



S3 REST API supported operations and limitations

StorageGRID

NetApp

October 03, 2025

This PDF was generated from <https://docs.netapp.com/us-en/storagegrid-115/s3/authenticating-requests.html> on October 03, 2025. Always check docs.netapp.com for the latest.

Table of Contents

S3 REST API supported operations and limitations	1
Date handling	1
Common request headers	1
Common response headers	2
Authenticating requests	2
Using the HTTP Authorization header	2
Using query parameters	2
Operations on the service	2
Operations on buckets	3
Creating an S3 lifecycle configuration	11
Custom operations on buckets	16
Operations on objects	17
Using S3 Object Lock	22
Using server-side encryption	24
GET Object	26
HEAD Object	28
POST Object restore	30
PUT Object	32
PUT Object - Copy	36
Operations for multipart uploads	40
List Multipart Uploads	41
Initiate Multipart Upload	41
Upload Part	44
Upload Part - Copy	45
Complete Multipart Upload	46
Error responses	47
Supported S3 API error codes	47
StorageGRID custom error codes	49

S3 REST API supported operations and limitations

The StorageGRID system implements the Simple Storage Service API (API Version 2006-03-01) with support for most operations, and with some limitations. You need to understand the implementation details when you are integrating S3 REST API client applications.

The StorageGRID system supports both virtual hosted-style requests and path-style requests.

- [Authenticating requests](#)
- [Operations on the service](#)
- [Operations on buckets](#)
- [Custom operations on buckets](#)
- [Operations on objects](#)
- [Operations for multipart uploads](#)
- [Error responses](#)

Date handling

The StorageGRID implementation of the S3 REST API only supports valid HTTP date formats.

The StorageGRID system only supports valid HTTP date formats for any headers that accept date values. The time portion of the date can be specified in Greenwich Mean Time (GMT) format, or in Universal Coordinated Time (UTC) format with no time zone offset (+0000 must be specified). If you include the `x-amz-date` header in your request, it overrides any value specified in the `Date` request header. When using AWS Signature Version 4, the `x-amz-date` header must be present in the signed request because the date header is not supported.

Common request headers

The StorageGRID system supports common request headers defined by the *Simple Storage Service API Reference*, with one exception.

Request header	Implementation
Authorization	<p>Full support for AWS Signature Version 2</p> <p>Support for AWS Signature Version 4, with the following exceptions:</p> <ul style="list-style-type: none">• The SHA256 value is not calculated for the body of the request. The user-submitted value is accepted without validation, as if the value <code>UNSIGNED-PAYLOAD</code> had been provided for the <code>x-amz-content-sha256</code> header.

Request header	Implementation
x-amz-security-token	Not implemented. Returns XNotImplemented.

Common response headers

The StorageGRID system supports all of the common response headers defined by the *Simple Storage Service API Reference*, with one exception.

Response header	Implementation
x-amz-id-2	Not used

Related information

[Amazon Web Services \(AWS\) Documentation: Amazon Simple Storage Service API Reference](#)

Authenticating requests

The StorageGRID system supports both authenticated and anonymous access to objects using the S3 API.

The S3 API supports Signature version 2 and Signature version 4 for authenticating S3 API requests.

Authenticated requests must be signed using your access key ID and secret access key.

The StorageGRID system supports two authentication methods: the `HTTP Authorization header` and using query parameters.

Using the `HTTP Authorization header`

The `HTTP Authorization header` is used by all S3 API operations except Anonymous requests where permitted by the bucket policy. The `Authorization` header contains all of the required signing information to authenticate a request.

Using query parameters

You can use query parameters to add authentication information to a URL. This is known as presigning the URL, which can be used to grant temporary access to specific resources. Users with the presigned URL do not need to know the secret access key in order to access the resource, which enables you to provide third-party restricted access to a resource.

Operations on the service

The StorageGRID system supports the following operations on the service.

Operation	Implementation
GET Service	Implemented with all Amazon S3 REST API behavior.

Operation	Implementation
GET Storage Usage	The GET Storage Usage request tells you the total amount of storage in use by an account, and for each bucket associated with the account. This is an operation on the service with a path of / and a custom query parameter (?x-ntap-sg-usage) added.
OPTIONS /	Client applications can issue OPTIONS / requests to the S3 port on a Storage Node, without providing S3 authentication credentials, to determine whether the Storage Node is available. You can use this request for monitoring, or to allow external load balancers to identify when a Storage Node is down.

Related information

[GET Storage Usage request](#)

Operations on buckets

The StorageGRID system supports a maximum of 1,000 buckets for each S3 tenant account.

Bucket name restrictions follow the AWS US Standard region restrictions, but you should further restrict them to DNS naming conventions in order to support S3 virtual hosted-style requests.

[Amazon Web Services \(AWS\) Documentation: Bucket Restrictions and Limitations](#)

[Endpoint domain names for S3 request](#)

The GET Bucket (List Objects) and GET Bucket versions operations support StorageGRID consistency controls.

You can check whether updates to last access time are enabled or disabled for individual buckets.

The following table describes how StorageGRID implements S3 REST API bucket operations. To perform any of these operations, the necessary access credentials must be provided for the account.

Operation	Implementation
DELETE Bucket	Implemented with all Amazon S3 REST API behavior.
DELETE Bucket cors	This operation deletes the CORS configuration for the bucket.
DELETE Bucket encryption	This operation deletes the default encryption from the bucket. Existing encrypted objects remain encrypted, but any new objects added to the bucket are not encrypted.

Operation	Implementation
DELETE Bucket lifecycle	This operation deletes the lifecycle configuration from the bucket.
DELETE Bucket policy	This operation deletes the policy attached to the bucket.
DELETE Bucket replication	This operation deletes the replication configuration attached to the bucket.
DELETE Bucket tagging	This operation uses the tagging subresource to remove all tags from a bucket.
GET Bucket (List Objects), version 1 and version 2	<p>This operation returns some or all (up to 1,000) of the objects in a bucket. The Storage Class for objects can have either of two values, even if the object was ingested with the <code>REDUCED_REDUNDANCY</code> storage class option:</p> <ul style="list-style-type: none"> • <code>STANDARD</code>, which indicates the object is stored in a storage pool consisting of Storage Nodes. • <code>GLACIER</code>, which indicates that the object has been moved to the external bucket specified by the Cloud Storage Pool. <p>If the bucket contains large numbers of deleted keys that have the same prefix, the response might include some <code>CommonPrefixes</code> that do not contain keys.</p>
GET Bucket acl	This operation returns a positive response and the ID, <code>DisplayName</code> , and <code>Permission</code> of the bucket owner, indicating that the owner has full access to the bucket.
GET Bucket cors	This operation returns the <code>cors</code> configuration for the bucket.
GET Bucket encryption	This operation returns the default encryption configuration for the bucket.
GET Bucket lifecycle	This operation returns the lifecycle configuration for the bucket.
GET Bucket location	This operation returns the region that was set using the <code>LocationConstraint</code> element in the <code>PUT Bucket</code> request. If the bucket's region is <code>us-east-1</code> , an empty string is returned for the region.

Operation	Implementation
GET Bucket notification	This operation returns the notification configuration attached to the bucket.
GET Bucket Object versions	With READ access on a bucket, this operation with the <code>versions</code> subresource lists metadata of all of the versions of objects in the bucket.
GET Bucket policy	This operation returns the policy attached to the bucket.
GET Bucket replication	This operation returns the replication configuration attached to the bucket.
GET Bucket tagging	This operation uses the <code>tagging</code> subresource to return all tags for a bucket.
GET Bucket versioning	This implementation uses the <code>versioning</code> subresource to return the versioning state of a bucket. The versioning state returned indicates if the bucket is “Unversioned” or if the bucket is version “Enabled” or “Suspended.”
GET Object Lock Configuration	This operation determines if S3 Object Lock is enabled for a bucket. Using S3 Object Lock
HEAD Bucket	This operation determines if a bucket exists and you have permission to access it.

Operation	Implementation
PUT Bucket	<p>This operation creates a new bucket. By creating the bucket, you become the bucket owner.</p> <ul style="list-style-type: none"> Bucket names must comply with the following rules: <ul style="list-style-type: none"> Must be unique across each StorageGRID system (not just unique within the tenant account). Must be DNS compliant. Must contain at least 3 and no more than 63 characters. Can be a series of one or more labels, with adjacent labels separated by a period. Each label must start and end with a lowercase letter or a number and can only use lowercase letters, numbers, and hyphens. Must not look like a text-formatted IP address. Should not use periods in virtual hosted style requests. Periods will cause problems with server wildcard certificate verification. By default, buckets are created in the <code>us-east-1</code> region; however, you can use the <code>LocationConstraint</code> request element in the request body to specify a different region. When using the <code>LocationConstraint</code> element, you must specify the exact name of a region that has been defined using the Grid Manager or the Grid Management API. Contact your system administrator if you do not know the region name you should use. Note: An error will occur if your PUT Bucket request uses a region that has not been defined in StorageGRID. You can include the <code>x-amz-bucket-object-lock-enabled</code> request header to create a bucket with S3 Object Lock enabled. <p>You must enable S3 Object Lock when you create the bucket. You cannot add or disable S3 Object Lock after a bucket is created. S3 Object Lock requires bucket versioning, which is enabled automatically when you create the bucket.</p> <p>Using S3 Object Lock</p>

