NetApp

Support for Amazon S3 REST API

StorageGRID software

NetApp
December 03, 2025

Table of Contents

Support for Amazon S3 REST API

S3 REST API implementation details

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

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

Date handling

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

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

Common request headers

The StorageGRID system supports the common request headers defined by Amazon Simple Storage Service API Reference: Common Request Headers, with one exception.

Request header	Implementation
Authorization	Full support for AWS Signature Version 2 Support for AWS Signature Version 4, with the following exceptions: • When you provide the actual payload checksum value in `x-amz-content-sha256`, the value is accepted without validation, as if the value `UNSIGNED-PAYLOAD` had been provided for the header. When you provide an `x-amz-content-sha256` header value that implies `aws-chunked` streaming (for example, STREAMING-AWS4-HMAC-SHA256-PAYLOAD), the chunk signatures aren't verified against the chunk data.
x-amz-security-token	Not implemented. Returns `XNotImplemented`.

Common response headers

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

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

Authenticate requests

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

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

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

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

Use the HTTP Authorization header

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

Use query parameters

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

Operations on the service

The StorageGRID system supports the following operations on the service.

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

Operations on buckets

The StorageGRID system supports a maximum of 5,000 buckets for each S3 tenant account.

Each grid can have a maximum of 100,000 buckets.

To support 5,000 buckets, each Storage Node in the grid must have a minimum of 64 GB of RAM.

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

See the following for more information:

- Amazon Simple Storage Service User Guide: Bucket quotas, restrictions, and limitations
- Configure S3 endpoint domain names

The ListObjects (GET Bucket) and ListObjectVersions (GET Bucket object versions) operations support StorageGRID consistency values.

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

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

Operation	Implementation

CreateBucket	Creates a new bucket. By creating the bucket, you become the bucket owner.
	
	• Bucket names must comply with the following rules:
	  ◦ Must be unique across each StorageGRID system (not just unique within the tenant account).
	  ◦ Must be DNS compliant.
	  ◦ Must contain at least 3 and no more than 63 characters.
	  ◦ Can be a series of one or more labels, with adjacent labels separated by a period. Each label must start and end with a lowercase letter or a number and can only use lowercase letters, numbers, and hyphens.
	  ◦ Must not look like a text-formatted IP address.
	  ◦ Should not use periods in virtual hosted style requests. Periods will cause problems with server wildcard certificate verification.
	
	• By default, buckets are created in the `us-east-1` region; however, you can use the `LocationConstraint` request element in the request body to specify a different region. When using the `LocationConstraint` element, you must specify the exact name of a region that has been defined using the Grid Manager or the Grid Management API. Contact your system administrator if you don't know the region name you should use.
	
	Note: An error will occur if your CreateBucket request uses a region that has not been defined in StorageGRID.
	
	• You can include the `x-amz-bucket-object-lock-enabled` request header to create a bucket with S3 Object Lock enabled. See Use S3 REST API to configure S3 Object Lock.
	
	You must enable S3 Object Lock when you create the bucket. You can't add or disable S3 Object Lock after a bucket is created. S3 Object Lock requires bucket versioning, which is enabled automatically when you create the bucket.

DeleteBucket	Deletes the bucket.
DeleteBucketCors	Deletes the CORS configuration for the bucket.
DeleteBucketEncryption	Deletes the default encryption from the bucket. Existing encrypted objects remain encrypted, but any new objects added to the bucket aren't encrypted.
DeleteBucketLifecycle	Deletes the lifecycle configuration from the bucket. See Create S3 lifecycle configuration.
DeleteBucketPolicy	Deletes the policy attached to the bucket.
DeleteBucketReplication	Deletes the replication configuration attached to the bucket.

