Operations for multipart uploads

StorageGRID 11.8

NetApp
May 17, 2024

This PDF was generated from https://docs.netapp.com/us-en/storagegrid-118/s3/operations-for-multipart-uploads.html on May 17, 2024. Always check docs.netapp.com for the latest.

Table of Contents

Operations for multipart uploads

Operations for multipart uploads: Overview

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

Operation	Implementation
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.

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.

Request headers

The `x-amz-storage-class` request header is supported, and affects how many object copies StorageGRID creates if the matching ILM rule specifies 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.

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.

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

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

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.

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

The following request headers are supported:

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

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

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

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

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

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

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

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

  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 is not supported and returns `XNotImplemented`

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

Versioning

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

- `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.

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.

Copyright information

Copyright © 2024 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

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

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

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

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

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

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

Trademark information

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