Operation	Implementation
PUT Bucket cors	<p>This operation sets the CORS configuration for a bucket so that the bucket can service cross-origin requests. Cross-origin resource sharing (CORS) is a security mechanism that allows client web applications in one domain to access resources in a different domain. For example, suppose you use an S3 bucket named <code>images</code> to store graphics. By setting the CORS configuration for the <code>images</code> bucket, you can allow the images in that bucket to be displayed on the website http://www.example.com.</p>
PUT Bucket encryption	<p>This operation sets the default encryption state of an existing bucket. When bucket-level encryption is enabled, any new objects added to the bucket are encrypted. StorageGRID supports server-side encryption with StorageGRID-managed keys. When specifying the server-side encryption configuration rule, set the <code>SSEAlgorithm</code> parameter to <code>AES256</code>, and do not use the <code>KMSMasterKeyID</code> parameter.</p> <p>Bucket default encryption configuration is ignored if the object upload request already specifies encryption (that is, if the request includes the <code>x-amz-server-side-encryption-*</code> request header).</p>

Operation	Implementation
PUT Bucket lifecycle	<p>This operation creates a new lifecycle configuration for the bucket or replaces an existing lifecycle configuration. StorageGRID supports up to 1,000 lifecycle rules in a lifecycle configuration. Each rule can include the following XML elements:</p> <ul style="list-style-type: none"> • Expiration (Days, Date) • NoncurrentVersionExpiration (NoncurrentDays) • Filter (Prefix, Tag) • Status • ID <p>StorageGRID does not support these actions:</p> <ul style="list-style-type: none"> • AbortIncompleteMultipartUpload • ExpiredObjectDeleteMarker • Transition <p>To understand how the Expiration action in a bucket lifecycle interacts with ILM placement instructions, see “How ILM operates throughout an object’s life” in the instructions for managing objects with information lifecycle management.</p> <p>Note: Bucket lifecycle configuration can be used with buckets that have S3 Object Lock enabled, but bucket lifecycle configuration is not supported for legacy Compliant buckets.</p>

Operation	Implementation
PUT Bucket notification	<p>This operation configures notifications for the bucket using the notification configuration XML included in the request body. You should be aware of the following implementation details:</p> <ul style="list-style-type: none"> StorageGRID supports Simple Notification Service (SNS) topics as destinations. Simple Queue Service (SQS) or Amazon Lambda endpoints are not supported. The destination for notifications must be specified as the URN of an StorageGRID endpoint. Endpoints can be created using the Tenant Manager or the Tenant Management API. <p>The endpoint must exist for notification configuration to succeed. If the endpoint does not exist, a 400 Bad Request error is returned with the code InvalidArgument.</p> <ul style="list-style-type: none"> You cannot configure a notification for the following event types. These event types are not supported. <ul style="list-style-type: none"> ◦ s3:ReducedRedundancyLostObject ◦ s3:ObjectRestore:Completed Event notifications sent from StorageGRID use the standard JSON format except that they do not include some keys and use specific values for others, as shown in the following listing: <ul style="list-style-type: none"> eventSource <ul style="list-style-type: none"> sgws:s3 awsRegion <ul style="list-style-type: none"> not included x-amz-id-2 <ul style="list-style-type: none"> not included arn <ul style="list-style-type: none"> urn:sgws:s3:::bucket_name
PUT Bucket policy	This operation sets the policy attached to the bucket.

Operation	Implementation
PUT Bucket replication	<p>This operation configures StorageGRID CloudMirror replication for the bucket using the replication configuration XML provided in the request body. For CloudMirror replication, you should be aware of the following implementation details:</p> <ul style="list-style-type: none"> StorageGRID only supports V1 of the replication configuration. This means that StorageGRID does not support the use of the <code>Filter</code> element for rules, and follows V1 conventions for deletion of object versions. See the Amazon documentation on replication configuration for details. Bucket replication can be configured on versioned or unversioned buckets. You can specify a different destination bucket in each rule of the replication configuration XML. A source bucket can replicate to more than one destination bucket. Destination buckets must be specified as the URN of StorageGRID endpoints as specified in the Tenant Manager or the Tenant Management API. <p>The endpoint must exist for replication configuration to succeed. If the endpoint does not exist, the request fails as a 400 Bad Request. The error message states: Unable to save the replication policy. The specified endpoint URN does not exist: <i>URN</i>.</p> <ul style="list-style-type: none"> You do not need to specify a <code>Role</code> in the configuration XML. This value is not used by StorageGRID and will be ignored if submitted. If you omit the storage class from the configuration XML, StorageGRID uses the STANDARD storage class by default. If you delete an object from the source bucket or you delete the source bucket itself, the cross-region replication behavior is as follows: <ul style="list-style-type: none"> If you delete the object or bucket before it has been replicated, the object/bucket is not replicated and you are not notified. If you delete the object or bucket after it has been replicated, StorageGRID follows standard Amazon S3 delete behavior for V1 of cross-region replication.

Operation	Implementation
PUT Bucket tagging	<p>This operation uses the tagging subresource to add or update a set of tags for a bucket. When adding bucket tags, be aware of the following limitations:</p> <ul style="list-style-type: none"> Both StorageGRID and Amazon S3 support up to 50 tags for each bucket. Tags associated with a bucket must have unique tag keys. A tag key can be up to 128 Unicode characters in length. Tag values can be up to 256 Unicode characters in length. Key and values are case sensitive.
PUT Bucket versioning	<p>This implementation uses the versioning subresource to set the versioning state of an existing bucket. You can set the versioning state with one of the following values:</p> <ul style="list-style-type: none"> Enabled: Enables versioning for the objects in the bucket. All objects added to the bucket receive a unique version ID. Suspended: Disables versioning for the objects in the bucket. All objects added to the bucket receive the version ID null.

Related information

[Amazon Web Services \(AWS\) Documentation: Cross-Region Replication](#)

[Consistency controls](#)

[GET Bucket last access time request](#)

[Bucket and group access policies](#)

[Using S3 Object Lock](#)

[S3 operations tracked in the audit logs](#)

[Manage objects with ILM](#)

[Use a tenant account](#)

Creating an S3 lifecycle configuration

You can create an S3 lifecycle configuration to control when specific objects are deleted from the StorageGRID system.

The simple example in this section illustrates how an S3 lifecycle configuration can control when certain objects are deleted (expired) from specific S3 buckets. The example in this section is for illustration purposes

only. For complete details on creating S3 lifecycle configurations, see the section on object lifecycle management in the *Amazon Simple Storage Service Developer Guide*. Note that StorageGRID only supports Expiration actions; it does not support Transition actions.

[Amazon Simple Storage Service Developer Guide: Object lifecycle management](#)

What a lifecycle configuration is

A lifecycle configuration is a set of rules that are applied to the objects in specific S3 buckets. Each rule specifies which objects are affected and when those objects will expire (on a specific date or after some number of days).

StorageGRID supports up to 1,000 lifecycle rules in a lifecycle configuration. Each rule can include the following XML elements:

- **Expiration:** Delete an object when a specified date is reached or when a specified number of days is reached, starting from when the object was ingested.
- **NoncurrentVersionExpiration:** Delete an object when a specified number of days is reached, starting from when the object became noncurrent.
- **Filter (Prefix, Tag)**
- **Status**
- **ID**

If you apply a lifecycle configuration to a bucket, the lifecycle settings for the bucket always override StorageGRID ILM settings. StorageGRID uses the Expiration settings for the bucket, not ILM, to determine whether to delete or retain specific objects.

As a result, an object might be removed from the grid even though the placement instructions in an ILM rule still apply to the object. Or, an object might be retained on the grid even after any ILM placement instructions for the object have lapsed. For details, see “How ILM operates throughout an object’s life” in the instructions for managing objects with information lifecycle management.



Bucket lifecycle configuration can be used with buckets that have S3 Object Lock enabled, but bucket lifecycle configuration is not supported for legacy Compliant buckets.

StorageGRID supports the use of the following bucket operations to manage lifecycle configurations:

- **DELETE Bucket lifecycle**
- **GET Bucket lifecycle**
- **PUT Bucket lifecycle**

Creating the lifecycle configuration

As the first step in creating a lifecycle configuration, you create a JSON file that includes one or more rules. For example, this JSON file includes three rules, as follows:

1. Rule 1 applies only to objects that match the prefix `category1/` and that have a `key2` value of `tag2`. The `Expiration` parameter specifies that objects matching the filter will expire at midnight on 22 August 2020.
2. Rule 2 applies only to objects that match the prefix `category2/`. The `Expiration` parameter specifies that objects matching the filter will expire 100 days after they are ingested.



Rules that specify a number of days are relative to when the object was ingested. If the current date exceeds the ingest date plus the number of days, some objects might be removed from the bucket as soon as the lifecycle configuration is applied.

3. Rule 3 applies only to objects that match the prefix `category3/`. The `Expiration` parameter specifies that any noncurrent versions of matching objects will expire 50 days after they become noncurrent.

```
{  
    "Rules": [  
        {  
            "ID": "rule1",  
            "Filter": {  
                "And": {  
                    "Prefix": "category1/",  
                    "Tags": [  
                        {  
                            "Key": "key2",  
                            "Value": "tag2"  
                        }  
                    ]  
                }  
            },  
            "Expiration": {  
                "Date": "2020-08-22T00:00:00Z"  
            },  
            "Status": "Enabled"  
        },  
        {  
            "ID": "rule2",  
            "Filter": {  
                "Prefix": "category2/"  
            },  
            "Expiration": {  
                "Days": 100  
            },  
            "Status": "Enabled"  
        },  
        {  
            "ID": "rule3",  
            "Filter": {  
                "Prefix": "category3/"  
            },  
            "NoncurrentVersionExpiration": {  
                "NoncurrentDays": 50  
            },  
            "Status": "Enabled"  
        }  
    ]  
}
```

Applying a lifecycle configuration to a bucket

After you have created the lifecycle configuration file, you apply it to a bucket by issuing a PUT Bucket lifecycle request.