Operation	Implementation
DeleteBucketTagging	Uses the `tagging` subresource to remove all tags from a bucket. **Caution**: If a non-default ILM policy tag is set for this bucket, there will be a `NTAP-SG-ILM-BUCKET-TAG` bucket tag with a value assigned to it. Do not issue a DeleteBucketTagging request if there is a `NTAP-SG-ILM-BUCKET-TAG` bucket tag. Instead, issue a PutBucketTagging request with only the `NTAP-SG-ILM-BUCKET-TAG` tag and its assigned value to remove all other tags from the bucket. Do not modify or remove the `NTAP-SG-ILM-BUCKET-TAG` bucket tag.
GetBucketAcl	Returns a positive response and the ID, DisplayName, and Permission of the bucket owner, indicating that the owner has full access to the bucket.
GetBucketCors	Returns the `cors` configuration for the bucket.
GetBucketEncryption	Returns the default encryption configuration for the bucket.
GetBucketLifecycleConfiguration (previously named GET Bucket lifecycle)	Returns the lifecycle configuration for the bucket. See Create S3 lifecycle configuration.
GetBucketLocation	Returns the region that was set using the `LocationConstraint` element in the CreateBucket request. If the bucket's region is `us-east-1`, an empty string is returned for the region.
GetBucketNotificationConfiguration (previously named GET Bucket notification)	Returns the notification configuration attached to the bucket.
GetBucketPolicy	Returns the policy attached to the bucket.
GetBucketReplication	Returns the replication configuration attached to the bucket.
GetBucketTagging	Uses the `tagging` subresource to return all tags for a bucket. **Caution**: If a non-default ILM policy tag is set for this bucket, there will be a `NTAP-SG-ILM-BUCKET-TAG` bucket tag with a value assigned to it. Do not modify or remove this tag.

Operation	Implementation
GetBucketVersioning	This implementation uses the `versioning` subresource to return the versioning state of a bucket. • *blank*: Versioning has never been enabled (bucket is "Unversioned") • Enabled: Versioning is enabled • Suspended: Versioning was previously enabled and is suspended
GetObjectLockConfiguration	Returns the bucket default retention mode and default retention period, if configured. See Use S3 REST API to configure S3 Object Lock.
HeadBucket	Determines if a bucket exists and you have permission to access it. This operation returns: • `x-ntap-sg-bucket-id`: The UUID of the bucket in UUID format. • `x-ntap-sg-trace-id`: The unique trace ID of the associated request.
ListObjects and ListObjectsV2 (previously named GET Bucket)	Returns some or all (up to 1,000) of the objects in a bucket. The Storage Class for objects can have either of two values, even if the object was ingested with the `REDUCED_REDUNDANCY` storage class option: • `STANDARD`, which indicates the object is stored in a storage pool consisting of Storage Nodes. • `GLACIER`, which indicates that the object has been moved to the external bucket specified by the Cloud Storage Pool. If the bucket contains large numbers of deleted keys that have the same prefix, the response might include some `CommonPrefixes` that don't contain keys.
ListObjectVersions (previously named GET Bucket Object versions)	With READ access on a bucket, using this operation with the `versions` subresource lists metadata of all of the versions of objects in the bucket.
PutBucketCors	Sets the CORS configuration for a bucket so that the bucket can service cross-origin requests. Cross-origin resource sharing (CORS) is a security mechanism that allows client web applications in one domain to access resources in a different domain. For example, suppose you use an S3 bucket named `images` to store graphics. By setting the CORS configuration for the `images` bucket, you can allow the images in that bucket to be displayed on the website `http://www.example.com`.

Operation	Implementation
PutBucketEncryption	Sets the default encryption state of an existing bucket. When bucket-level encryption is enabled, any new objects added to the bucket are encrypted.StorageGRID supports server-side encryption with StorageGRID-managed keys. When specifying the server-side encryption configuration rule, set the `SSEAlgorithm` parameter to `AES256`, and don't use the `KMSMasterKeyID` parameter. Bucket default encryption configuration is ignored if the object upload request already specifies encryption (that is, if the request includes the `x-amz-server-side-encryption-*` request header).
PutBucketLifecycleConfiguration (previously named PUT Bucket lifecycle)	Creates a new lifecycle configuration for the bucket or replaces an existing lifecycle configuration. StorageGRID supports up to 1,000 lifecycle rules in a lifecycle configuration. Each rule can include the following XML elements: • Expiration (Days, Date, ExpiredObjectDeleteMarker) • NoncurrentVersionExpiration (NewerNoncurrentVersions, NoncurrentDays) • Filter (Prefix, Tag) • Status • ID StorageGRID does not support these actions: • AbortIncompleteMultipartUpload • Transition See Create S3 lifecycle configuration. To understand how the Expiration action in a bucket lifecycle interacts with ILM placement instructions, see How ILM operates throughout an object's life. **Note**: Bucket lifecycle configuration can be used with buckets that have S3 Object Lock enabled, but bucket lifecycle configuration is not supported for legacy Compliant buckets.

Operation	Implementation

PutBucketNotificationConfiguration
 	
(previously named PUT Bucket notification)
	Configures notifications for the bucket using the notification configuration XML included in the request body. You should be aware of the following implementation details:
	
	• StorageGRID supports Amazon Simple Notification Service (Amazon SNS) or Kafka topics as destinations. Simple Queue Service (SQS) or Amazon Lambda endpoints aren't supported.
	
