": "2008-10-17",
  "Statement": [
   {
      "Sid": "Deny_Ip",
      "Effect": "Deny",
      "Principal": {
        "DDN": ["2ce26dd4ed36a5d9698ffe22ee55c77948284433"]
      },
      "Action": [ "S3:GetObject"],
      "Resource": "bucket1",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": ["10.12.14.0/32"]
        }
      }
    }
  ]
}
```

3. Allow all, except for a specific IP address

The following bucket policy will allow the UUID “2ce26dd4ed36a5d9698ffe22ee55c77948284433” for “S3:PutObject” permission in resource “bucket1” from all IP addresses except for 10.12.14.136.

```
{
  "Id": " bucket1_Policy ",
  "Version": "2008-10-17",
  "Statement": [
   {
      "Sid": "Allow_NotIp",
      "Effect": "Allow",
      "Principal": {
        "DDN": ["2ce26dd4ed36a5d9698ffe22ee55c77948284433"]
      },
      "Action": [ "S3:PutObject"],
      "Resource": "bucket1",
      "Condition": {
        "NotIpAddress": {
          "aws:SourceIp": ["10.12.14.136/32"]
        }
      }
    }
  ]
}
```

4. Deny for all, except for a specific IP address or range

The following bucket policy will deny user “2ce26dd4ed36a5d9698ffe22ee55c77948284433” for “S3:GetObjectAcl” permission in resource “bucket1”, except when the request has been originated from the IP address range of 10.12.14.0 to 10.12.14.255.

```
{
  "Id": " bucket1_Policy ",
  "Version": "2008-10-17",
  "Statement": [
   {
      "Sid": "Deny_NotIp",
      "Effect": "Deny",
      "Principal": {
        "DDN": ["2ce26dd4ed36a5d9698ffe22ee55c77948284433"]
      },
      "Action": [ "S3:GetObjectAcl"],
     "Resource": "bucket1",
      "Condition": {
        "NotIpAddress": {
          "aws:SourceIp": ["10.12.14.0/32"]
        }
      }
    }
  ]
}
```

5. Deny all users to perform any S3 object operation from any IP address, except for a specific IP address

The following bucket policy denies permissions to all users to perform any S3 operations on objects in the resource “bucket1” unless the request has been originated from IP address 10.12.14.138.

Note that since EXAScaler Access S3 skips bucket policy authorization for the bucket owner (see Condition Element in EXAScaler Access S3), bucket owner can still perform any operation from any IP address even after applying the below bucket policy.

Also, other than the bucket owner, only the users having permission with ACL can perform S3 object operations in the resource “bucket1” from IP address 10.12.14.138 after applying the below policy.

```
{
  "Id": " bucket1_Policy ",
  "Version": "2008-10-17",
  "Statement": [
   {
      "Sid": "Deny_AllNotIp",
      "Effect": "Deny",
      "Principal": { "DDN": ["*"] },
      "Action": [ "S3:*"],
      "Resource": "bucket1",
      "Condition": {
        "NotIpAddress": {
          "aws:SourceIp": ["10.12.14.138"]
        }
      }
    }
  ]
}
```

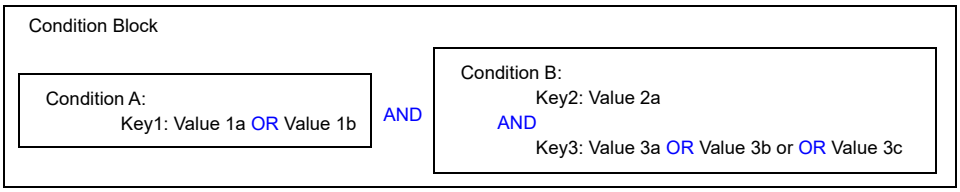
6. Multiple condition operators in a single condition

If you try to use multiple statements or conditions in a bucket policy, be careful and make sure you understand how it works.

In EXAScaler Access S3, the evaluation logic for conditions with multiple keys and/or values is the same as the AWS’s:

- For a single condition operator that includes multiple values for one key, that condition operator is evaluated using a logical OR (see condition A in Figure 2).
- For multiple condition operators in a policy or multiple keys attached to a single condition operator, the conditions are evaluated using a logical AND (see condition block and condition B in Figure 2).
- All conditions must resolve to true to trigger the desired Deny or Allow effect.

Figure 2. Evaluation Logic for Conditions with Multiple Keys or Values



The two examples below demonstrate how multiple statements and conditions are evaluated.

a. You can use both IP address operators in a condition element.
The following bucket policy will allow the user “2ce26dd4ed36a5d9698ffe22ee55c77948284433” for “S3GetObject” permission in the resource “bucket1”, only if the request has been originated from the IP address range of 10.12.14.0 to 10.12.14.255, except for IP address 10.12.14.138.

Note that if you specify the same IP address or range in both operators’ key value, no access will be given to the user “2ce26dd4ed36a5d9698ffe22ee55c77948284433” as the “NotIpAddress” operator’s key values would be applied first.