This request applies the lifecycle configuration in the example file to objects in a bucket named testbucket:bucket

```
aws s3api --endpoint-url <StorageGRID endpoint> put-bucket-lifecycle-configuration  
--bucket testbucket --lifecycle-configuration file://bktjson.json
```

To validate that a lifecycle configuration was successfully applied to the bucket, issue a GET Bucket lifecycle request. For example:

```
aws s3api --endpoint-url <StorageGRID endpoint> get-bucket-lifecycle-configuration  
--bucket testbucket
```

A successful response lists the lifecycle configuration you just applied.

Validating that bucket lifecycle expiration applies to an object

You can determine if an expiration rule in the lifecycle configuration applies to a specific object when issuing a PUT Object, HEAD Object, or GET Object request. If a rule applies, the response includes an `Expiration` parameter that indicates when the object expires and which expiration rule was matched.



Because bucket lifecycle overrides ILM, the `expiry-date` shown is the actual date the object will be deleted. For details, see “How object retention is determined” in the instructions for performing StorageGRID administration.

For example, this PUT Object request was issued on 22 Jun 2020 and places an object in the testbucket bucket.

```
aws s3api --endpoint-url <StorageGRID endpoint> put-object  
--bucket testbucket --key obj2test2 --body bktjson.json
```

The success response indicates that the object will expire in 100 days (01 Oct 2020) and that it matched Rule 2 of the lifecycle configuration.

```
{  
    "Expiration": "expiry-date=\\"Thu, 01 Oct 2020 09:07:49 GMT\\", rule-id=\\"rule2\\",  
    "ETag": "\\"9762f8a803bc34f5340579d4446076f7\\"  
}
```

For example, this HEAD Object request was used to get metadata for the same object in the testbucket bucket.

```
aws s3api --endpoint-url <StorageGRID endpoint> head-object  
--bucket testbucket --key obj2test2
```

The success response includes the object's metadata and indicates that the object will expire in 100 days and that it matched Rule 2.

```
{  
    "AcceptRanges": "bytes",  
    * "Expiration": "expiry-date=\\"Thu, 01 Oct 2020 09:07:48 GMT\\\"", rule-  
    id=\\"rule2\\\"",  
    "LastModified": "2020-06-23T09:07:48+00:00",  
    "ContentLength": 921,  
    "ETag": "\\"9762f8a803bc34f5340579d4446076f7\\\""  
    "ContentType": "binary/octet-stream",  
    "Metadata": {}  
}
```

Related information

[Operations on buckets](#)

[Manage objects with ILM](#)

Custom operations on buckets

The StorageGRID system supports custom bucket operations that are added on to the S3 REST API and are specific to the system.

The following table lists the custom bucket operations supported by StorageGRID.

Operation	Description	For more information
GET Bucket consistency	Returns the consistency level being applied to a particular bucket.	GET Bucket consistency request
PUT Bucket consistency	Sets the consistency level applied to a particular bucket.	PUT Bucket consistency request
GET Bucket last access time	Returns whether last access time updates are enabled or disabled for a particular bucket.	GET Bucket last access time request
PUT Bucket last access time	Allows you to enable or disable last access time updates for a particular bucket.	PUT Bucket last access time request

Operation	Description	For more information
DELETE Bucket metadata notification configuration	Deletes the metadata notification configuration XML associated with a particular bucket.	DELETE Bucket metadata notification configuration request
GET Bucket metadata notification configuration	Returns the metadata notification configuration XML associated with a particular bucket.	GET Bucket metadata notification configuration request
PUT Bucket metadata notification configuration	Configures the metadata notification service for a bucket.	PUT Bucket metadata notification configuration request
PUT Bucket modifications for compliance	Deprecated and not supported: You can no longer create new buckets with Compliance enabled.	Deprecated: PUT Bucket request modifications for compliance
GET Bucket compliance	Deprecated but supported: Returns the compliance settings currently in effect for an existing legacy Compliant bucket.	Deprecated: GET Bucket compliance request
PUT Bucket compliance	Deprecated but supported: Allows you to modify the compliance settings for an existing legacy Compliant bucket.	Deprecated: PUT Bucket compliance request

Related information

[S3 operations tracked in the audit logs](#)

Operations on objects

This section describes how the StorageGRID system implements S3 REST API operations for objects.

- [Using S3 Object Lock](#)
- [Using server-side encryption](#)
- [GET Object](#)
- [HEAD Object](#)
- [POST Object restore](#)
- [PUT Object](#)
- [PUT Object - Copy](#)

The following conditions apply to all object operations:

- StorageGRID consistency controls are supported by all operations on objects, with the exception of the following:

- GET Object ACL
- OPTIONS /
- PUT Object legal hold
- PUT Object retention
- Conflicting client requests, such as two clients writing to the same key, are resolved on a “latest-wins” basis. The timing for the “latest-wins” evaluation is based on when the StorageGRID system completes a given request, and not on when S3 clients begin an operation.
- All objects in a StorageGRID bucket are owned by the bucket owner, including objects created by an anonymous user, or by another account.
- Data objects ingested to the StorageGRID system through Swift cannot be accessed through S3.

The following table describes how StorageGRID implements S3 REST API object operations.

Operation	Implementation
DELETE Object	<p>Multi-Factor Authentication (MFA) and the response header <code>x-amz-mfa</code> are not supported.</p> <p>When processing a DELETE Object request, StorageGRID attempts to immediately remove all copies of the object from all stored locations. If successful, StorageGRID returns a response to the client immediately. If all copies cannot be removed within 30 seconds (for example, because a location is temporarily unavailable), StorageGRID queues the copies for removal and then indicates success to the client.</p> <p>Versioning</p> <p>To remove a specific version, the requestor must be the bucket owner and use the <code>versionId</code> subresource. Using this subresource permanently deletes the version. If the <code>versionId</code> corresponds to a delete marker, the response header <code>x-amz-delete-marker</code> is returned set to <code>true</code>.</p> <ul style="list-style-type: none"> • If an object is deleted without the <code>versionId</code> subresource on a version enabled bucket, it results in the generation of a delete marker. The <code>versionId</code> for the delete marker is returned using the <code>x-amz-version-id</code> response header, and the <code>x-amz-delete-marker</code> response header is returned set to <code>true</code>. • If an object is deleted without the <code>versionId</code> subresource on a version suspended bucket, it results in a permanent deletion of an already existing 'null' version or a 'null' delete marker, and the generation of a new 'null' delete marker. The <code>x-amz-delete-marker</code> response header is returned set to <code>true</code>. <p>Note: In certain cases, multiple delete markers might exist for an object.</p>
DELETE Multiple Objects	<p>Multi-Factor Authentication (MFA) and the response header <code>x-amz-mfa</code> are not supported.</p> <p>Multiple objects can be deleted in the same request message.</p>

Operation	Implementation
DELETE Object tagging	<p>Uses the <code>tagging</code> subresource to remove all tags from an object. Implemented with all Amazon S3 REST API behavior.</p> <p>Versioning</p> <p>If the <code>versionId</code> query parameter is not specified in the request, the operation deletes all tags from the most recent version of the object in a versioned bucket. If the current version of the object is a delete marker, a “MethodNotAllowed” status is returned with the <code>x-amz-delete-marker</code> response header set to <code>true</code>.</p>
GET Object	GET Object
GET Object ACL	<p>If the necessary access credentials are provided for the account, the operation returns a positive response and the ID, DisplayName, and Permission of the object owner, indicating that the owner has full access to the object.</p>
GET Object legal hold	Using S3 Object Lock
GET Object retention	Using S3 Object Lock
GET Object tagging	<p>Uses the <code>tagging</code> subresource to return all tags for an object. Implemented with all Amazon S3 REST API behavior</p> <p>Versioning</p> <p>If the <code>versionId</code> query parameter is not specified in the request, the operation returns all tags from the most recent version of the object in a versioned bucket. If the current version of the object is a delete marker, a “MethodNotAllowed” status is returned with the <code>x-amz-delete-marker</code> response header set to <code>true</code>.</p>
HEAD Object	HEAD Object
POST Object restore	POST Object restore
PUT Object	PUT Object
PUT Object - Copy	PUT Object - Copy

Operation	Implementation
PUT Object legal hold	Using S3 Object Lock
PUT Object retention	Using S3 Object Lock
PUT Object tagging	<p>Uses the <code>tagging</code> subresource to add a set of tags to an existing object. Implemented with all Amazon S3 REST API behavior</p> <p>Tag updates and ingest behavior</p> <p>When you use PUT Object tagging to update an object's tags, StorageGRID does not re-ingest the object. This means that the option for Ingest Behavior specified in the matching ILM rule is not used. Any changes to object placement that are triggered by the update are made when ILM is re-evaluated by normal background ILM processes.</p> <p>This means that if the ILM rule uses the Strict option for ingest behavior, no action is taken if the required object placements cannot be made (for example, because a newly required location is unavailable). The updated object retains its current placement until the required placement is possible.</p> <p>Resolving conflicts</p> <p>Conflicting client requests, such as two clients writing to the same key, are resolved on a "latest-wins" basis. The timing for the "latest-wins" evaluation is based on when the StorageGRID system completes a given request, and not on when S3 clients begin an operation.</p> <p>Versioning</p> <p>If the <code>versionId</code> query parameter is not specified in the request, the operation add tags to the most recent version of the object in a versioned bucket. If the current version of the object is a delete marker, a "MethodNotAllowed" status is returned with the <code>x-amz-delete-marker</code> response header set to <code>true</code>.</p>

Related information

[Consistency controls](#)

[S3 operations tracked in the audit logs](#)

Using S3 Object Lock

If the global S3 Object Lock setting is enabled for your StorageGRID system, you can create buckets with S3 Object Lock enabled and then specify retain-until-date and legal hold settings for each object version you add to that bucket.

