Use S3 REST API

StorageGRID 11.7

NetApp
April 10, 2024

Table of Contents

Use S3 REST API

S3 REST API supported versions and updates

StorageGRID supports the Simple Storage Service (S3) API, which is implemented as a set of Representational State Transfer (REST) web services.

Support for the S3 REST API enables you to connect service-oriented applications developed for S3 web services with on-premise object storage that uses the StorageGRID system. Minimal changes to a client application's current use of S3 REST API calls are required.

Supported versions

StorageGRID supports the following specific versions of S3 and HTTP.

Item	Version
S3 specification	*Simple Storage Service API Reference* 2006-03-01
HTTP	1.1 For more information about HTTP, see HTTP/1.1 (RFCs 7230-35). **Note**: StorageGRID does not support HTTP/1.1 pipelining.

Related information

IETF RFC 2616: Hypertext Transfer Protocol (HTTP/1.1)

Amazon Web Services (AWS) Documentation: Amazon Simple Storage Service API Reference

Updates to S3 REST API support

Release	Comments
11.7	• Added Quick reference: Supported S3 API requests. • Added support for using GOVERNANCE mode with S3 Object Lock. • Added support for the StorageGRID-specific `x-ntap-sg-cgr-replication-status` response header for GET Object and HEAD Object requests. This header provides an object's replication status for cross-grid replication. • SelectObjectContent requests now support Parquet objects.

Release	Comments
11.6	• Added support for using the `partNumber` request parameter in GET Object and HEAD Object requests. • Added support for a default retention mode and a default retention period at the bucket level for S3 Object Lock. • Added support for the `s3:object-lock-remaining-retention-days` policy condition key to set the range of allowable retention periods for your objects. • Changed the maximum *recommended* size for a single PUT Object operation to 5 GiB (5,368,709,120 bytes). If you have objects that are larger than 5 GiB, use multipart upload instead.
11.5	• Added support for managing bucket encryption. • Added support for S3 Object Lock and deprecated legacy Compliance requests. • Added support for using DELETE Multiple Objects on versioned buckets. • The `Content-MD5` request header is now correctly supported.
11.4	• Added support for DELETE Bucket tagging, GET Bucket tagging, and PUT Bucket tagging. Cost allocation tags aren't supported. • For buckets created in StorageGRID 11.4, restricting object key names to meet performance best practices is no longer required. • Added support for bucket notifications on the `s3:ObjectRestore:Post` event type. • AWS size limits for multipart parts are now enforced. Each part in a multipart upload must be between 5 MiB and 5 GiB. The last part can be smaller than 5 MiB. • Added support for TLS 1.3
11.3	• Added support for server-side encryption of object data with customer-provided keys (SSE-C). • Added support for DELETE, GET, and PUT Bucket lifecycle operations (Expiration action only) and for the `x-amz-expiration` response header. • Updated PUT Object, PUT Object - Copy, and Multipart Upload to describe the impact of ILM rules that use synchronous placement at ingest. • TLS 1.1 ciphers are no longer supported.
11.2	Added support for POST Object restore for use with Cloud Storage Pools. Added support for using the AWS syntax for ARN, policy condition keys, and policy variables in group and bucket policies. Existing group and bucket policies that use the StorageGRID syntax will continue to be supported. **Note:** Uses of ARN/URN in other configuration JSON/XML, including those used in custom StorageGRID features, have not changed.
11.1	Added support for cross-origin resource sharing (CORS), HTTP for S3 client connections to grid nodes, and compliance settings on buckets.

Release	Comments
11.0	Added support for configuring platform services (CloudMirror replication, notifications, and Elasticsearch search integration) for buckets. Also added support for object tagging location constraints for buckets, and the Available consistency control setting.
10.4	Added support for ILM scanning changes to versioning, Endpoint Domain Names page updates, conditions and variables in policies, policy examples, and the PutOverwriteObject permission.
10.3	Added support for versioning.
10.2	Added support for group and bucket access policies, and for multipart copy (Upload Part - Copy).
10.1	Added support for multipart upload, virtual hosted-style requests, and v4 authentication.
10.0	Initial support of the S3 REST API by the StorageGRID system.The currently supported version of the *Simple Storage Service API Reference* is 2006-03-01.

Quick reference: Supported S3 API requests

This page summarizes how StorageGRID supports Amazon Simple Storage Service (S3) APIs.

This page includes only the S3 operations that are supported by StorageGRID.

ⓘ To see the AWS documentation for each operation, select the link in the heading.