```
{
  "Id": "bucket1_Policy",
  "Version": "2008-10-17",
  "Statement": [
   {
      "Sid": "AllowNotIp_U2",
      "Effect": "Allow",
      "Principal": {
        "DDN": ["2ce26dd4ed36a5d9698ffe22ee55c77948284433"]
      },
      "Action": [ "S3:GetObject"],
      "Resource": "bucket1",
      "Condition": {
        "NotIpAddress": {
          "aws:SourceIp": ["10.12.14.138/32"]
        },
        "IpAddress": {
          "aws:SourceIp": ["10.12.14.0/32"]
        }
      }
  }
]
}
```

b. Once you allow all users without condition in a statement, you cannot deny any user for any specific IP by using an “Allow” effect with the “NotIpAddress” operator. You must use a “Deny” effect.

```
{
  "Id": "bucket1_Policy",
  "Version": "2008-10-17",
  "Statement": [
   {
      "Sid": "Allow_All",
      "Effect": "Allow",
      "Principal":{"DDN":["*"]},
      "Action": [ "S3:*"],
      "Resource": "bucket1"
   },
   {
      "Sid": "AllowNotIp_U2",
      "Effect": "Allow",
      "Principal": {
        "DDN": ["2ce26dd4ed36a5d9698ffe22ee55c77948284433"]
      },
      "Action": [ "S3:GetObject"],
      "Resource": "bucket1",
      "Condition": {
        "NotIpAddress": {
          "aws:SourceIp": ["10.12.13.138/32"]
        }
      }
   }
]
}
```

The above bucket policy will allow all users to perform any S3 object operation. No restrictions would be applied to the user “2ce26dd4ed36a5d9698ffe22ee55c77948284433” as the “NotIpAddress” operator does not give an explicit “Deny” here, and only not give an “Allow” to the user if the request has been originated from IP address 10.12.13.138. However, another statement “Allow_All” will give that permission.

3.7 Flow of Authorization with Bucket Policy and ACL

EXAScaler Access S3 is compatible with AWS S3 in the authorization process, except that there is no support for IAM user and IAM user policy in EXAScaler Access S3.

EXAScaler Access S3 authorization also works on least privilege principle and will not grant access if any deny is found with no explicit allow in all authorization methods.

Currently, EXAScaler Access S3 supports two authorization methods, Bucket Policy and ACL. If no method specifies an ALLOW, then the request will be denied by default. Only if no method specifies a DENY and one or more methods specify an ALLOW, the request will be allowed.

Contacting DDN Support

If you have questions or require assistance, contact DDN Support:

Web

Support Portal	https://community.ddn.com/login
Portal Assistance	webportal.support@ddn.com

Telephone

DDN Worldwide Directory	https://www.ddn.com/support/global-services-overview/

E-mail

Support E-mail	support@ddn.com

Bulletins

Support Bulletins	https://www.ddn.com/support/technical-support-bulletins
End-of-Life Notices	https://www.ddn.com/support/end-of-life-notices
Bulletin Subscription Request	support-tsb@ddn.com

Corporate Headquarters · 9351 Deering Avenue, Chatsworth, CA 91311 · ddn.com · +1.818.700.4000 · info@ddn.com