	• The destination for notifications must be specified as the URN of an StorageGRID endpoint. Endpoints can be created using the Tenant Manager or the Tenant Management API.
	
	The endpoint must exist for notification configuration to succeed. If the endpoint does not exist, a `400 Bad Request` error is returned with the code `InvalidArgument`.
	
	• You can't configure a notification for the following event types. These event types are **not** supported.
	
	  ◦ `s3:ReducedRedundancyLostObject`
	
	  ◦ `s3:ObjectRestore:Completed`
	
	• Event notifications sent from StorageGRID use the standard JSON format except that they don't include some keys and use specific values for others, as shown in the following list:
	
	  ◦ **eventSource**
	
	    `sgws:s3`
	
	  ◦ **awsRegion**
	
	    not included
	
	  ◦ **x-amz-id-2**
	
	    not included
	
	  ◦ **arn**
	
	    `urn:sgws:s3:::bucket_name`

PutBucketPolicy	Sets the policy attached to the bucket. See Use bucket and group access policies.

Operation	Implementation

PutBucketReplication	Configures StorageGRID CloudMirror replication for the bucket using the replication configuration XML provided in the request body. For CloudMirror replication, you should be aware of the following implementation details:
	
	 • StorageGRID only supports V1 of the replication configuration. This means that StorageGRID does not support the use of the `Filter` element for rules, and follows V1 conventions for deletion of object versions. For details, see Amazon Simple Storage Service User Guide: Replication configuration.
	
	 • Bucket replication can be configured on versioned or unversioned buckets.
	
	 • You can specify a different destination bucket in each rule of the replication configuration XML. A source bucket can replicate to more than one destination bucket.
	
	 • Destination buckets must be specified as the URN of StorageGRID endpoints as specified in the Tenant Manager or the Tenant Management API. See Configure CloudMirror replication.
	
	 The endpoint must exist for replication configuration to succeed. If the endpoint does not exist, the request fails as a `400 Bad Request`. The error message states: `Unable to save the replication policy. The specified endpoint URN does not exist:` *URN*.
	
	 • You don't need to specify a `Role` in the configuration XML. This value is not used by StorageGRID and will be ignored if submitted.
	
	 • If you omit the storage class from the configuration XML, StorageGRID uses the `STANDARD` storage class by default.
	
	 • If you delete an object from the source bucket or you delete the source bucket itself, the cross-region replication behavior is as follows:
	   ◦ If you delete the object or bucket before it has been replicated, the object/bucket is not replicated and you aren't notified.
	   ◦ If you delete the object or bucket after it has been replicated, StorageGRID follows standard Amazon S3 delete behavior for V1 of cross-region replication.

Operation	Implementation
PutBucketTagging	Uses the `tagging` subresource to add or update a set of tags for a bucket. When adding bucket tags, be aware of the following limitations: • Both StorageGRID and Amazon S3 support up to 50 tags for each bucket. • Tags associated with a bucket must have unique tag keys. A tag key can be up to 128 Unicode characters in length. • Tag values can be up to 256 Unicode characters in length. • Key and values are case sensitive. **Caution**: If a non-default ILM policy tag is set for this bucket, there will be a `NTAP-SG-ILM-BUCKET-TAG` bucket tag with a value assigned to it. Make sure that the `NTAP-SG-ILM-BUCKET-TAG` bucket tag is included with the assigned value in all PutBucketTagging requests. Do not modify or remove this tag. **Note**: This operation will overwrite any current tags the bucket already has. If any existing tags are omitted from the set, those tags will be removed for the bucket.
PutBucketVersioning	Uses the `versioning` subresource to set the versioning state of an existing bucket. You can set the versioning state with one of the following values: • Enabled: Enables versioning for the objects in the bucket. All objects added to the bucket receive a unique version ID. • Suspended: Disables versioning for the objects in the bucket. All objects added to the bucket receive the version ID `null`.
PutObjectLockConfiguration	Configures or removes the bucket default retention mode and default retention period. If the default retention period is modified, the retain-until-date of existing object versions remains the same and is not recalculated using the new default retention period. See Use S3 REST API to configure S3 Object Lock for detailed information.

Operations on objects

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 values are supported by all operations on objects, with the exception of the following:
   ◦ GetObjectAcl

- ◦ `OPTIONS /`
  - ◦ PutObjectLegalHold
  - ◦ PutObjectRetention
  - ◦ SelectObjectContent
- 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 can't be accessed through S3.