S3 Object Lock allows you to specify object-level settings to prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely.

The StorageGRID S3 Object Lock feature provides a single retention mode that is equivalent to the Amazon S3 compliance mode. By default, a protected object version cannot be overwritten or deleted by any user. The StorageGRID S3 Object Lock feature does not support a governance mode, and it does not allow users with special permissions to bypass retention settings or to delete protected objects.

Enabling S3 Object Lock for a bucket

If the global S3 Object Lock setting is enabled for your StorageGRID system, you can optionally enable S3 Object Lock when you create each bucket. You can use either of these methods:

- Create the bucket using the Tenant Manager.

[Use a tenant account](#)

- Create the bucket using a PUT Bucket request with the `x-amz-bucket-object-lock_enabled` request header.

[Operations on buckets](#)

You cannot add or disable S3 Object Lock after the bucket is created. S3 Object Lock requires bucket versioning, which is enabled automatically when you create the bucket.

A bucket with S3 Object Lock enabled can contain a combination of objects with and without S3 Object Lock settings. StorageGRID does not support default retention for the objects in S3 Object Lock buckets, so the PUT Object Lock Configuration bucket operation is not supported.

Determining if S3 Object Lock is enabled for a bucket

To determine if S3 Object Lock is enabled, use the GET Object Lock Configuration request.

[Operations on buckets](#)

Creating an object with S3 Object Lock settings

To specify S3 Object Lock settings when adding an object version to a bucket that has S3 Object Lock enabled, issue a PUT Object, PUT Object - Copy, or Initiate Multipart Upload request. Use the following request headers.



You must enable S3 Object Lock when you create a bucket. You cannot add or disable S3 Object Lock after a bucket is created.

- `x-amz-object-lock-mode`, which must be `COMPLIANCE` (case sensitive).



If you specify `x-amz-object-lock-mode`, you must also specify `x-amz-object-lock-retain-until-date`.

- `x-amz-object-lock-retain-until-date`

- The retain-until-date value must be in the format `2020-08-10T21:46:00Z`. Fractional seconds are allowed, but only 3 decimal digits are preserved (milliseconds precision). Other ISO 8601 formats are not allowed.

- The retain-until-date must be in the future.

- `x-amz-object-lock-legal-hold`

If legal hold is ON (case-sensitive), the object is placed under a legal hold. If legal hold is OFF, no legal hold is placed. Any other value results in a 400 Bad Request (InvalidArgument) error.

If you use any of these request headers, be aware of these restrictions:

- The `Content-MD5` request header is required if any `x-amz-object-lock-*` request header is present in the `PUT Object` request. `Content-MD5` is not required for `PUT Object - Copy` or `Initiate Multipart Upload`.
- If the bucket does not have S3 Object Lock enabled and a `x-amz-object-lock-*` request header is present, a 400 Bad Request (InvalidRequest) error is returned.
- The `PUT Object` request supports the use of `x-amz-storage-class: REDUCED_REDUNDANCY` to match AWS behavior. However, when an object is ingested into a bucket with S3 Object Lock enabled, StorageGRID will always perform a dual-commit ingest.
- A subsequent `GET` or `HEAD` Object version response will include the headers `x-amz-object-lock-mode`, `x-amz-object-lock-retain-until-date`, and `x-amz-object-lock-legal-hold`, if configured and if the request sender has the correct `s3:Get*` permissions.
- A subsequent `DELETE Object` version or `DELETE Objects` versions request will fail if it is before the retain-until-date or if a legal hold is on.

Updating S3 Object Lock settings

If you need to update the legal hold or retention settings for an existing object version, you can perform the following object subresource operations:

- `PUT Object legal-hold`

If the new legal-hold value is ON, the object is placed under a legal hold. If the legal-hold value is OFF, the legal hold is lifted.

- `PUT Object retention`

- The mode value must be `COMPLIANCE` (case sensitive).
 - The retain-until-date value must be in the format `2020-08-10T21:46:00Z`. Fractional seconds are allowed, but only 3 decimal digits are preserved (milliseconds precision). Other ISO 8601 formats are not allowed.
 - If an object version has an existing retain-until-date, you can only increase it. The new value must be in the future.

Related information

[Manage objects with ILM](#)

[Use a tenant account](#)

[PUT Object](#)

[PUT Object - Copy](#)

[Initiate Multipart Upload](#)

[Object versioning](#)

[Amazon Simple Storage Service User Guide: Using S3 Object Lock](#)

Using server-side encryption

Server-side encryption allows you to protect your object data at rest. StorageGRID encrypts the data as it writes the object and decrypts the data when you access the object.

If you want to use server-side encryption, you can choose either of two mutually exclusive options, based on how the encryption keys are managed:

- **SSE (server-side encryption with StorageGRID-managed keys):** When you issue an S3 request to store an object, StorageGRID encrypts the object with a unique key. When you issue an S3 request to retrieve the object, StorageGRID uses the stored key to decrypt the object.
- **SSE-C (server-side encryption with customer-provided keys):** When you issue an S3 request to store an object, you provide your own encryption key. When you retrieve an object, you provide the same encryption key as part of your request. If the two encryption keys match, the object is decrypted and your object data is returned.

While StorageGRID manages all object encryption and decryption operations, you must manage the encryption keys you provide.



The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object.



If an object is encrypted with SSE or SSE-C, any bucket-level or grid-level encryption settings are ignored.

Using SSE

To encrypt an object with a unique key managed by StorageGRID, you use the following request header:

`x-amz-server-side-encryption`

The SSE request header is supported by the following object operations:

- [PUT Object](#)
- [PUT Object - Copy](#)
- [Initiate Multipart Upload](#)

Using SSE-C

To encrypt an object with a unique key that you manage, you use three request headers:

Request header	Description
x-amz-server-side-encryption-customer-algorithm	Specify the encryption algorithm. The header value must be AES256.
x-amz-server-side-encryption-customer-key	Specify the encryption key that will be used to encrypt or decrypt the object. The value for the key must be 256-bit, base64-encoded.
x-amz-server-side-encryption-customer-key-MD5	Specify the MD5 digest of the encryption key according to RFC 1321, which is used to ensure the encryption key was transmitted without error. The value for the MD5 digest must be base64-encoded 128-bit.

The SSE-C request headers are supported by the following object operations:

- GET Object
- HEAD Object
- PUT Object
- PUT Object - Copy
- Initiate Multipart Upload
- Upload Part
- Upload Part - Copy

Considerations for using server-side encryption with customer-provided keys (SSE-C)

Before using SSE-C, be aware of the following considerations:

- You must use https.



StorageGRID rejects any requests made over http when using SSE-C. For security considerations, you should consider any key you send accidentally using http to be compromised. Discard the key, and rotate as appropriate.

- The ETag in the response is not the MD5 of the object data.
- You must manage the mapping of encryption keys to objects. StorageGRID does not store encryption keys. You are responsible for tracking the encryption key you provide for each object.
- If your bucket is versioning-enabled, each object version should have its own encryption key. You are responsible for tracking the encryption key used for each object version.
- Because you manage encryption keys on the client side, you must also manage any additional safeguards, such as key rotation, on the client side.



The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object.

- If CloudMirror replication is configured for the bucket, you cannot ingest SSE-C objects. The ingest operation will fail.

Related information

[GET Object](#)

[HEAD Object](#)

[PUT Object](#)

[PUT Object - Copy](#)

[Initiate Multipart Upload](#)

[Upload Part](#)

[Upload Part - Copy](#)

[Amazon S3 Developer Guide: Protecting Data Using Server-Side Encryption with Customer-Provided Encryption Keys \(SSE-C\)](#)

GET Object

You can use the S3 GET Object request to retrieve an object from an S3 bucket.

partNumber request parameter is not supported

The `partNumber` request parameter is not supported for GET Object requests. You cannot perform a GET request to retrieve a specific part of a multipart object. A 501 Not Implemented error is returned with the following message:

GET Object by partNumber is not implemented

Request headers for server-side encryption with customer-provided encryption keys (SSE-C)

Use all three of the headers if the object is encrypted with a unique key that you provided.

- `x-amz-server-side-encryption-customer-algorithm`: Specify AES256.
- `x-amz-server-side-encryption-customer-key`: Specify your encryption key for the object.
- `x-amz-server-side-encryption-customer-key-MD5`: Specify the MD5 digest of the object's encryption key.



The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object. Before using customer-provided keys to secure object data, review the considerations in "Using server-side encryption."

UTF-8 characters in user metadata

StorageGRID does not parse or interpret escaped UTF-8 characters in user-defined metadata. GET requests for an object with escaped UTF-8 characters in user-defined metadata do not return the `x-amz-missing-meta` header if the key name or value includes unprintable characters.

Unsupported request header

The following request header is not supported and returns `XNotImplemented`:

- `x-amz-website-redirect-location`

Versioning

If a `versionId` subresource is not specified, the operation fetches the most recent version of the object in a versioned bucket. If the current version of the object is a delete marker, a “Not Found” status is returned with the `x-amz-delete-marker` response header set to `true`.

Behavior of GET Object for Cloud Storage Pool objects

If an object has been stored in a Cloud Storage Pool (see the instructions for managing objects with information lifecycle management), the behavior of a GET Object request depends on the state of the object. See “HEAD Object” for more details.



If an object is stored in a Cloud Storage Pool and one or more copies of the object also exist on the grid, GET Object requests will attempt to retrieve data from the grid, before retrieving it from the Cloud Storage Pool.

