Operations on objects

StorageGRID

NetApp
March 02, 2022

Table of Contents

Operations on objects

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

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
  - SELECT Object content
- 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	Multi-Factor Authentication (MFA) and the response header `x-amz-mfa` are not supported. 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. **Versioning** To remove a specific version, the requestor must be the bucket owner and use the `versionId` subresource. Using this subresource permanently deletes the version. If the `versionId` corresponds to a delete marker, the response header `x-amz-delete-marker` is returned set to `true`. • If an object is deleted without the `versionId` subresource on a version enabled bucket, it results in the generation of a delete marker. The `versionId` for the delete marker is returned using the `x-amz-version-id` response header, and the `x-amz-delete-marker` response header is returned set to `true`. • If an object is deleted without the `versionId` 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 `x-amz-delete-marker` response header is returned set to `true`. **Note**: In certain cases, multiple delete markers might exist for an object.
DELETE Multiple Objects	Multi-Factor Authentication (MFA) and the response header `x-amz-mfa` are not supported. Multiple objects can be deleted in the same request message.
DELETE Object tagging	Uses the `tagging` subresource to remove all tags from an object. Implemented with all Amazon S3 REST API behavior. **Versioning** If the `versionId` 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 `x-amz-delete-marker` response header set to `true`.
GET Object	GET Object

Operation	Implementation
GET Object ACL	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.
GET Object legal hold	Use S3 Object Lock
GET Object retention	Use S3 Object Lock
GET Object tagging	Uses the `tagging` subresource to return all tags for an object. Implemented with all Amazon S3 REST API behavior **Versioning** If the `versionId` 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 `x-amz-delete-marker` response header set to `true`.
HEAD Object	HEAD Object
POST Object restore	POST Object restore
PUT Object	PUT Object
PUT Object - Copy	PUT Object - Copy
PUT Object legal hold	Use S3 Object Lock
PUT Object retention	Use S3 Object Lock

Operation	Implementation
PUT Object tagging	Uses the `tagging` subresource to add a set of tags to an existing object. Implemented with all Amazon S3 REST API behavior **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. **Tag updates and ingest behavior** 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. 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. **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. **Versioning** If the `versionId` 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 `x-amz-delete-marker` response header set to `true`.

Related information

S3 operations tracked in audit logs

Use 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 default retention periods for each bucket or specific 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.

Enable S3 Object Lock for 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 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 supports default retention periods for the objects in S3 Object Lock buckets and supports the PUT Object Lock Configuration bucket operation. The `s3:object-lock-remaining-retention-days` policy condition key sets the minimum and maximum allowable retention periods for your objects.

Determining if S3 Object Lock is enabled for bucket

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

Create 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.

Update 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 tenant account

PUT Object

PUT Object - Copy

Initiate Multipart Upload

Object versioning

Amazon Simple Storage Service User Guide: Using S3 Object Lock

Use S3 Select

StorageGRID supports the following AWS S3 Select clauses, data types, and operators for the SelectObjectContent command.

Any items not listed are not supported.

For syntax, see SelectObjectContent. For more information about S3 Select, see the AWS documentation for S3 Select.

Only tenant accounts that have S3 Select enabled can issue SelectObjectContent queries. See the considerations and requirements for using S3 Select.

Clauses

- SELECT list
- FROM clause
- WHERE clause
- LIMIT clause

Data types

- bool
- integer
- string
- float
- decimal, numeric
- timestamp

Operators

Logical operators

- AND
- NOT
- OR

Comparison operators

- <
- >

- <=
- >=
- =
- =
- <>
- !=
- BETWEEN
- IN

Pattern matching operators

- LIKE
- _
- %

Unitary operators

- IS NULL
- IS NOT NULL

Math operators

- +
- -
- *
- /
- %

StorageGRID follows the AWS S3 Select operator precedence.

Aggregate functions

- AVG()
- COUNT(*)
- MAX()
- MIN()
- SUM()

Conditional functions

- CASE
- COALESCE
- NULLIF

Conversion functions

- CAST (for supported datatype)

Date functions

- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

String functions

- CHAR_LENGTH, CHARACTER_LENGTH
- LOWER
- SUBSTRING
- TRIM
- UPPER

Use 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.

Use 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

Use 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.

GET object and multipart objects

You can use the `partNumber` request parameter to retrieve a specific part of a multipart or segmented object. The `x-amz-mp-parts-count` response element indicates how many parts the object has.