Common URI query parameters and request headers

Unless noted, the following common URI query parameters are supported:

- `versionId` (as required for object operations)

Unless noted, the following common request headers are supported:

- `Authorization`
- `Connection`
- `Content-Length`
- `Content-MD5`
- `Content-Type`
- `Date`
- `Expect`
- `Host`

- x-amz-date

Related information

- S3 REST API implementation details
- Amazon Simple Storage Service API Reference: Common Request Headers

AbortMultipartUpload

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus this additional URI query parameter:

- uploadId

Request body

None

StorageGRID documentation

Operations for multipart uploads

CompleteMultipartUpload

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus this additional URI query parameter:

- uploadId

Request body XML tags

StorageGRID supports these request body XML tags:

- CompleteMultipartUpload
- Part
- ETag
- PartNumber

StorageGRID documentation

Complete Multipart Upload

CopyObject

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus these additional headers:

- x-amz-copy-source
- x-amz-copy-source-if-match
- x-amz-copy-source-if-modified-since

- `x-amz-copy-source-if-none-match`
- `x-amz-copy-source-if-unmodified-since`
- `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-metadata-directive`
- `x-amz-object-lock-legal-hold`
- `x-amz-object-lock-mode`
- `x-amz-object-lock-retain-until-date`
- `x-amz-server-side-encryption`
- `x-amz-server-side-encryption-customer-algorithm`
- `x-amz-server-side-encryption-customer-key`
- `x-amz-server-side-encryption-customer-key-MD5`
- `x-amz-storage-class`
- `x-amz-tagging`
- `x-amz-tagging-directive`
- `x-amz-meta-<metadata-name>`

Request body

None

StorageGRID documentation

PUT Object-Copy

CreateBucket

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus these additional headers:

- `x-amz-bucket-object-lock-enabled`

Request body

StorageGRID supports all request body parameters defined by the Amazon S3 REST API at the time of implementation.

StorageGRID documentation

Operations on buckets

CreateMultipartUpload

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus these additional headers:

- `Cache-Control`
- `Content-Disposition`
- `Content-Encoding`
- `Content-Language`
- `Expires`
- `x-amz-server-side-encryption`
- `x-amz-storage-class`
- `x-amz-server-side-encryption-customer-algorithm`
- `x-amz-server-side-encryption-customer-key`
- `x-amz-server-side-encryption-customer-key-MD5`
- `x-amz-tagging`
- `x-amz-object-lock-mode`
- `x-amz-object-lock-retain-until-date`
- `x-amz-object-lock-legal-hold`
- `x-amz-meta-<metadata-name>`

Request body

None

StorageGRID documentation

Initiate Multipart Upload

DeleteBucket

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

StorageGRID documentation

Operations on buckets

DeleteBucketCors

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on buckets

DeleteBucketEncryption

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on buckets

DeleteBucketLifecycle

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

- Operations on buckets
- Create S3 lifecycle configuration

DeleteBucketPolicy

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on buckets

DeleteBucketReplication

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on buckets

DeleteBucketTagging

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on buckets

DeleteObject

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus this additional request header:

- `x-amz-bypass-governance-retention`

Request body

None

StorageGRID documentation

Operations on objects

DeleteObjects

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus this additional request header:

- `x-amz-bypass-governance-retention`

Request body

StorageGRID supports all request body parameters defined by the Amazon S3 REST API at the time of implementation.

StorageGRID documentation

Operations on objects (DELETE Multiple Objects)

DeleteObjectTagging

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on objects

GetBucketAcl

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on buckets

GetBucketCors

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on buckets

GetBucketEncryption

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on buckets

GetBucketLifecycleConfiguration

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

- Operations on buckets (GET Bucket lifecycle)
- Create S3 lifecycle configuration

GetBucketLocation

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on buckets

GetBucketNotificationConfiguration

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on buckets (GET Bucket notification)

GetBucketPolicy

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on buckets

GetBucketReplication

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on buckets

GetBucketTagging

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on buckets

GetBucketVersioning

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on buckets

GetObject

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus these additional URI query parameters:

- `partNumber`
- `response-cache-control`
- `response-content-disposition`
- `response-content-encoding`
- `response-content-language`
- `response-content-type`
- `response-expires`

And these additional request headers:

- `Range`
- `x-amz-server-side-encryption-customer-algorithm`
- `x-amz-server-side-encryption-customer-key`
- `x-amz-server-side-encryption-customer-key-MD5`
- `If-Match`
- `If-Modified-Since`
- `If-None-Match`
- `If-Unmodified-Since`

Request body

None

StorageGRID documentation

GET Object

GetObjectAcl

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on objects

GetObjectLegalHold

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Use S3 REST API to configure S3 Object Lock

GetObjectLockConfiguration

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Use S3 REST API to configure S3 Object Lock

GetObjectRetention

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Use S3 REST API to configure S3 Object Lock

GetObjectTagging

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on objects

HeadBucket

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on buckets

HeadObject

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus these additional headers:

- `x-amz-server-side-encryption-customer-algorithm`
- `x-amz-server-side-encryption-customer-key`
- `x-amz-server-side-encryption-customer-key-MD5`
- `If-Match`
- `If-Modified-Since`
- `If-None-Match`
- `If-Unmodified-Since`
- `Range`

Request body

None

StorageGRID documentation

HEAD Object

ListBuckets

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

None

StorageGRID documentation

Operations on the service > GET Service

ListMultipartUploads

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus these additional parameters:

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

Request body

None

StorageGRID documentation

List Multipart Uploads

ListObjects

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus these additional parameters:

- `delimiter`
- `encoding-type`
- `marker`
- `max-keys`
- `prefix`

Request body

None

StorageGRID documentation

Operations on buckets (GET Bucket)

ListObjectsV2

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus these additional parameters:

- `continuation-token`
- `delimiter`
- `encoding-type`
- `fetch-owner`

- `max-keys`
- `prefix`
- `start-after`

Request body

None

StorageGRID documentation

Operations on buckets (GET Bucket)

ListObjectVersions

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus these additional parameters:

- `delimiter`
- `encoding-type`
- `key-marker`
- `max-keys`
- `prefix`
- `version-id-marker`

Request body

None

StorageGRID documentation

Operations on buckets (GET Bucket Object versions)

ListParts

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus these additional parameters:

- `max-parts`
- `part-number-marker`
- `uploadId`

Request body

None

StorageGRID documentation

List Multipart Uploads

PutBucketCors

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

StorageGRID supports all request body parameters defined by the Amazon S3 REST API at the time of implementation.

StorageGRID documentation

Operations on buckets

PutBucketEncryption

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body XML tags

StorageGRID supports these request body XML tags:

- `ServerSideEncryptionConfiguration`
- `Rule`
- `ApplyServerSideEncryptionByDefault`
- `SSEAlgorithm`

StorageGRID documentation

Operations on buckets

PutBucketLifecycleConfiguration

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body XML tags

StorageGRID supports these request body XML tags:

- `NewerNoncurrentVersions`
- `LifecycleConfiguration`
- `Rule`
- `Expiration`
- `Days`
- `Filter`
- `And`
- `Prefix`
- `Tag`

- `Key`
- `Value`
- `Prefix`
- `Tag`
- `Key`
- `Value`
- `ID`
- `NoncurrentVersionExpiration`
- `NoncurrentDays`
- `Prefix`
- `Status`

StorageGRID documentation

- Operations on buckets (PUT Bucket lifecycle)
- Create S3 lifecycle configuration

PutBucketNotificationConfiguration

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body XML tags

StorageGRID supports these request body XML tags:

- `Prefix`
- `Suffix`
- `NotificationConfiguration`
- `TopicConfiguration`
- `Event`
- `Filter`
- `S3Key`
- `FilterRule`
- `Name`
- `Value`
- `Id`
- `Topic`

StorageGRID documentation

Operations on buckets (PUT Bucket notification)

PutBucketPolicy

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

For details about the supported JSON body fields, see Use bucket and group access policies.

PutBucketReplication

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body XML tags

- `ReplicationConfiguration`
- `Status`
- `Prefix`
- `Destination`
- `Bucket`
- `StorageClass`
- `Rule`

StorageGRID documentation

Operations on buckets

PutBucketTagging

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

StorageGRID supports all request body parameters defined by the Amazon S3 REST API at the time of implementation.

StorageGRID documentation

Operations on buckets

PutBucketVersioning

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body parameters

StorageGRID supports these request body parameters:

- `VersioningConfiguration`
- `Status`

StorageGRID documentation

Operations on buckets

PutObject

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus these additional headers:

- `Cache-Control`
- `Content-Disposition`
- `Content-Encoding`
- `Content-Language`
- `x-amz-server-side-encryption`
- `x-amz-storage-class`
- `x-amz-server-side-encryption-customer-algorithm`
- `x-amz-server-side-encryption-customer-key`
- `x-amz-server-side-encryption-customer-key-MD5`
- `x-amz-tagging`
- `x-amz-object-lock-mode`
- `x-amz-object-lock-retain-until-date`
- `x-amz-object-lock-legal-hold`
- `x-amz-meta-<metadata-name>`

Request body

- Binary data of the object

StorageGRID documentation

PUT Object

PutObjectLegalHold

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

StorageGRID supports all request body parameters defined by the Amazon S3 REST API at the time of implementation.

StorageGRID documentation

Use S3 REST API to configure S3 Object Lock

PutObjectLockConfiguration

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

StorageGRID supports all request body parameters defined by the Amazon S3 REST API at the time of implementation.