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

Operation	Implementation
DeleteObject	Multi-Factor Authentication (MFA) and the response header `x-amz-mfa` aren't supported. When processing a DeleteObject 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 can't 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 bucket with versioning enabled, 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 bucket with versioning suspended, 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. See Use S3 REST API to configure S3 Object Lock to learn how to delete object versions in GOVERNANCE mode.

Operation	Implementation
DeleteObjects (previously named DELETE Multiple Objects)	Multi-Factor Authentication (MFA) and the response header `x-amz-mfa` aren't supported. Multiple objects can be deleted in the same request message. See Use S3 REST API to configure S3 Object Lock to learn how to delete object versions in GOVERNANCE mode.
DeleteObjectTagging	Uses the `tagging` subresource to remove all tags from an object. **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`.
GetObject	GetObject
GetObjectAcl	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.
GetObjectLegalHold	Use S3 REST API to configure S3 Object Lock
GetObjectRetention	Use S3 REST API to configure S3 Object Lock
GetObjectTagging	Uses the `tagging` subresource to return all tags for an object. **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`.
HeadObject	HeadObject
RestoreObject	RestoreObject
PutObject	PutObject

Operation	Implementation
CopyObject (previously named PUT Object - Copy)	CopyObject
PutObjectLegalHold	Use S3 REST API to configure S3 Object Lock
PutObjectRetention	Use S3 REST API to configure S3 Object Lock
PutObjectTagging	Uses the `tagging` subresource to add a set of tags to an existing object. **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 PutObjectTagging 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 can't 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`.
SelectObjectContent	SelectObjectContent

Use S3 Select

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

> ⓘ   Any items not listed aren't 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 Amazon 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.

  > (i) 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:

- PutObject

- CopyObject
- CreateMultipartUpload

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:

- GetObject
- HeadObject
- PutObject
- CopyObject
- CreateMultipartUpload
- UploadPart
- UploadPartCopy

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 cross-grid replication or CloudMirror replication is configured for the bucket, you can't ingest SSE-C objects. The ingest operation will fail.

Related information

[Amazon S3 User Guide: Using server-side encryption with customer-provided keys (SSE-C)](#)

CopyObject

You can use the S3 CopyObject request to create a copy of an object that is already stored in S3. A CopyObject operation is the same as performing GetObject followed by PutObject.

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 PutObject operation is 5 GiB (5,368,709,120 bytes). If you have objects that are larger than 5 GiB, use [multipart upload](#) instead.

The maximum *supported* size for a single PutObject operation is 5 TiB (5,497,558,138,880 bytes).

> ℹ️ If you upgraded from StorageGRID 11.6 or earlier, the S3 PUT Object size too large alert will be triggered if you attempt to upload an object that exceeds 5 GiB. If you have a new installation of StorageGRID 11.7 or 11.8, the alert won't be triggered in this case. However, to align with the AWS S3 standard, future releases of StorageGRID won't support uploads of objects larger than 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 mode and retain-until-date. See Use S3 REST API to configure 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 aren't supported:

- `Cache-Control`

- `Content-Disposition`

- `Content-Encoding`

- `Content-Language`

- `Expires`

- `x-amz-checksum-algorithm`

  When you copy an object, if the source object has a checksum, StorageGRID doesn't copy that checksum value to the new object. This behavior applies whether or not you try to use `x-amz-checksum-algorithm` in the object request.

- `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 uses the Dual commit or Balanced ingest option.

- `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 CopyObject

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 can't use CopyObject 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 can't 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 CopyObject 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.

    > ! 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 for 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 CopyObject request:

    - `x-amz-server-side-encryption`

    > (i) The `server-side-encryption` value of the object can't 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.

GetObject

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

GetObject 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.

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 don't return the `x-amz-missing-meta` header if the key name or value includes unprintable characters.

Supported request header

The following request header is supported:

- `x-amz-checksum-mode`: Specify `ENABLED`

  The `Range` header isn't supported with `x-amz-checksum-mode` for GetObject. When you include `Range` in the request with `x-amz-checksum-mode` enabled, StorageGRID doesn't return a checksum value in the response.

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`.

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.

Behavior of GetObject for Cloud Storage Pool objects

If an object has been stored in a Cloud Storage Pool, the behavior of a GetObject request depends on the state of the object. See HeadObject for more details.