State of object	Behavior of GET Object
Object ingested into StorageGRID but not yet evaluated by ILM, or object stored in a traditional storage pool or using erasure coding	200 OK A copy of the object is retrieved.
Object in Cloud Storage Pool but not yet transitioned to a non-retrievable state	200 OK A copy of the object is retrieved.
Object transitioned to a non-retrievable state	403 Forbidden, <code>InvalidObjectState</code> Use a POST Object restore request to restore the object to a retrievable state.
Object in process of being restored from a non-retrievable state	403 Forbidden, <code>InvalidObjectState</code> Wait for the POST Object restore request to complete.
Object fully restored to the Cloud Storage Pool	200 OK A copy of the object is retrieved.

Multipart or segmented objects in a Cloud Storage Pool

If you uploaded a multipart object or if StorageGRID split a large object into segments, StorageGRID determines whether the object is available in the Cloud Storage Pool by sampling a subset of the object's parts or segments. In some cases, a GET Object request might incorrectly return 200 OK when some parts of the object have already been transitioned to a non-retrievable state or when some parts of the object have not yet been restored.

In these cases:

- The GET Object request might return some data but stop midway through the transfer.
- A subsequent GET Object request might return 403 Forbidden.

Related information

[Using server-side encryption](#)

[Manage objects with ILM](#)

[POST Object restore](#)

[S3 operations tracked in the audit logs](#)

HEAD Object

You can use the S3 HEAD Object request to retrieve metadata from an object without returning the object itself. If the object is stored in a Cloud Storage Pool, you can use HEAD Object to determine the object's transition state.

Request headers for server-side encryption with customer-provided encryption keys (SSE-C)

Use all three of these headers if the object is encrypted with a unique key that you provided.

- `x-amz-server-side-encryption-customer-algorithm`: Specify AES256.
- `x-amz-server-side-encryption-customer-key`: Specify your encryption key for the object.
- `x-amz-server-side-encryption-customer-key-MD5`: Specify the MD5 digest of the object's encryption key.



The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object. Before using customer-provided keys to secure object data, review the considerations in "Using server-side encryption."

UTF-8 characters in user metadata

StorageGRID does not parse or interpret escaped UTF-8 characters in user-defined metadata. HEAD requests for an object with escaped UTF-8 characters in user-defined metadata do not return the `x-amz-missing-meta` header if the key name or value includes unprintable characters.

Unsupported request header

The following request header is not supported and returns XNotImplemented:

- `x-amz-website-redirect-location`

Response headers for Cloud Storage Pool objects

If the object is stored in a Cloud Storage Pool (see the instructions for managing objects with information lifecycle management), the following response headers are returned:

- `x-amz-storage-class: GLACIER`
- `x-amz-restore`

The response headers provide information about the state of an object as it is moved to a Cloud Storage Pool, optionally transitioned to a non-retrievable state, and restored.

State of object	Response to HEAD object
Object ingested into StorageGRID but not yet evaluated by ILM, or object stored in a traditional storage pool or using erasure coding	200 OK (No special response header is returned.)
Object in Cloud Storage Pool but not yet transitioned to a non-retrievable state	<p>200 OK</p> <p><code>x-amz-storage-class: GLACIER</code></p> <p><code>x-amz-restore: ongoing-request="false", expiry-date="Sat, 23 July 20 2030 00:00:00 GMT"</code></p> <p>Until the object is transitioned to a non-retrievable state, the value for <code>expiry-date</code> is set to some distant time in the future. The exact time of transition is not controlled by the StorageGRID system.</p>
Object has transitioned to non-retrievable state, but at least one copy also exists on the grid	<p>200 OK</p> <p><code>x-amz-storage-class: GLACIER</code></p> <p><code>x-amz-restore: ongoing-request="false", expiry-date="Sat, 23 July 20 2030 00:00:00 GMT"</code></p> <p>The value for <code>expiry-date</code> is set to some distant time in the future.</p> <p>Note: If the copy on the grid is not available (for example, a Storage Node is down), you must issue a POST Object restore request to restore the copy from the Cloud Storage Pool before you can successfully retrieve the object.</p>

State of object	Response to HEAD object
Object transitioned to a non-retrievable state, and no copy exists on the grid	200 OK x-amz-storage-class: GLACIER
Object in process of being restored from a non-retrievable state	200 OK x-amz-storage-class: GLACIER x-amz-restore: ongoing-request="true"
Object fully restored to the Cloud Storage Pool	200 OK x-amz-storage-class: GLACIER x-amz-restore: ongoing-request="false", expiry-date="Sat, 23 July 20 2018 00:00:00 GMT" The expiry-date indicates when the object in the Cloud Storage Pool will be returned to a non-retrievable state.

Multipart or segmented objects in a Cloud Storage Pool

If you uploaded a multipart object or if StorageGRID split a large object into segments, StorageGRID determines whether the object is available in the Cloud Storage Pool by sampling a subset of the object's parts or segments. In some cases, a HEAD Object request might incorrectly return `x-amz-restore: ongoing-request="false"` when some parts of the object have already been transitioned to a non-retrievable state or when some parts of the object have not yet been restored.

Versioning

If a `versionId` subresource is not specified, the operation fetches the most recent version of the object in a versioned bucket. If the current version of the object is a delete marker, a "Not Found" status is returned with the `x-amz-delete-marker` response header set to true.

Related information

[Using server-side encryption](#)

[Manage objects with ILM](#)

[POST Object restore](#)

[S3 operations tracked in the audit logs](#)

POST Object restore

You can use the S3 POST Object restore request to restore an object that is stored in a Cloud Storage Pool.

Supported request type

StorageGRID only supports POST Object restore requests to restore an object. It does not support the SELECT type of restoration. Select requests return XNotImplemented.

Versioning

Optionally, specify `versionId` to restore a specific version of an object in a versioned bucket. If you do not specify `versionId`, the most recent version of the object is restored

Behavior of POST Object restore on Cloud Storage Pool objects

If an object has been stored in a Cloud Storage Pool (see the instructions for managing objects with information lifecycle management), a POST Object restore request has the following behavior, based on the state of the object. See “HEAD Object” for more details.



If an object is stored in a Cloud Storage Pool and one or more copies of the object also exist on the grid, there is no need to restore the object by issuing a POST Object restore request. Instead, the local copy can be retrieved directly, using a GET Object request.

State of object	Behavior of POST Object restore
Object ingested into StorageGRID but not yet evaluated by ILM, or object is not in a Cloud Storage Pool	403 Forbidden, InvalidObjectState
Object in Cloud Storage Pool but not yet transitioned to a non-retrievable state	200 OK No changes are made. Note: Before an object has been transitioned to a non-retrievable state, you cannot change its expiry-date.
Object transitioned to a non-retrievable state	202 Accepted Restores a retrievable copy of the object to the Cloud Storage Pool for the number of days specified in the request body. At the end of this period, the object is returned to a non-retrievable state. Optionally, use the <code>Tier</code> request element to determine how long the restore job will take to finish (Expedited, Standard, or Bulk). If you do not specify <code>Tier</code> , the Standard tier is used. Attention: If an object has been transitioned to S3 Glacier Deep Archive or the Cloud Storage Pool uses Azure Blob Storage, you cannot restore it using the Expedited tier. The following error is returned 403 Forbidden, InvalidTier: Retrieval option is not supported by this storage class.

State of object	Behavior of POST Object restore
Object in process of being restored from a non-retrievable state	409 Conflict, RestoreAlreadyInProgress
Object fully restored to the Cloud Storage Pool	<p>200 OK</p> <p>Note: If an object has been restored to a retrievable state, you can change its expiry-date by reissuing the POST Object restore request with a new value for Days. The restoration date is updated relative to the time of the request.</p>

Related information

[Manage objects with ILM](#)

[HEAD Object](#)

[S3 operations tracked in the audit logs](#)

PUT Object

You can use the S3 PUT Object request to add an object to a bucket.

Resolving conflicts

Conflicting client requests, such as two clients writing to the same key, are resolved on a “latest-wins” basis. The timing for the “latest-wins” evaluation is based on when the StorageGRID system completes a given request, and not on when S3 clients begin an operation.

Object size

StorageGRID supports objects up to 5 TB in size.

User metadata size

Amazon S3 limits the size of user-defined metadata within each PUT request header to 2 KB. StorageGRID limits user metadata to 24 KiB. The size of user-defined metadata is measured by taking the sum of the number of bytes in the UTF-8 encoding of each key and value.

UTF-8 characters in user metadata

If a request includes (unesaped) UTF-8 values in the key name or value of user-defined metadata, StorageGRID behavior is undefined.

StorageGRID does not parse or interpret escaped UTF-8 characters included in the key name or value of user-defined metadata. Escaped UTF-8 characters are treated as ASCII characters:

- PUT, PUT Object-Copy, GET, and HEAD requests succeed if user-defined metadata includes escaped UTF-8 characters.
- StorageGRID does not return the x-amz-missing-meta header if the interpreted value of the key name or value includes unprintable characters.

Object tag limits

You can add tags to new objects when you upload them, or you can add them to existing objects. Both StorageGRID and Amazon S3 support up to 10 tags for each object. Tags associated with an object must have unique tag keys. A tag key can be up to 128 Unicode characters in length and tag values can be up to 256 Unicode characters in length. Key and values are case sensitive.

Object ownership

In StorageGRID, all objects are owned by the bucket owner account, including objects created by a non-owner account or an anonymous user.

Supported request headers

The following request headers are supported:

- Cache-Control
- Content-Disposition
- Content-Encoding

When you specify `aws-chunked` for `Content-Encoding` StorageGRID does not verify the following items:

- StorageGRID does not verify the `chunk-signature` against the chunk data.
- StorageGRID does not verify the value that you provide for `x-amz-decoded-content-length` against the object.