StorageGRID documentation

Use S3 REST API to configure S3 Object Lock

PutObjectRetention

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus this additional header:

- `x-amz-bypass-governance-retention`

Request body

StorageGRID supports all request body parameters defined by the Amazon S3 REST API at the time of implementation.

StorageGRID documentation

Use S3 REST API to configure S3 Object Lock

PutObjectTagging

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

StorageGRID supports all request body parameters defined by the Amazon S3 REST API at the time of implementation.

StorageGRID documentation

Operations on objects

SelectObjectContent

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request.

Request body

For details about the supported body fields, see the following:

- Use S3 Select
- Select Object Content

UploadPart

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus these additional URI query parameters:

- `partNumber`
- `uploadId`

And these additional request headers:

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

Request body

- Binary data of the part

StorageGRID documentation

Upload Part

UploadPartCopy

URI query parameters and request headers

StorageGRID supports all common parameters and headers for this request, plus these additional URI query parameters:

- `partNumber`
- `uploadId`

And these additional request headers:

- `x-amz-copy-source`
- `x-amz-copy-source-if-match`
- `x-amz-copy-source-if-modified-since`
- `x-amz-copy-source-if-none-match`
- `x-amz-copy-source-if-unmodified-since`
- `x-amz-copy-source-range`
- `x-amz-server-side-encryption-customer-algorithm`
- `x-amz-server-side-encryption-customer-key`
- `x-amz-server-side-encryption-customer-key-MD5`
- `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`

Request body

None

StorageGRID documentation

Upload Part - Copy

Configure tenant accounts and connections

Configuring StorageGRID to accept connections from client applications requires creating one or more tenant accounts and setting up the connections.

Create and configure S3 tenant accounts

An S3 tenant account is required before S3 API clients can store and retrieve objects on StorageGRID. Each tenant account has its own account ID, groups, users, buckets, and objects.

S3 tenant accounts are created by a StorageGRID grid administrator using the Grid Manager or the Grid Management API. See Manage tenants for details. After an S3 tenant account is created, tenant users can access the Tenant Manager to manage groups, users, access keys, and buckets. See Use a tenant account for details.

> While S3 tenant users can create and manage S3 access key and buckets with the Tenant Manager, they must use an S3 client application to ingest and manage objects. See Use S3 REST API for details.

How to configure client connections

A grid administrator makes configuration choices that affect how S3 clients connect to StorageGRID to store and retrieve data. There are four basic steps for connecting StorageGRID to any S3 application:

- Perform prerequisite tasks in StorageGRID, based on how the client application will connect to StorageGRID.
- Use StorageGRID to obtain the values the application needs to connect to the grid. You can either use the S3 setup wizard or configure each StorageGRID entity manually.
- Use the S3 application to complete the connection to StorageGRID. Create DNS entries to associate IP addresses to any domain names you plan to use.
- Perform ongoing tasks in the application and in StorageGRID to manage and monitor object storage over time.

For details on these steps, see Configure client connections.

Information required for client connections

To store or retrieve objects, S3 client applications connect to the Load Balancer service, which is included on all Admin Nodes and Gateway Nodes, or to the Local Distribution Router (LDR) service, which is included on all Storage Nodes.

Client applications can connect to StorageGRID using the IP address of a grid node and the port number of the service on that node. Optionally, you can create high availability (HA) groups of load-balancing nodes to provide highly available connections that use virtual IP (VIP) addresses. If you want to connect to StorageGRID using a fully qualified domain name (FQDN) instead of an IP or VIP address, you can configure

DNS entries.

See Summary: IP addresses and ports for client connections for more information.

Decide to use HTTPS or HTTP connections

When client connections are made using a Load Balancer endpoint, connections must be made using the protocol (HTTP or HTTPS) that was specified for that endpoint. To use HTTP for client connections to Storage Nodes, you must enable its use.

By default, when client applications connect to Storage Nodes, they must use encrypted HTTPS for all connections. Optionally, you can enable less-secure HTTP connections by selecting **CONFIGURATION** > **Security settings** > **Network and objects** > **Enable HTTP for Storage Node connections** in the Grid Manager. For example, a client application might use HTTP when testing the connection to a Storage Node in a non-production environment.

> ! Be careful when enabling HTTP for a production grid because requests and responses will be sent unencrypted.

Related information

Administer StorageGRID

Benefits of active, idle, and concurrent HTTP connections

S3 endpoint domain names for S3 requests

Before you can use S3 endpoint domain names for client requests, a StorageGRID administrator must configure the system to accept connections that use S3 endpoint domain names in S3 path-style and S3 virtual-hosted-style requests.