You can set `partNumber` to 1 for both segmented/multipart objects and non-segmented/non-multipart objects; however, the `x-amz-mp-parts-count` response element is only returned for segmented or multipart objects.

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 "Use 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.

State of object	Behavior of GET Object
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, InvalidObjectState` 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, 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

Use server-side encryption

Manage objects with ILM

POST Object restore

S3 operations tracked in 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.

HEAD object and multipart objects

You can use the `partNumber` request parameter to retrieve metadata for a specific part of a multipart or segmented object. The `x-amz-mp-parts-count` response element indicates how many parts the object has.

You can set `partNumber` to 1 for both segmented/multipart objects and non-segmented/non-multipart objects; however, the `x-amz-mp-parts-count` response element is only returned for segmented or multipart objects.

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 "Use 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`
	
	`x-amz-storage-class: GLACIER`
	
	`x-amz-restore: ongoing-request="false", expiry-date="Sat, 23 July 20 2030 00:00:00 GMT"`
	
	Until the object is transitioned to a non-retrievable state, the value for `expiry-date` is set to some distant time in the future. The exact time of transition is not controlled by the StorageGRID system.

Object has transitioned to non-retrievable state, but at least one copy also exists on the grid	`200 OK`
	
	`x-amz-storage-class: GLACIER`
	
	`x-amz-restore: ongoing-request="false", expiry-date="Sat, 23 July 20 2030 00:00:00 GMT"`
	
	The value for `expiry-date` is set to some distant time in the future.
	
	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.

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"`

State of object	Response to HEAD object

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 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

Use server-side encryption

Manage objects with ILM

POST Object restore

S3 operations tracked in 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 `Tier` request element to determine how long the restore job will take to finish (`Expedited`, `Standard`, or `Bulk`). If you do not specify `Tier`, 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`.
Object in process of being restored from a non-retrievable state	`409 Conflict`, `RestoreAlreadyInProgress`

State of object	Behavior of POST Object restore
Object fully restored to the Cloud Storage Pool	`200 OK` **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.

Related information

Manage objects with ILM

HEAD Object

S3 operations tracked in audit logs

PUT Object

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

Resolve 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

The maximum *recommended* size for a single PUT Object operation is 5 GiB (5,368,709,120 bytes). If you have objects that are larger than 5 GiB, use multipart upload instead.

> In StorageGRID 11.6, the maximum *supported* size for a single PUT Object operation is 5 TiB (5,497,558,138,880 bytes). However, the **S3 PUT Object size too large** alert will be triggered if you attempt to upload an object that exceeds 5 GiB.

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 (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:

- 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-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.

> 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`

    If a request is made without these headers, the bucket default retention settings are used to calculate the object version retain-until-date.

    Use 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`

    See Request headers for server-side encryption

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 "Use 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 audit logs

Use 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.

Resolve 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

The maximum *recommended* size for a single PUT Object operation is 5 GiB (5,368,709,120 bytes). If you have objects that are larger than 5 GiB, use multipart upload instead.

In StorageGRID 11.6, the maximum *supported* size for a single PUT Object operation is 5 TiB (5,497,558,138,880 bytes). However, the **S3 PUT Object size too large** alert will be triggered if you attempt to upload an object that exceeds 5 GiB.

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`

    If a request is made without these headers, the bucket default retention settings are used to calculate the object version retain-until-date.

Use 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`

  See 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 "Use 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

Use server-side encryption

S3 operations tracked in audit logs

PUT Object

SelectObjectContent

You can use the S3 SelectObjectContent request to filter the contents of an S3 object based on a simple SQL statement.

For more information see the AWS documentation for SelectObjectContent.

What you'll need

- The tenant account has the S3 Select permission.
- You have `s3:GetObject` permission for the object you want to query.
- The object you want to query is in CSV format, or is a GZIP or BZIP2 compressed file containing a CSV formatted file.
- Your SQL expression has a maximum length of 256 KB.
- Any record in the input or results has a maximum length of 1 MiB.