- Content-Language
- Content-Length
- Content-MD5
- Content-Type
- Expires
- Transfer-Encoding

Chunked transfer encoding is supported if `aws-chunked` payload signing is also used.

- `x-amz-meta-`, followed by a name-value pair containing user-defined metadata.

When specifying the name-value pair for user-defined metadata, use this general format:

```
x-amz-meta-<em>name</em>: <em>value</em>
```

If you want to use the **User Defined Creation Time** option as the Reference Time for an ILM rule, you must use `creation-time` as the name of the metadata that records when the object was created. For example:

```
x-amz-meta-creation-time: 1443399726
```

The value for `creation-time` is evaluated as seconds since January 1, 1970.



An ILM rule cannot use both a **User Defined Creation Time** for the Reference Time and the Balanced or Strict options for Ingest Behavior. An error is returned when the ILM rule is created.

- `x-amz-tagging`
- S3 Object Lock request headers
 - `x-amz-object-lock-mode`
 - `x-amz-object-lock-retain-until-date`
 - `x-amz-object-lock-legal-hold`

Using S3 Object Lock

- SSE request headers:
 - `x-amz-server-side-encryption`
 - `x-amz-server-side-encryption-customer-key-MD5`
 - `x-amz-server-side-encryption-customer-key`
 - `x-amz-server-side-encryption-customer-algorithm`

S3 REST API supported operations and limitations

Unsupported request headers

The following request headers are not supported:

- The `x-amz-acl` request header is not supported.
- The `x-amz-website-redirect-location` request header is not supported and returns `XNotImplemented`.

Storage class options

The `x-amz-storage-class` request header is supported. The value submitted for `x-amz-storage-class` affects how StorageGRID protects object data during ingest and not how many persistent copies of the object are stored in the StorageGRID system (which is determined by ILM).

If the ILM rule matching an ingested object uses the Strict option for Ingest Behavior, the `x-amz-storage-class` header has no effect.

The following values can be used for `x-amz-storage-class`:

- STANDARD (Default)
 - **Dual commit:** If the ILM rule specifies the Dual commit option for Ingest Behavior, as soon as an object is ingested a second copy of that object is created and distributed to a different Storage Node (dual

commit). When the ILM is evaluated, StorageGRID determines if these initial interim copies satisfy the placement instructions in the rule. If they do not, new object copies might need to be made in different locations and the initial interim copies might need to be deleted.

- **Balanced:** If the ILM rule specifies the Balanced option and StorageGRID cannot immediately make all copies specified in the rule, StorageGRID makes two interim copies on different Storage Nodes.

If StorageGRID can immediately create all object copies specified in the ILM rule (synchronous placement), the `x-amz-storage-class` header has no effect.

- **REDUCED_REDUNDANCY**

- **Dual commit:** If the ILM rule specifies the Dual commit option for Ingest Behavior, StorageGRID creates a single interim copy as the object is ingested (single commit).
- **Balanced:** If the ILM rule specifies the Balanced option, StorageGRID makes a single interim copy only if the system cannot immediately make all copies specified in the rule. If StorageGRID can perform synchronous placement, this header has no effect. The `REDUCED_REDUNDANCY` option is best used when the ILM rule that matches the object creates a single replicated copy. In this case using `REDUCED_REDUNDANCY` eliminates the unnecessary creation and deletion of an extra object copy for every ingest operation.

Using the `REDUCED_REDUNDANCY` option is not recommended in other circumstances.

`REDUCED_REDUNDANCY` increases the risk of object data loss during ingest. For example, you might lose data if the single copy is initially stored on a Storage Node that fails before ILM evaluation can occur.

Attention: Having only one replicated copy for any time period puts data at risk of permanent loss. If only one replicated copy of an object exists, that object is lost if a Storage Node fails or has a significant error. You also temporarily lose access to the object during maintenance procedures such as upgrades.

Specifying `REDUCED_REDUNDANCY` only affects how many copies are created when an object is first ingested. It does not affect how many copies of the object are made when the object is evaluated by the active ILM policy, and does not result in data being stored at lower levels of redundancy in the StorageGRID system.

Note: If you are ingesting an object into a bucket with S3 Object Lock enabled, the `REDUCED_REDUNDANCY` option is ignored. If you are ingesting an object into a legacy Compliant bucket, the `REDUCED_REDUNDANCY` option returns an error. StorageGRID will always perform a dual-commit ingest to ensure that compliance requirements are satisfied.

Request headers for server-side encryption

You can use the following request headers to encrypt an object with server-side encryption. The SSE and SSE-C options are mutually exclusive.

- **SSE:** Use the following header if you want to encrypt the object with a unique key managed by StorageGRID.
 - `x-amz-server-side-encryption`
- **SSE-C:** Use all three of these headers if you want to encrypt the object with a unique key that you provide and manage.
 - `x-amz-server-side-encryption-customer-algorithm`: Specify AES256.
 - `x-amz-server-side-encryption-customer-key`: Specify your encryption key for the new object.

- `x-amz-server-side-encryption-customer-key-MD5`: Specify the MD5 digest of the new object's encryption key.

Attention: The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object. Before using customer-provided keys to secure object data, review the considerations in “Using server-side encryption.”

Note: If an object is encrypted with SSE or SSE-C, any bucket-level or grid-level encryption settings are ignored.

Versioning

If versioning is enabled for a bucket, a unique `versionId` is automatically generated for the version of the object being stored. This `versionId` is also returned in the response using the `x-amz-version-id` response header.

If versioning is suspended, the object version is stored with a null `versionId` and if a null version already exists it will be overwritten.

Related information

[Manage objects with ILM](#)

[Operations on buckets](#)

[S3 operations tracked in the audit logs](#)

[Using server-side encryption](#)

[How client connections can be configured](#)

PUT Object - Copy

You can use the S3 PUT Object - Copy request to create a copy of an object that is already stored in S3. A PUT Object - Copy operation is the same as performing a GET and then a PUT.

Resolving conflicts

Conflicting client requests, such as two clients writing to the same key, are resolved on a “latest-wins” basis. The timing for the “latest-wins” evaluation is based on when the StorageGRID system completes a given request, and not on when S3 clients begin an operation.

Object size

StorageGRID supports objects up to 5 TB in size.

UTF-8 characters in user metadata

If a request includes (unesaped) UTF-8 values in the key name or value of user-defined metadata, StorageGRID behavior is undefined.

StorageGRID does not parse or interpret escaped UTF-8 characters included in the key name or value of user-defined metadata. Escaped UTF-8 characters are treated as ASCII characters:

- Requests succeed if user-defined metadata includes escaped UTF-8 characters.
- StorageGRID does not return the `x-amz-missing-meta` header if the interpreted value of the key name or value includes unprintable characters.

Supported request headers

The following request headers are supported:

- `Content-Type`
- `x-amz-copy-source`
- `x-amz-copy-source-if-match`
- `x-amz-copy-source-if-none-match`
- `x-amz-copy-source-if-unmodified-since`
- `x-amz-copy-source-if-modified-since`
- `x-amz-meta-`, followed by a name-value pair containing user-defined metadata
- `x-amz-metadata-directive`: The default value is `COPY`, which enables you to copy the object and associated metadata.

You can specify `REPLACE` to overwrite the existing metadata when copying the object, or to update the object metadata.

- `x-amz-storage-class`
- `x-amz-tagging-directive`: The default value is `COPY`, which enables you to copy the object and all tags.

You can specify `REPLACE` to overwrite the existing tags when copying the object, or to update the tags.

- S3 Object Lock request headers:
 - `x-amz-object-lock-mode`
 - `x-amz-object-lock-retain-until-date`
 - `x-amz-object-lock-legal-hold`

Using S3 Object Lock

- SSE request headers:
 - `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`
 - `x-amz-server-side-encryption`
 - `x-amz-server-side-encryption-customer-key-MD5`
 - `x-amz-server-side-encryption-customer-key`
 - `x-amz-server-side-encryption-customer-algorithm`

Request headers for server-side encryption

Unsupported request headers

The following request headers are not supported:

- Cache-Control
- Content-Disposition
- Content-Encoding
- Content-Language
- Expires
- x-amz-website-redirect-location

Storage class options

The `x-amz-storage-class` request header is supported, and affects how many object copies StorageGRID creates if the matching ILM rule specifies an Ingest Behavior of Dual commit or Balanced.

- STANDARD

(Default) Specifies a dual-commit ingest operation when the ILM rule uses the Dual commit option, or when the Balanced option falls back to creating interim copies.

- REDUCED_REDUNDANCY

Specifies a single-commit ingest operation when the ILM rule uses the Dual commit option, or when the Balanced option falls back to creating interim copies.



If you are ingesting an object into a bucket with S3 Object Lock enabled, the REDUCED_REDUNDANCY option is ignored. If you are ingesting an object into a legacy Compliant bucket, the REDUCED_REDUNDANCY option returns an error. StorageGRID will always perform a dual-commit ingest to ensure that compliance requirements are satisfied.

Using `x-amz-copy-source` in PUT Object - Copy

If the source bucket and key, specified in the `x-amz-copy-source` header, are different from the destination bucket and key, a copy of the source object data is written to the destination.

If the source and destination match, and the `x-amz-metadata-directive` header is specified as `REPLACE`, the object's metadata is updated with the metadata values supplied in the request. In this case, StorageGRID does not re-ingest the object. This has two important consequences:

- You cannot use PUT Object - Copy to encrypt an existing object in place, or to change the encryption of an existing object in place. If you supply the `x-amz-server-side-encryption` header or the `x-amz-server-side-encryption-customer-algorithm` header, StorageGRID rejects the request and returns `XNotImplemented`.
- The option for Ingest Behavior specified in the matching ILM rule is not used. Any changes to object placement that are triggered by the update are made when ILM is re-evaluated by normal background ILM processes.

