



Operations on objects

StorageGRID

NetApp

October 03, 2025

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

Table of Contents

Operations on objects	1
Using S3 Object Lock	5
Enabling S3 Object Lock for a bucket	5
Determining if S3 Object Lock is enabled for a bucket	5
Creating an object with S3 Object Lock settings	5
Updating S3 Object Lock settings	6
Using server-side encryption	7
Using SSE	7
Using SSE-C	8
Considerations for using server-side encryption with customer-provided keys (SSE-C)	8
GET Object	9
partNumber request parameter is not supported	9
Request headers for server-side encryption with customer-provided encryption keys (SSE-C)	9
UTF-8 characters in user metadata	10
Unsupported request header	10
Versioning	10
Behavior of GET Object for Cloud Storage Pool objects	10
Multipart or segmented objects in a Cloud Storage Pool	11
HEAD Object	11
Request headers for server-side encryption with customer-provided encryption keys (SSE-C)	11
UTF-8 characters in user metadata	12
Unsupported request header	12
Response headers for Cloud Storage Pool objects	12
Multipart or segmented objects in a Cloud Storage Pool	13
Versioning	14
POST Object restore	14
Supported request type	14
Versioning	14
Behavior of POST Object restore on Cloud Storage Pool objects	14
PUT Object	15
Resolving conflicts	16
Object size	16
User metadata size	16
UTF-8 characters in user metadata	16
Object tag limits	16
Object ownership	16
Supported request headers	16
Unsupported request headers	18
Storage class options	18
Request headers for server-side encryption	19
Versioning	19
PUT Object - Copy	20
Resolving conflicts	20

Object size	20
UTF-8 characters in user metadata	20
Supported request headers	20
Unsupported request headers	21
Storage class options	21
Using x-amz-copy-source in PUT Object - Copy	22
Request headers for server-side encryption	22
Versioning	23

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.

State of object	Behavior of GET Object
Object in process of being restored from a non-retrievable state	403 Forbidden, InvalidObjectState 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	200 OK <code>x-amz-storage-class: GLACIER</code> <code>x-amz-restore: ongoing-request="false", expiry-date="Sat, 23 July 20 2030 00:00:00 GMT"</code> 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.

State of object	Response to HEAD object
Object has transitioned to non-retrievable state, but at least one copy also exists on the grid	<p>200 OK</p> <p>x-amz-storage-class: GLACIER</p> <p>x-amz-restore: ongoing-request="false", expiry-date="Sat, 23 July 20 2030 00:00:00 GMT"</p> <p>The value for expiry-date 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>
Object transitioned to a non-retrievable state, and no copy exists on the grid	<p>200 OK</p> <p>x-amz-storage-class: GLACIER</p>
Object in process of being restored from a non-retrievable state	<p>200 OK</p> <p>x-amz-storage-class: GLACIER</p> <p>x-amz-restore: ongoing-request="true"</p>
Object fully restored to the Cloud Storage Pool	<p>200 OK</p> <p>x-amz-storage-class: GLACIER</p> <p>x-amz-restore: ongoing-request="false", expiry-date="Sat, 23 July 20 2018 00:00:00 GMT"</p> <p>The expiry-date indicates when the object in the Cloud Storage Pool will be returned to a non-retrievable state.</p>

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 <code>Forbidden</code> , <code>InvalidObjectState</code>

State of object	Behavior of POST Object restore
Object in Cloud Storage Pool but not yet transitioned to a non-retrievable state	<p>200 OK No changes are made.</p> <p>Note: Before an object has been transitioned to a non-retrievable state, you cannot change its expiry-date.</p>
Object transitioned to a non-retrievable state	<p>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.</p> <p>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.</p> <p>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, <code>InvalidTier: Retrieval</code> option is not supported by this storage class.</p>
Object in process of being restored from a non-retrievable state	409 Conflict, <code>RestoreAlreadyInProgress</code>
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 <code>Days</code>. 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 (unescaped) 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 **XNot Implemented**.
- 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](#)

[PUT Object](#)

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—with 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.