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

State of object	Behavior of GetObject
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, InvalidObjectState` Use a RestoreObject 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 RestoreObject 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 GetObject 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 GetObject request might return some data but stop midway through the transfer.
- A subsequent GetObject request might return `403 Forbidden`.

GetObject and cross-grid replication

If you are using grid federation and cross-grid replication is enabled for a bucket, the S3 client can verify an object's replication status by issuing a GetObject request. The response includes the StorageGRID-specific `x-ntap-sg-cgr-replication-status` response header, which will have one of the following values:

Grid	Replication status
Source	• **COMPLETED**: The replication was successful. • **PENDING**: The object hasn't been replicated yet. • **FAILURE**: The replication failed with a permanent failure. A user must resolve the error.
Destination	**REPLICA**: The object was replicated from the source grid.

> ℹ️ StorageGRID does not support the `x-amz-replication-status` header.

HeadObject

You can use the S3 HeadObject 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 HeadObject to determine the object's transition state.

HeadObject 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.

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 don't return the `x-amz-missing-meta` header if the key name or value includes unprintable characters.

Supported request header

The following request header is supported:

- `x-amz-checksum-mode`

  The `partNumber` parameter and `Range` header aren't supported with `x-amz-checksum-mode` for HeadObject. When you include them in the request with `x-amz-checksum-mode` enabled, StorageGRID doesn't return a checksum value in the response.

Unsupported request header

The following request header isn't 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`.

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.

HeadObject responses for Cloud Storage Pool objects

If the object is stored in a Cloud Storage Pool, 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 HeadObject
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.

State of object	Response to HeadObject

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 RestoreObject 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"`

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 HeadObject 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.

HeadObject and cross-grid replication

If you are using [grid federation](#) and [cross-grid replication](#) is enabled for a bucket, the S3 client can verify an object's replication status by issuing a HeadObject request. The response includes the StorageGRID-specific `x-ntap-sg-cgr-replication-status` response header, which will have one of the following values:

Grid	Replication status
Source	• **COMPLETED**: The replication was successful. • **PENDING**: The object hasn't been replicated yet. • **FAILURE**: The replication failed with a permanent failure. A user must resolve the error.
Destination	**REPLICA**: The object was replicated from the source grid.

> ⓘ StorageGRID does not support the `x-amz-replication-status` header.

PutObject

You can use the S3 PutObject 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 PutObject operation is 5 GiB (5,368,709,120 bytes). If you have objects that are larger than 5 GiB, use [multipart upload](#) instead.

The maximum *supported* size for a single PutObject operation is 5 TiB (5,497,558,138,880 bytes).

> ⓘ If you upgraded from StorageGRID 11.6 or earlier, the S3 PUT Object size too large alert will be triggered if you attempt to upload an object that exceeds 5 GiB. If you have a new installation of StorageGRID 11.7 or 11.8, the alert won't be triggered in this case. However, to align with the AWS S3 standard, future releases of StorageGRID won't support uploads of objects larger than 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:

- PutObject, CopyObject, GetObject, and HeadObject 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-checksum-sha256`
- `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 can't use both a **User defined creation time** for the Reference time and the Balanced or Strict ingest option. 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 mode and retain-until-date. See Use S3 REST API to configure 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 aren't supported:

- `x-amz-acl`
- `x-amz-sdk-checksum-algorithm`
- `x-amz-trailer`
- `x-amz-website-redirect-location`

    The `x-amz-website-redirect-location` header 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 ingest option, 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 don't, 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 can't 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 can't 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.

> ⚠ 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 policies, and does not result in data being stored at lower levels of redundancy in the StorageGRID system.

> ⓘ 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`

  When the `x-amz-server-side-encryption` header isn't included in the PutObject request, the grid-wide stored object encryption setting is omitted from the PutObject response.

- **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.

> ⚠️ 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 for using server-side encryption.

> ℹ️ 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.

Signature calculations for the Authorization header

When using the `Authorization` header to authenticate requests, StorageGRID differs from AWS in the following ways:

- StorageGRID doesn't require `host` headers to be included within `CanonicalHeaders`.

- StorageGRID doesn't require `Content-Type` to be included within `CanonicalHeaders`.

- StorageGRID doesn't require `x-amz-*` headers to be included within `CanonicalHeaders`.

> ℹ️ As a general best practice, always include these headers within `CanonicalHeaders` to ensure they are verified; however, if you exclude these headers, StorageGRID does not return an error.

For details, refer to Signature Calculations for the Authorization Header: Transferring Payload in a Single

[Chunk (AWS Signature Version 4)](#).