About this task

To enable you to use S3 virtual hosted style-requests, a grid administrator must perform the following tasks:

- Use the Grid Manager to add the S3 endpoint domain names to the StorageGRID system.
- Ensure that the certificate the client uses for HTTPS connections to StorageGRID is signed for all domain names that the client requires.

  For example, if the S3 API service endpoint domain endpoint is `s3.company.com`, the grid administrator must ensure that the certificate used for HTTPS connections has `s3.company.com` as the Subject Common Name and in the Subject Alternative Names, and `*.s3.company.com` in the Subject Alternative Names.
- Configure the DNS server used by the client to include DNS records that match the S3 endpoint domain names, including any required wildcard records.

If the client connects using the Load Balancer service, the certificate that the grid administrator configures is the certificate for the load balancer endpoint that the client uses.

> i Each load balancer endpoint has its own certificate, and each endpoint can be configured to recognize different S3 endpoint domain names.

If the client connects to Storage Nodes, the certificate that the grid administrator configures is the single custom server certificate used for the grid.

See the instructions for administering StorageGRID for more information.

After these steps have been completed, you can use virtual hosted-style requests.

Test your S3 REST API configuration

You can use the Amazon Web Services Command Line Interface (AWS CLI) to test your connection to the system and to verify that you can read and write objects to the system.

Before you begin

- You have downloaded and installed the AWS CLI from aws.amazon.com/cli.
- You have created an S3 tenant account in the StorageGRID system.
- You have created an access key in the tenant account.

Steps

1. Configure the AWS CLI settings to use the account you created in the StorageGRID system:
   a. Enter configuration mode: `aws configure`
   b. Enter the access key ID for the account you created.
   c. Enter the secret access key for the account you created.
   d. Enter the default region to use, for example, us-east-1.
   e. Enter the default output format to use, or press **Enter** to select JSON.
2. Create a bucket.

   This example assumes you configured a load balancer endpoint to use IP address 10.96.101.17 and port 10443.

   ```
   aws s3api --endpoint-url https://10.96.101.17:10443
   --no-verify-ssl create-bucket --bucket testbucket
   ```

   If the bucket is created successfully, the location of the bucket is returned, as seen in the following example:

   ```
   "Location": "/testbucket"
   ```

3. Upload an object.

   ```
   aws s3api --endpoint-url https://10.96.101.17:10443 --no-verify-ssl
   put-object --bucket testbucket --key s3.pdf --body C:\s3-
   test\upload\s3.pdf
   ```

   If the object is uploaded successfully, an Etag is returned which is a hash of the object data.

4. List the contents of the bucket to verify that the object was uploaded.

```
aws s3api --endpoint-url https://10.96.101.17:10443 --no-verify-ssl
list-objects --bucket testbucket
```

5. Delete the object.

```
aws s3api --endpoint-url https://10.96.101.17:10443 --no-verify-ssl
delete-object --bucket testbucket --key s3.pdf
```

6. Delete the bucket.

```
aws s3api --endpoint-url https://10.96.101.17:10443 --no-verify-ssl
delete-bucket --bucket testbucket
```

Support for StorageGRID platform services

StorageGRID platform services enable StorageGRID tenant accounts to leverage external services such as a remote S3 bucket, a Simple Notification Service (SNS) endpoint, or an Elasticsearch cluster to extend the services provided by a grid.

The following table summarizes the available platform services and the S3 APIs used to configure them.

Platform service	**Purpose**	**S3 API used to configure the service**
CloudMirror replication	Replicates objects from a source StorageGRID bucket to the configured remote S3 bucket.	PUT Bucket replication (see Operations on buckets)
Notifications	Sends notifications about events in a source StorageGRID bucket to a configured Simple Notification Service (SNS) endpoint.	PUT Bucket notification (see Operations on buckets)
Search integration	Sends object metadata for objects stored in a StorageGRID bucket to a configured Elasticsearch index.	PUT Bucket metadata notification configuration **Note:** This is a StorageGRID custom S3 API.

A grid administrator must enable the use of platform services for a tenant account before they can be used. See Administer StorageGRID. Then, a tenant administrator must create an endpoint that represents the remote service in the tenant account. This step is required before a service can be configured. See Use a tenant account.

Recommendations for using platform services

Before using platform services, you must be aware of the following recommendations:

- NetApp recommends that you allow no more than 100 active tenants with S3 requests requiring CloudMirror replication, notifications, and search integration. Having more than 100 active tenants can result in slower S3 client performance.
- If an S3 bucket in the StorageGRID system has both versioning and CloudMirror replication enabled, NetApp recommends that the destination endpoint also have S3 bucket versioning enabled. This allows CloudMirror replication to generate similar object versions on the endpoint.
- CloudMirror replication is not supported if the source bucket has S3 Object Lock enabled.
- CloudMirror replication will fail with an AccessDenied error if the destination bucket has legacy Compliance enabled.