This means that if the ILM rule uses the Strict option for ingest behavior, no action is taken if the required object placements cannot be made (for example, because a newly required location is unavailable). The updated object retains its current placement until the required placement is possible.

Request headers for server-side encryption

If you use server-side encryption, the request headers you provide depend on whether the source object is encrypted and on whether you plan to encrypt the target object.

- If the source object is encrypted using a customer-provided key (SSE-C), you must include the following three headers in the PUT Object - Copy request, so the object can be decrypted and then copied:
 - `x-amz-copy-source-server-side-encryption-customer-algorithm` Specify AES256.
 - `x-amz-copy-source-server-side-encryption-customer-key` Specify the encryption key you provided when you created the source object.
 - `x-amz-copy-source-server-side-encryption-customer-key-MD5`: Specify the MD5 digest you provided when you created the source object.
- If you want to encrypt the target object (the copy) with a unique key that you provide and manage, include the following three headers:
 - `x-amz-server-side-encryption-customer-algorithm`: Specify AES256.
 - `x-amz-server-side-encryption-customer-key`: Specify a new encryption key for the target object.
 - `x-amz-server-side-encryption-customer-key-MD5`: Specify the MD5 digest of the new encryption key.

Attention: The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object. Before using customer-provided keys to secure object data, review the considerations in “Using server-side encryption.”

- If you want to encrypt the target object (the copy) with a unique key managed by StorageGRID (SSE), include this header in the PUT Object - Copy request:
 - `x-amz-server-side-encryption`

Note: The `server-side-encryption` value of the object cannot be updated. Instead, make a copy with a new `server-side-encryption` value using `x-amz-metadata-directive: REPLACE`.

Versioning

If the source bucket is versioned, you can use the `x-amz-copy-source` header to copy the latest version of an object. To copy a specific version of an object, you must explicitly specify the version to copy using the `versionId` subresource. If the destination bucket is versioned, the generated version is returned in the `x-amz-version-id` response header. If versioning is suspended for the target bucket, then `x-amz-version-id` returns a “null” value.

Related information

[Manage objects with ILM](#)

[Using server-side encryption](#)

[S3 operations tracked in the audit logs](#)

Operations for multipart uploads

This section describes how StorageGRID supports operations for multipart uploads.

- [List multipart uploads](#)
- [Initiate Multipart Upload](#)
- [Upload Part](#)
- [Upload Part - Copy](#)
- [Complete Multipart Upload](#)

The following conditions and notes apply to all multipart upload operations:

- You should not exceed 1,000 concurrent multipart uploads to a single bucket because the results of List Multipart Uploads queries for that bucket might return incomplete results.
- StorageGRID enforces AWS size limits for multipart parts. S3 clients must follow these guidelines:
 - Each part in a multipart upload must be between 5 MiB (5,242,880 bytes) and 5 GiB (5,368,709,120 bytes).
 - The last part can be smaller than 5 MiB (5,242,880 bytes).
 - In general, part sizes should be as large as possible. For example, use part sizes of 5 GiB for a 100 GiB object. Since each part is considered a unique object, using large part sizes reduces StorageGRID metadata overhead.
 - For objects smaller than 5 GiB, consider using non-multipart upload instead.
- ILM is evaluated for each part of a multipart object as it is ingested and for the object as a whole when the multipart upload completes, if the ILM rule uses the Strict or Balanced ingest behavior. You should be aware of how this affects object and part placement:
 - If ILM changes while an S3 multipart upload is in progress, when the multipart upload completes some parts of the object might not meet current ILM requirements. Any part that is not placed correctly is queued for ILM re-evaluation, and is moved to the correct location later.
 - When evaluating ILM for a part, StorageGRID filters on the size of the part, not the size of the object. This means that parts of an object can be stored in locations that do not meet ILM requirements for the object as a whole. For example, if a rule specifies that all objects 10 GB or larger are stored at DC1 while all smaller objects are stored at DC2, at ingest each 1 GB part of a 10-part multipart upload is stored at DC2. When ILM is evaluated for the object as a whole, all parts of the object are moved to DC1.
- All of the multipart upload operations support StorageGRID consistency controls.
- As required, you can use server-side encryption with multipart uploads. To use SSE (server-side encryption with StorageGRID-managed keys), you include the `x-amz-server-side-encryption` request header in the Initiate Multipart Upload request only. To use SSE-C (server-side encryption with customer-provided keys), you specify the same three encryption key request headers in the Initiate Multipart Upload request and in each subsequent Upload Part request.

Operation	Implementation
List Multipart Uploads	See List Multipart Uploads

Operation	Implementation
Initiate Multipart Upload	See Initiate Multipart Upload
Upload Part	See Upload Part
Upload Part - Copy	See Upload Part - Copy
Complete Multipart Upload	See Complete Multipart Upload
Abort Multipart Upload	Implemented with all Amazon S3 REST API behavior
List Parts	Implemented with all Amazon S3 REST API behavior

Related information

[Consistency controls](#)

[Using server-side encryption](#)

List Multipart Uploads

The List Multipart Uploads operation lists in-progress multipart uploads for a bucket.

The following request parameters are supported:

- `encoding-type`
- `max-uploads`
- `key-marker`
- `prefix`
- `upload-id-marker`

The `delimiter` request parameter is not supported.

Versioning

Multipart upload consists of separate operations for initiating the upload, listing uploads, uploading parts, assembling the uploaded parts, and completing the upload. When the Complete Multipart Upload operation is performed, that is the point when objects are created (and versioned if applicable).

Initiate Multipart Upload

The Initiate Multipart Upload operation initiates a multipart upload for an object, and returns an upload ID.

The `x-amz-storage-class` request header is supported. The value submitted for `x-amz-storage-class` affects how StorageGRID protects object data during ingest and not how many persistent copies of the object are stored in the StorageGRID system (which is determined by ILM).

If the ILM rule matching an ingested object uses the Strict option for Ingest Behavior, the `x-amz-storage-class` header has no effect.

The following values can be used for `x-amz-storage-class`:

- **STANDARD (Default)**
 - **Dual commit:** If the ILM rule specifies the Dual commit option for Ingest Behavior, as soon as an object is ingested a second copy of that object is created and distributed to a different Storage Node (dual commit). When the ILM is evaluated, StorageGRID determines if these initial interim copies satisfy the placement instructions in the rule. If they do not, new object copies might need to be made in different locations and the initial interim copies might need to be deleted.
 - **Balanced:** If the ILM rule specifies the Balanced option and StorageGRID cannot immediately make all copies specified in the rule, StorageGRID makes two interim copies on different Storage Nodes.
- If StorageGRID can immediately create all object copies specified in the ILM rule (synchronous placement), the `x-amz-storage-class` header has no effect.
- **REDUCED_REDUNDANCY**
 - **Dual commit:** If the ILM rule specifies the Dual commit option for Ingest Behavior, StorageGRID creates a single interim copy as the object is ingested (single commit).
 - **Balanced:** If the ILM rule specifies the Balanced option, StorageGRID makes a single interim copy only if the system cannot immediately make all copies specified in the rule. If StorageGRID can perform synchronous placement, this header has no effect. The `REDUCED_REDUNDANCY` option is best used when the ILM rule that matches the object creates a single replicated copy. In this case using `REDUCED_REDUNDANCY` eliminates the unnecessary creation and deletion of an extra object copy for every ingest operation.

Using the `REDUCED_REDUNDANCY` option is not recommended in other circumstances.

`REDUCED_REDUNDANCY` increases the risk of object data loss during ingest. For example, you might lose data if the single copy is initially stored on a Storage Node that fails before ILM evaluation can occur.

Attention: Having only one replicated copy for any time period puts data at risk of permanent loss. If only one replicated copy of an object exists, that object is lost if a Storage Node fails or has a significant error. You also temporarily lose access to the object during maintenance procedures such as upgrades.

Specifying `REDUCED_REDUNDANCY` only affects how many copies are created when an object is first ingested. It does not affect how many copies of the object are made when the object is evaluated by the active ILM policy, and does not result in data being stored at lower levels of redundancy in the StorageGRID system.

Note: If you are ingesting an object into a bucket with S3 Object Lock enabled, the `REDUCED_REDUNDANCY` option is ignored. If you are ingesting an object into a legacy Compliant bucket, the `REDUCED_REDUNDANCY` option returns an error. StorageGRID will always perform a dual-commit ingest to ensure that compliance requirements are satisfied.

The following request headers are supported:

- `Content-Type`
- `x-amz-meta-`, followed by a name-value pair containing user-defined metadata

When specifying the name-value pair for user-defined metadata, use this general format:

```
x-amz-meta-_name_: `value`
```

If you want to use the **User Defined Creation Time** option as the Reference Time for an ILM rule, you must use `creation-time` as the name of the metadata that records when the object was created. For example:

```
x-amz-meta-creation-time: 1443399726
```

The value for `creation-time` is evaluated as seconds since January 1, 1970.



Adding `creation-time` as user-defined metadata is not allowed if you are adding an object to a bucket that has legacy Compliance enabled. An error will be returned.

- S3 Object Lock request headers:

- `x-amz-object-lock-mode`
- `x-amz-object-lock-retain-until-date`
- `x-amz-object-lock-legal-hold`

Using S3 Object Lock

- SSE request headers:

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

S3 REST API supported operations and limitations



For information on how StorageGRID handles UTF-8 characters, see the documentation for PUT Object.

Request headers for server-side encryption

You can use the following request headers to encrypt a multipart object with server-side encryption. The SSE and SSE-C options are mutually exclusive.