Related information

- [Manage objects with ILM](#)

- [Amazon Simple Storage Service API Reference: PutObject](#)

RestoreObject

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

Supported request type

StorageGRID only supports RestoreObject 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 don't specify `versionId`, the most recent version of the object is restored

Behavior of RestoreObject on Cloud Storage Pool objects

If an object has been stored in a [Cloud Storage Pool](#), a RestoreObject request has the following behavior, based on the state of the object. See [HeadObject](#) 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 RestoreObject request. Instead, the local copy can be retrieved directly, using a GetObject request.

State of object	Behavior of RestoreObject
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 can't change its `expiry-date`.

State of object	Behavior of RestoreObject
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 don't specify `Tier`, the `Standard` tier is used. **Important**: If an object has been transitioned to S3 Glacier Deep Archive or the Cloud Storage Pool uses Azure Blob storage, you can't 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`
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 RestoreObject request with a new value for `Days`. The restoration date is updated relative to the time of the request.

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 Amazon Simple Storage Service API Reference: SelectObjectContent.

Before you begin

- 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 must be in one of the following formats:

  ○ **CSV**. Can be used as is or compressed into GZIP or BZIP2 archives.

  ○ **Parquet**. Additional requirements for Parquet objects:

    ▪ S3 Select supports only columnar compression using GZIP or Snappy. S3 Select doesn't support whole-object compression for Parquet objects.

    ▪ S3 Select doesn't support Parquet output. You must specify the output format as CSV or JSON.

    ▪ The maximum uncompressed row group size is 512 MB.

    ▪ You must use the data types specified in the object's schema.

    ▪ You can't use INTERVAL, JSON, LIST, TIME, or UUID logical types.

- Your SQL expression has a maximum length of 256 KB.

- Any record in the input or results has a maximum length of 1 MiB.

CSV 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>
```

Parquet 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>
        <PARQUET>
        </PARQUET>
    </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 aren't 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 (CSV)

```
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,4779736,4854803,1.5705260708959658022953568983726297854
Alaska,710231,738430,3.9703983633493891424057806544631253775
Arizona,6392017,6832810,6.8959922978928247531256565807005832431
Arkansas,2915918,2979732,2.1884703204959810255295244928012378949
California,37253956,38904296,4.4299724839960620557988526104449148971
Colorado,5029196,5454328,8.4532796097030221132761578590295546246
```

AWS-CLI usage example (Parquet)

```
aws s3api select-object-content  -endpoint-url https://10.224.7.44:10443
--bucket 619c0755-9e38-42e0-a614-05064f74126d --key SUB-
EST2020_ALL.parquet --expression "SELECT STNAME, CENSUS2010POP,
POPESTIMATE2015, CAST((POPESTIMATE2015 - CENSUS2010POP) AS DECIMAL) /
CENSUS2010POP * 100.0 FROM S3Object WHERE NAME = STNAME" --expression-type
'SQL' --input-serialization '{"Parquet":{}}'  --output-serialization
'{"CSV": {}}' changes.csv
```

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