Request syntax example

```
POST /{Key+}?select&select-type=2 HTTP/1.1
Host: Bucket.s3.abc-company.com
x-amz-expected-bucket-owner: ExpectedBucketOwner
<?xml version="1.0" encoding="UTF-8"?>
<SelectObjectContentRequest xmlns="http://s3.amazonaws.com/doc/2006-03-
01/">
   <Expression>string</Expression>
   <ExpressionType>string</ExpressionType>
   <RequestProgress>
      <Enabled>boolean</Enabled>
   </RequestProgress>
   <InputSerialization>
      <CompressionType>GZIP</CompressionType>
      <CSV>
         <AllowQuotedRecordDelimiter>boolean</AllowQuotedRecordDelimiter>
         <Comments>#</Comments>
         <FieldDelimiter>\t</FieldDelimiter>
         <FileHeaderInfo>USE</FileHeaderInfo>
         <QuoteCharacter>'</QuoteCharacter>
         <QuoteEscapeCharacter>\\</QuoteEscapeCharacter>
         <RecordDelimiter>\n</RecordDelimiter>
      </CSV>
   </InputSerialization>
   <OutputSerialization>
      <CSV>
         <FieldDelimiter>string</FieldDelimiter>
         <QuoteCharacter>string</QuoteCharacter>
         <QuoteEscapeCharacter>string</QuoteEscapeCharacter>
         <QuoteFields>string</QuoteFields>
         <RecordDelimiter>string</RecordDelimiter>
      </CSV>
   </OutputSerialization>
   <ScanRange>
      <End>long</End>
      <Start>long</Start>
   </ScanRange>
</SelectObjectContentRequest>
```

SQL query example

This query gets the state name, 2010 populations, estimated 2015 populations, and the percentage of change from US census data. Records in the file that are not states are ignored.

```
SELECT STNAME, CENSUS2010POP, POPESTIMATE2015, CAST((POPESTIMATE2015 -
CENSUS2010POP) AS DECIMAL) / CENSUS2010POP * 100.0 FROM S3Object WHERE
NAME = STNAME
```

The first few lines of the file to be queried, `SUB-EST2020_ALL.csv`, look like this:

```
SUMLEV,STATE,COUNTY,PLACE,COUSUB,CONCIT,PRIMGEO_FLAG,FUNCSTAT,NAME,STNAME,
CENSUS2010POP,
ESTIMATESBASE2010,POPESTIMATE2010,POPESTIMATE2011,POPESTIMATE2012,POPESTIM
ATE2013,POPESTIMATE2014,
POPESTIMATE2015,POPESTIMATE2016,POPESTIMATE2017,POPESTIMATE2018,POPESTIMAT
E2019,POPESTIMATE042020,
POPESTIMATE2020
040,01,000,00000,00000,00000,0,A,Alabama,Alabama,4779736,4780118,4785514,4
799642,4816632,4831586,
4843737,4854803,4866824,4877989,4891628,4907965,4920706,4921532
162,01,000,00124,00000,00000,0,A,Abbeville
city,Alabama,2688,2705,2699,2694,2645,2629,2610,2602,
2587,2578,2565,2555,2555,2553
162,01,000,00460,00000,00000,0,A,Adamsville
city,Alabama,4522,4487,4481,4474,4453,4430,4399,4371,
4335,4304,4285,4254,4224,4211
162,01,000,00484,00000,00000,0,A,Addison
town,Alabama,758,754,751,750,745,744,742,734,734,728,
725,723,719,717
```

AWS-CLI usage example

```
aws s3api select-object-content --endpoint-url https://10.224.7.44:10443
--no-verify-ssl  --bucket 619c0755-9e38-42e0-a614-05064f74126d --key SUB-
EST2020_ALL.csv --expression-type SQL --input-serialization '{"CSV":
{"FileHeaderInfo": "USE", "Comments": "#", "QuoteEscapeCharacter": "\"",
"RecordDelimiter": "\n", "FieldDelimiter": ",", "QuoteCharacter": "\"",
"AllowQuotedRecordDelimiter": false}, "CompressionType": "NONE"}' --output
-serialization '{"CSV": {"QuoteFields": "ASNEEDED",
"QuoteEscapeCharacter": "#", "RecordDelimiter": "\n", "FieldDelimiter":
",", "QuoteCharacter": "\""}}' --expression "SELECT STNAME, CENSUS2010POP,
POPESTIMATE2015, CAST((POPESTIMATE2015 - CENSUS2010POP) AS DECIMAL) /
CENSUS2010POP * 100.0 FROM S3Object WHERE NAME = STNAME" changes.csv
```

The first few lines of the output file, `changes.csv`, look like this:

```
Alabama,4779736,4854803,1.57052607089596580229535689837262978 54
Alaska,710231,738430,3.97039836334938914240578065446312537 75
Arizona,6392017,6832810,6.89599229789282475312565658070058324 31
Arkansas,2915918,2979732,2.18847032049598102552952449280123789 49
California,37253956,38904296,4.42997248399606205579885261044491489 71
Colorado,5029196,5454328,8.45327960970302211327615785902955462 46
```