How StorageGRID implements S3 REST API

Conflicting client requests

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.

Consistency controls

Consistency controls provide a balance between the availability of the objects and the consistency of those objects across different Storage Nodes and sites, as required by your application.

By default, StorageGRID guarantees read-after-write consistency for newly created objects. Any GET following a successfully completed PUT will be able to read the newly written data. Overwrites of existing objects, metadata updates, and deletes are eventually consistent. Overwrites generally take seconds or minutes to propagate, but can take up to 15 days.

If you want to perform object operations at a different consistency level, you can specify a consistency control for each bucket or for each API operation.

Consistency controls

The consistency control affects how the metadata that StorageGRID uses to track objects is distributed between nodes, and therefore the availability of objects for client requests.

You can set the consistency control for a bucket or an API operation to one of the following values:

- **all**: All nodes receive the data immediately, or the request will fail.
- **strong-global**: Guarantees read-after-write consistency for all client requests across all sites.
- **strong-site**: Guarantees read-after-write consistency for all client requests within a site.
- **read-after-new-write**: (Default) Provides read-after-write consistency for new objects and eventual consistency for object updates. Offers high availability and data protection guarantees. Recommended for

most cases.

- **available**: Provides eventual consistency for both new objects and object updates. For S3 buckets, use only as required (for example, for a bucket that contains log values that are rarely read, or for HEAD or GET operations on keys that don’t exist). Not supported for S3 FabricPool buckets.

Use “read-after-new-write” and “available” consistency controls

When a HEAD or GET operation uses the “read-after-new-write” consistency control, StorageGRID performs the lookup in multiple steps, as follows:

- It first looks up the object using a low consistency.
- If that lookup fails, it repeats the lookup at the next consistency level until it reaches a consistency level equivalent to the behavior for strong-global.

If a HEAD or GET operation uses the “read-after-new-write” consistency control but the object does not exist, the object lookup will always reach a consistency level equivalent to the behavior for strong-global. Because this consistency level requires multiple copies of the object metadata to be available at each site, you can receive a high number of 500 Internal Server errors if two or more Storage Nodes at the same site are unavailable.

Unless you require consistency guarantees similar to Amazon S3, you can prevent these errors for HEAD and GET operations by setting the consistency control to “available.” When a HEAD or GET operation uses the “available” consistency control, StorageGRID provides eventual consistency only. It does not retry a failed operation at increasing consistency levels, so it does not require that multiple copies of the object metadata be available.

Specify consistency control for API operation

To set the consistency control for an individual API operation, consistency controls must be supported for the operation, and you must specify the consistency control in the request header. This example sets the consistency control to “strong-site” for a GET Object operation.

```
GET /bucket/object HTTP/1.1
Date: date
Authorization: authorization name
Host: host
Consistency-Control: strong-site
```

ⓘ You must use the same consistency control for both the PUT Object and GET Object operations.

Specify consistency control for bucket

To set the consistency control for bucket, you can use the StorageGRID PUT Bucket consistency request and the GET Bucket consistency request. Or you can use the Tenant Manager or the Tenant Management API.

When setting the consistency controls for a bucket, be aware of the following:

- Setting the consistency control for a bucket determines which consistency control is used for S3 operations performed on the objects in the bucket or on the bucket configuration. It does not affect operations on the bucket itself.

- The consistency control for an individual API operation overrides the consistency control for the bucket.
- In general, buckets should use the default consistency control, “read-after-new-write.” If requests aren’t working correctly, change the application client behavior if possible. Or, configure the client to specify the consistency control for each API request. Set the consistency control at the bucket level only as a last resort.

How consistency controls and ILM rules interact to affect data protection

Both your choice of consistency control and your ILM rule affect how objects are protected. These settings can interact.

For example, the consistency control used when an object is stored affects the initial placement of object metadata, while the ingest behavior selected for the ILM rule affects the initial placement of object copies. Because StorageGRID requires access to both an object’s metadata and its data to fulfill client requests, selecting matching levels of protection for the consistency level and ingest behavior can provide better initial data protection and more predictable system responses.

The following ingest behaviors are available for ILM rules:

- **Dual commit**: StorageGRID immediately makes interim copies of the object and returns success to the client. Copies specified in the ILM rule are made when possible.
- **Strict**: All copies specified in the ILM rule must be made before success is returned to the client.
- **Balanced**: StorageGRID attempts to make all copies specified in the ILM rule at ingest; if this is not possible, interim copies are made and success is returned to the client. The copies specified in the ILM rule are made when possible.

> Before selecting the ingest behavior for an ILM rule, read the full description of these settings in the instructions for managing objects with information lifecycle management.