```
Alabama,4779736,4854803,1.5705260708959658022953568983726297854
Alaska,710231,738430,3.9703983633493891424057806544631253775
Arizona,6392017,6832810,6.8959922978928247531256565807005832431
Arkansas,2915918,2979732,2.1884703204959810255295244928012378949
California,37253956,38904296,4.4299724839960620557988526104449148971
Colorado,5029196,5454328,8.4532796097030221132761578590295546246
```

Operations for multipart uploads

Operations for multipart uploads

This section describes how StorageGRID supports operations for multipart uploads.

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

- You should not exceed 1,000 concurrent multipart uploads to a single bucket because the results of ListMultipartUploads queries for that bucket might return incomplete results.

- StorageGRID enforces AWS size limits for multipart parts. S3 clients must follow these guidelines:

  - Each part in a multipart upload must be between 5 MiB (5,242,880 bytes) and 5 GiB (5,368,709,120 bytes).

  - The last part can be smaller than 5 MiB (5,242,880 bytes).

  - In general, part sizes should be as large as possible. For example, use part sizes of 5 GiB for a 100 GiB object. Because each part is considered a unique object, using large part sizes reduces StorageGRID metadata overhead.

  - For objects smaller than 5 GiB, consider using non-multipart upload instead.

- ILM is evaluated for each part of a multipart object as it is ingested and for the object as a whole when the multipart upload completes, if the ILM rule uses the Balanced or Strict ingest option. You should be aware of how this affects object and part placement:

  - If ILM changes while an S3 multipart upload is in progress, some parts of the object might not meet current ILM requirements when the multipart upload completes. Any part that is not placed correctly is queued for ILM re-evaluation and moved to the correct location later.

  - When evaluating ILM for a part, StorageGRID filters on the size of the part, not the size of the object.

This means that parts of an object can be stored in locations that don't meet ILM requirements for the object as a whole. For example, if a rule specifies that all objects 10 GB or larger are stored at DC1 while all smaller objects are stored at DC2, each 1 GB part of a 10-part multipart upload is stored at DC2 at ingest. However, when ILM is evaluated for the object as a whole, all parts of the object are moved to DC1.

- All of the multipart upload operations support StorageGRID consistency values.

- When an object is ingested using multipart upload, the object segmentation threshold (1 GiB) is not applied.

- As required, you can use server-side encryption with multipart uploads. To use SSE (server-side encryption with StorageGRID-managed keys), you include the `x-amz-server-side-encryption` request header in the CreateMultipartUpload request only. To use SSE-C (server-side encryption with customer-provided keys), you specify the same three encryption key request headers in the CreateMultipartUpload request and in each subsequent UploadPart request.

Operation	Implementation
AbortMultipartUpload	Implemented with all Amazon S3 REST API behavior. Subject to change without notice.
CompleteMultipartUpload	See CompleteMultipartUpload
CreateMultipartUpload (previously named Initiate Multipart Upload)	See CreateMultipartUpload
ListMultipartUploads	See ListMultipartUploads
ListParts	Implemented with all Amazon S3 REST API behavior. Subject to change without notice.
UploadPart	See UploadPart
UploadPartCopy	See UploadPartCopy

CompleteMultipartUpload

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

> (i) StorageGRID supports non-consecutive values in ascending order for the `partNumber` request parameter with CompleteMultipartUpload. The parameter can start with any value.

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.

Supported request headers

The following request headers are supported:

- `x-amz-checksum-sha256`
- `x-amz-storage-class`

  The `x-amz-storage-class` header affects how many object copies StorageGRID creates if the matching ILM rule specifies the Dual commit or Balanced ingest option.

- `STANDARD`

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

- `REDUCED_REDUNDANCY`

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

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

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

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

Unsupported request headers

The following request headers aren't supported:

- `x-amz-sdk-checksum-algorithm`
- `x-amz-trailer`

Versioning

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

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

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

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

Failed replication, notification, or metadata notification

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

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

Refer to Troubleshoot platform services.

CreateMultipartUpload

The CreateMultipartUpload (previously named Initiate Multipart Upload) operation initiates a multipart upload for an object, and returns an upload ID.

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

If the ILM rule matching an ingested object uses the Strict ingest option, 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 ingest option, 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 don't, 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 can't 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, 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 can't 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.

> ⚠️ 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 policies, and does not result in data being stored at lower levels of redundancy in the StorageGRID system.

> ℹ️ 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.

Supported request headers

The following request headers are supported:

- `Content-Type`

- `x-amz-checksum-algorithm`

  Currently, only the SHA256 value for `x-amz-checksum-algorithm` is supported.

- `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.

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

- S3 Object Lock request headers:

- ◦ `x-amz-object-lock-mode`

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

- ◦ `x-amz-object-lock-legal-hold`