- **SSE**: Use the following header in the Initiate Multipart Upload request if you want to encrypt the object with a unique key managed by StorageGRID. Do not specify this header in any of the Upload Part requests.
 - `x-amz-server-side-encryption`
- **SSE-C**: Use all three of these headers in the Initiate Multipart Upload request (and in each subsequent Upload Part request) if you want to encrypt the object with a unique key that you provide and manage.
 - `x-amz-server-side-encryption-customer-algorithm`: Specify AES256.

- `x-amz-server-side-encryption-customer-key`: Specify your encryption key for the new object.
- `x-amz-server-side-encryption-customer-key-MD5`: Specify the MD5 digest of the new object's encryption key.

Attention: The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object. Before using customer-provided keys to secure object data, review the considerations in “Using server-side encryption.”

Unsupported request headers

The following request header is not supported and returns `XNotImplemented`

- `x-amz-website-redirect-location`

Versioning

Multipart upload consists of separate operations for initiating the upload, listing uploads, uploading parts, assembling the uploaded parts, and completing the upload. Objects are created (and versioned if applicable) when the Complete Multipart Upload operation is performed.

Related information

[Manage objects with ILM](#)

[Using server-side encryption](#)

[PUT Object](#)

Upload Part

The Upload Part operation uploads a part in a multipart upload for an object.

Supported request headers

The following request headers are supported:

- `Content-Length`
- `Content-MD5`

Request headers for server-side encryption

If you specified SSE-C encryption for the Initiate Multipart Upload request, you must also include the following request headers in each Upload Part request:

- `x-amz-server-side-encryption-customer-algorithm`: Specify `AES256`.
- `x-amz-server-side-encryption-customer-key`: Specify the same encryption key that you provided in the Initiate Multipart Upload request.
- `x-amz-server-side-encryption-customer-key-MD5`: Specify the same MD5 digest that you provided in the Initiate Multipart Upload request.



The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object. Before using customer-provided keys to secure object data, review the considerations in “Using server-side encryption.”

Versioning

Multipart upload consists of separate operations for initiating the upload, listing uploads, uploading parts, assembling the uploaded parts, and completing the upload. Objects are created (and versioned if applicable) when the Complete Multipart Upload operation is performed.

Related information

[Using server-side encryption](#)

Upload Part - Copy

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

The Upload Part - Copy operation is implemented with all Amazon S3 REST API behavior.

This request reads and writes the object data specified in `x-amz-copy-source-range` within the StorageGRID system.

The following request headers are supported:

- `x-amz-copy-source-if-match`
- `x-amz-copy-source-if-none-match`
- `x-amz-copy-source-if-unmodified-since`
- `x-amz-copy-source-if-modified-since`

Request headers for server-side encryption

If you specified SSE-C encryption for the Initiate Multipart Upload request, you must also include the following request headers in each Upload Part - Copy request:

- `x-amz-server-side-encryption-customer-algorithm`: Specify AES256.
- `x-amz-server-side-encryption-customer-key`: Specify the same encryption key that you provided in the Initiate Multipart Upload request.
- `x-amz-server-side-encryption-customer-key-MD5`: Specify the same MD5 digest that you provided in the Initiate Multipart Upload request.

If the source object is encrypted using a customer-provided key (SSE-C), you must include the following three headers in the Upload Part - Copy request, so the object can be decrypted and then copied:

- `x-amz-copy-source-server-side-encryption-customer-algorithm`: Specify AES256.
- `x-amz-copy-source-server-side-encryption-customer-key`: Specify the encryption key you provided when you created the source object.
- `x-amz-copy-source-server-side-encryption-customer-key-MD5`: Specify the MD5 digest you provided when you created the source object.



The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object. Before using customer-provided keys to secure object data, review the considerations in “Using server-side encryption.”

Versioning

Multipart upload consists of separate operations for initiating the upload, listing uploads, uploading parts, assembling the uploaded parts, and completing the upload. Objects are created (and versioned if applicable) when the Complete Multipart Upload operation is performed.

Complete Multipart Upload

The Complete Multipart Upload operation completes a multipart upload of an object by assembling the previously uploaded parts.

Resolving conflicts

Conflicting client requests, such as two clients writing to the same key, are resolved on a “latest-wins” basis. The timing for the “latest-wins” evaluation is based on when the StorageGRID system completes a given request, and not on when S3 clients begin an operation.

Object size

StorageGRID supports objects up to 5 TB in size.

Request headers

The `x-amz-storage-class` request header is supported, and affects how many object copies StorageGRID creates if the matching ILM rule specifies an Ingest Behavior of Dual commit or Balanced.

- STANDARD

(Default) Specifies a dual-commit ingest operation when the ILM rule uses the Dual commit option, or when the Balanced option falls back to creating interim copies.

- REDUCED_REDUNDANCY

Specifies a single-commit ingest operation when the ILM rule uses the Dual commit option, or when the Balanced option falls back to creating interim copies.



If you are ingesting an object into a bucket with S3 Object Lock enabled, the REDUCED_REDUNDANCY option is ignored. If you are ingesting an object into a legacy Compliant bucket, the REDUCED_REDUNDANCY option returns an error. StorageGRID will always perform a dual-commit ingest to ensure that compliance requirements are satisfied.



If a multipart upload is not completed within 15 days, the operation is marked as inactive and all associated data is deleted from the system.



The `ETag` value returned is not an MD5 sum of the data, but follows the Amazon S3 API implementation of the `ETag` value for multipart objects.

Versioning

This operation completes a multipart upload. If versioning is enabled for a bucket, the object version is created upon completion of the multipart upload.

If versioning is enabled for a bucket, a unique `versionId` is automatically generated for the version of the object being stored. This `versionId` is also returned in the response using the `x-amz-version-id` response header.

If versioning is suspended, the object version is stored with a null `versionId` and if a null version already exists it will be overwritten.

 When versioning is enabled for a bucket, completing a multipart upload always creates a new version, even if there are concurrent multipart uploads completed on the same object key. When versioning is not enabled for a bucket, it is possible to initiate a multipart upload and then have another multipart upload initiate and complete first on the same object key. On non-versioned buckets, the multipart upload that completes last takes precedence.

Failed replication, notification, or metadata notification

If the bucket where the multipart upload occurs is configured for a platform service, multipart upload succeeds even if the associated replication or notification action fails.

If this occurs, an alarm is raised in the Grid Manager on Total Events (SMTT). The Last Event message displays “Failed to publish notifications for bucket-nameobject key” for the last object whose notification failed. (To see this message, select **Nodes > Storage Node > Events**. View Last Event at the top of the table.) Event messages are also listed in `/var/local/log/bycast-err.log`.

A tenant can trigger the failed replication or notification by updating the object’s metadata or tags. A tenant can resubmit the existing values to avoid making unwanted changes.

Related information

[Manage objects with ILM](#)

Error responses

The StorageGRID system supports all standard S3 REST API error responses that apply. In addition, the StorageGRID implementation adds several custom responses.

Supported S3 API error codes

Name	HTTP status
AccessDenied	403 Forbidden
BadDigest	400 Bad Request
BucketAlreadyExists	409 Conflict
BucketNotEmpty	409 Conflict

Name	HTTP status
IncompleteBody	400 Bad Request
InternalError	500 Internal Server Error
InvalidAccessKeyId	403 Forbidden
InvalidArgument	400 Bad Request
InvalidBucketName	400 Bad Request
InvalidBucketState	409 Conflict
InvalidDigest	400 Bad Request
InvalidEncryptionAlgorithmError	400 Bad Request
InvalidPart	400 Bad Request
InvalidPartOrder	400 Bad Request
InvalidRange	416 Requested Range Not Satisfiable
InvalidRequest	400 Bad Request
InvalidStorageClass	400 Bad Request
InvalidTag	400 Bad Request
InvalidURI	400 Bad Request
KeyTooLong	400 Bad Request
MalformedXML	400 Bad Request
MetadataTooLarge	400 Bad Request
MethodNotAllowed	405 Method Not Allowed
MissingContentLength	411 Length Required
MissingRequestBodyError	400 Bad Request
MissingSecurityHeader	400 Bad Request

Name	HTTP status
NoSuchBucket	404 Not Found
NoSuchKey	404 Not Found
NoSuchUpload	404 Not Found
NotImplemented	501 Not Implemented
NoSuchBucketPolicy	404 Not Found
ObjectLockConfigurationNotFoundError	404 Not Found
PreconditionFailed	412 Precondition Failed
RequestTimeTooSkewed	403 Forbidden
ServiceUnavailable	503 Service Unavailable
SignatureDoesNotMatch	403 Forbidden
TooManyBuckets	400 Bad Request
UserKeyMustBeSpecified	400 Bad Request

StorageGRID custom error codes

Name	Description	HTTP status
XBucketLifecycleNotAllowed	Bucket lifecycle configuration is not allowed in a legacy Compliant bucket	400 Bad Request
XBucketPolicyParseException	Failed to parse received bucket policy JSON.	400 Bad Request
XComplianceConflict	Operation denied because of legacy Compliance settings.	403 Forbidden
XComplianceReducedRedundancyForbidden	Reduced redundancy is not allowed in legacy Compliant bucket	400 Bad Request
XMaxBucketPolicyLengthExceeded	Your policy exceeds the maximum allowed bucket policy length.	400 Bad Request

Name	Description	HTTP status
XMissingInternalRequestHeader	Missing a header of an internal request.	400 Bad Request
XNoSuchBucketCompliance	The specified bucket does not have legacy Compliance enabled.	404 Not Found
XNotAcceptable	The request contains one or more accept headers that could not be satisfied.	406 Not Acceptable
XNotImplemented	The request you provided implies functionality that is not implemented.	501 Not Implemented

Copyright information

Copyright © 2025 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

LIMITED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

Trademark information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.