Example of how the consistency control and ILM rule can interact

Suppose you have a two-site grid with the following ILM rule and the following consistency level setting:

- **ILM rule**: Create two object copies, one at the local site and one at a remote site. The Strict ingest behavior is selected.
- **Consistency level**: “strong-global” (Object metadata is immediately distributed to all sites.)

When a client stores an object to the grid, StorageGRID makes both object copies and distributes metadata to both sites before returning success to the client.

The object is fully protected against loss at the time of the ingest successful message. For example, if the local site is lost shortly after ingest, copies of both the object data and the object metadata still exist at the remote site. The object is fully retrievable.

If you instead used the same ILM rule and the “strong-site” consistency level, the client might receive a success message after object data is replicated to the remote site but before object metadata is distributed there. In this case, the level of protection of object metadata does not match the level of protection for object data. If the local site is lost shortly after ingest, object metadata is lost. The object can’t be retrieved.

The inter-relationship between consistency levels and ILM rules can be complex. Contact NetApp if you require assistance.

Related information

Manage objects with ILM

GET Bucket consistency

PUT Bucket consistency

How StorageGRID ILM rules manage objects

The grid administrator creates information lifecycle management (ILM) rules to manage object data ingested into the StorageGRID system from S3 REST API client applications. These rules are then added to the ILM policy to determine how and where object data is stored over time.

ILM settings determine the following aspects of an object:

- **Geography**

  The location of an object’s data, either within the StorageGRID system (storage pool) or in a Cloud Storage Pool.

- **Storage grade**

  The type of storage used to store object data: for example flash or spinning disk.

- **Loss protection**

  How many copies are made and the types of copies that are created: replication, erasure coding, or both.

- **Retention**

  The changes over time to how an object’s data is managed, where it is stored, and how it is protected from loss.

- **Protection during ingest**

  The method used to protect object data during ingest: synchronous placement (using the Balanced or Strict options for Ingest Behavior), or making interim copies (using the Dual commit option).

ILM rules can filter and select objects. For objects ingested using S3, ILM rules can filter objects based on the following metadata:

- Tenant account
- Bucket name
- Ingest time
- Key
- Last access time

ⓘ By default, updates to last access time are disabled for all S3 buckets. If your StorageGRID system includes an ILM rule that uses the Last access time option, you must enable updates to last access time for the S3 buckets specified in that rule. Use the PUT Bucket last access time request, the Tenant Manager (see Enable or disable last access time updates), or the Tenant Management API. When enabling last access time updates, be aware that StorageGRID performance might be reduced, especially in systems with small objects.

- Location constraint
- Object size
- User metadata
- Object tag

Related information

Use a tenant account

Manage objects with ILM

PUT Bucket last access time

Object versioning

You can use versioning to retain multiple versions of an object, which protects against accidental deletion of objects, and enables you to retrieve and restore earlier versions of an object.

The StorageGRID system implements versioning with support for most features, and with some limitations. StorageGRID supports up to 1,000 versions of each object.

Object versioning can be combined with StorageGRID information lifecycle management (ILM) or with S3 bucket lifecycle configuration. You must explicitly enable versioning for each bucket to turn on this functionality for the bucket. Each object in your bucket is assigned a version ID, which is generated by the StorageGRID system.

Using MFA (multi-factor authentication) Delete is not supported.

ⓘ Versioning can be enabled only on buckets created with StorageGRID version 10.3 or later.

ILM and versioning

ILM policies are applied to each version of an object. An ILM scanning process continuously scans all objects and re-evaluates them against the current ILM policy. Any changes you make to ILM policies are applied to all previously ingested objects. This includes previously ingested versions if versioning is enabled. ILM scanning applies new ILM changes to previously ingested objects.

For S3 objects in versioning-enabled buckets, versioning support allows you to create ILM rules that use “Noncurrent time” as the Reference time (select **Yes** for the question, “Apply this rule to older object versions only?” in Step 1 of the Create an ILM rule wizard). When an object is updated, its previous versions become noncurrent. Using a “Noncurrent time” filter allows you to create policies that reduce the storage impact of previous versions of objects.

When you upload a new version of an object using a multipart upload operation, the noncurrent time for the original version of the object reflects when the multipart upload was created for the new version, not when the multipart upload was completed. In limited cases, the noncurrent time for the original version might be hours or days earlier than the time for the current version.

See ILM rules and policies for S3 versioned objects (Example 4).

Use S3 REST API to configure S3 Object Lock

If the global S3 Object Lock setting is enabled for your StorageGRID system, you can create buckets with S3 Object Lock enabled. You can specify default retention for each bucket or retention settings for each object version.

How to enable S3 Object Lock for a bucket

If the global S3 Object Lock setting is enabled for your StorageGRID system, you can optionally enable S3 Object Lock when you create each bucket.