   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 REST API to configure 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`

    Request headers for server-side encryption

    > ⓘ For information about how StorageGRID handles UTF-8 characters, see PutObject.

Request headers for server-side encryption

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

- **SSE**: Use the following header in the CreateMultipartUpload request if you want to encrypt the object with a unique key managed by StorageGRID. Don't specify this header in any of the UploadPart requests.

  - ◦ `x-amz-server-side-encryption`

- **SSE-C**: Use all three of these headers in the CreateMultipartUpload request (and in each subsequent UploadPart request) if you want to encrypt the object with a unique key that you provide and manage.

  - ◦ `x-amz-server-side-encryption-customer-algorithm`: Specify `AES256`.

  - ◦ `x-amz-server-side-encryption-customer-key`: Specify your encryption key for the new object.

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

    > ⚠ 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 for using server-side encryption.

Unsupported request headers

The following request header isn't supported:

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

  The `x-amz-website-redirect-location` header returns `XNotImplemented`.

Versioning

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

ListMultipartUploads

The ListMultipartUploads operation lists in-progress multipart uploads for a bucket.

The following request parameters are supported:

* `encoding-type`
* `key-marker`
* `max-uploads`
* `prefix`
* `upload-id-marker`
* `Host`
* `Date`
* `Authorization`

Versioning

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

UploadPart

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

Supported request headers

The following request headers are supported:

* `x-amz-checksum-sha256`
* `Content-Length`
* `Content-MD5`

Request headers for server-side encryption

If you specified SSE-C encryption for the CreateMultipartUpload request, you must also include the following request headers in each UploadPart request:

* `x-amz-server-side-encryption-customer-algorithm`: Specify `AES256`.
* `x-amz-server-side-encryption-customer-key`: Specify the same encryption key that you provided in the CreateMultipartUpload request.

- `x-amz-server-side-encryption-customer-key-MD5`: Specify the same MD5 digest that you provided in the CreateMultipartUpload request.

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

If you specified a SHA-256 checksum during the CreateMultipartUpload request, you must also include the following request header in each UploadPart request:

- `x-amz-checksum-sha256`: Specify the SHA-256 checksum for this part.

Unsupported request headers

The following request headers aren't supported:

- `x-amz-sdk-checksum-algorithm`
- `x-amz-trailer`

Versioning

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

UploadPartCopy

The UploadPartCopy operation uploads a part of an object by copying data from an existing object as the data source.

The UploadPartCopy operation is implemented with all Amazon S3 REST API behavior. Subject to change without notice.

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

The following request headers are supported:

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

Request headers for server-side encryption

If you specified SSE-C encryption for the CreateMultipartUpload request, you must also include the following request headers in each UploadPartCopy request:

- `x-amz-server-side-encryption-customer-algorithm`: Specify `AES256`.
- `x-amz-server-side-encryption-customer-key`: Specify the same encryption key that you

provided in the CreateMultipartUpload request.

- `x-amz-server-side-encryption-customer-key-MD5`: Specify the same MD5 digest that you provided in the CreateMultipartUpload request.

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

- `x-amz-copy-source-server-side-encryption-customer-algorithm`: Specify `AES256`.

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

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

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

Versioning

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

Error responses

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

Supported S3 API error codes

Name	HTTP status
AccessDenied	403 Forbidden
BadDigest	400 Bad Request
BucketAlreadyExists	409 Conflict
BucketNotEmpty	409 Conflict
IncompleteBody	400 Bad Request
InternalError	500 Internal Server Error
InvalidAccessKeyId	403 Forbidden
InvalidArgument	400 Bad Request

Name	HTTP status
InvalidBucketName	400 Bad Request
InvalidBucketState	409 Conflict
InvalidDigest	400 Bad Request
InvalidEncryptionAlgorithmError	400 Bad Request
InvalidPart	400 Bad Request
InvalidPartOrder	400 Bad Request
InvalidRange	416 Requested Range Not Satisfiable
InvalidRequest	400 Bad Request
InvalidStorageClass	400 Bad Request
InvalidTag	400 Bad Request
InvalidURI	400 Bad Request
KeyTooLong	400 Bad Request
MalformedXML	400 Bad Request
MetadataTooLarge	400 Bad Request
MethodNotAllowed	405 Method Not Allowed
MissingContentLength	411 Length Required
MissingRequestBodyError	400 Bad Request
MissingSecurityHeader	400 Bad Request
NoSuchBucket	404 Not Found
NoSuchKey	404 Not Found
NoSuchUpload	404 Not Found
NotImplemented	501 Not Implemented

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

StorageGRID custom error codes

Name	Description	HTTP status
XBucketLifecycleNotAllowed	Bucket lifecycle configuration is not allowed in a legacy Compliant bucket	400 Bad Request
XBucketPolicyParseException	Failed to parse received bucket policy JSON.	400 Bad Request
XComplianceConflict	Operation denied because of legacy Compliance settings.	403 Forbidden
XComplianceReducedRedundancyForbidden	Reduced redundancy is not allowed in legacy Compliant bucket	400 Bad Request
XMaxBucketPolicyLengthExceeded	Your policy exceeds the maximum allowed bucket policy length.	400 Bad Request
XMissingInternalRequestHeader	Missing a header of an internal request.	400 Bad Request
XNoSuchBucketCompliance	The specified bucket does not have legacy Compliance enabled.	404 Not Found
XNotAcceptable	The request contains one or more accept headers that could not be satisfied.	406 Not Acceptable

Name	Description	HTTP status
XNotImplemented	The request you provided implies functionality that is not implemented.	501 Not Implemented