S3 Object Lock is a permanent setting that can only be enabled when you create a bucket. You can’t add or disable S3 Object Lock after a bucket is created.

To enable S3 Object Lock for a bucket, use either of these methods:

- Create the bucket using the Tenant Manager. See Create S3 bucket.
- Create the bucket using a PUT Bucket request with the `x-amz-bucket-object-lock-enabled` request header. See Operations on buckets.

S3 Object Lock requires bucket versioning, which is enabled automatically when the bucket is created. You can’t suspend versioning for the bucket. See Object versioning.

Default retention settings for a bucket

When S3 Object Lock is enabled for a bucket, you can optionally enable default retention for the bucket and specify a default retention mode and default retention period.

Default retention mode

- In COMPLIANCE mode:
  - The object can’t be deleted until its retain-until-date is reached.
  - The object’s retain-until-date can be increased, but it can’t be decreased.
  - The object’s retain-until-date can’t be removed until that date is reached.
- In GOVERNANCE mode:
  - Users with the `s3:BypassGovernanceRetention` permission can use the `x-amz-bypass-governance-retention: true` request header to bypass retention settings.
  - These users can delete an object version before its retain-until-date is reached.
  - These users can increase, decrease, or remove an object’s retain-until-date.

Default retention period

Each bucket can have a default retention period specified in years or days.

How to set default retention for a bucket

To set the default retention for a bucket, use either of these methods:

- Manage bucket settings from the Tenant Manager. See Create an S3 bucket and Update S3 Object Lock default retention.
- Issue a PUT Object Lock Configuration request for the bucket to specify the default mode and default number of days or years.

PUT Object Lock Configuration

The PUT Object Lock Configuration request allows you to set and modify the default retention mode and default retention period for a bucket that has S3 Object Lock enabled. You can also remove previously configured default retention settings.

When new object versions are ingested to the bucket, the default retention mode is applied if `x-amz-object-lock-mode` and `x-amz-object-lock-retain-until-date` aren't specified. The default retention period is used to calculate the retain-until-date if `x-amz-object-lock-retain-until-date` is not specified.

If the default retention period is modified after ingest of an object version, the retain-until-date of the object version remains the same and is not recalculated using the new default retention period.

You must have the `s3:PutBucketObjectLockConfiguration` permission, or be account root, to complete this operation.

The `Content-MD5` request header must be specified in the PUT request.

Request example

This example enables S3 Object Lock for a bucket and sets the default retention mode to COMPLIANCE and the default retention period to 6 years.

```
PUT /bucket?object-lock HTTP/1.1
Accept-Encoding: identity
Content-Length: 308
Host: host
Content-MD5: request header
User-Agent: s3sign/1.0.0 requests/2.24.0 python/3.8.2
X-Amz-Date: date
X-Amz-Content-SHA256: authorization-string
Authorization: authorization-string

<ObjectLockConfiguration>
    <ObjectLockEnabled>Enabled</ObjectLockEnabled>
    <Rule>
        <DefaultRetention>
            <Mode>COMPLIANCE</Mode>
            <Years>6</Years>
        </DefaultRetention>
    </Rule>
</ObjectLockConfiguration>
```

How to determine the default retention for a bucket

To determine if S3 Object Lock is enabled for a bucket and to see the default retention mode and retention period, use either of these methods:

- View the bucket in the Tenant Manager. See View S3 buckets.
- Issue a GET Object Lock Configuration request.

GET Object Lock Configuration

The GET Object Lock Configuration request allows you to determine if S3 Object Lock is enabled for a bucket and, if it is enabled, see if there is a default retention mode and retention period configured for the bucket.

When new object versions are ingested to the bucket, the default retention mode is applied if `x-amz-object-lock-mode` is not specified. The default retention period is used to calculate the retain-until-date if `x-amz-object-lock-retain-until-date` is not specified.

You must have the `s3:GetBucketObjectLockConfiguration` permission, or be account root, to complete this operation.

Request example

```
GET /bucket?object-lock HTTP/1.1
Host: host
Accept-Encoding: identity
User-Agent: aws-cli/1.18.106 Python/3.8.2 Linux/4.4.0-18362-Microsoft
botocore/1.17.29
x-amz-date: date
x-amz-content-sha256: authorization-string
Authorization: authorization-string
```

Response example

```
HTTP/1.1 200 OK
x-amz-id-2:
iVmcB7OXXJRkRH1FiVq1151/T24gRfpwpuZrEG11Bb9ImOMAAe98oxSpXlknabA0LTvBYJpSIX
k=
x-amz-request-id: B34E94CACB2CEF6D
Date: Fri, 04 Sep 2020 22:47:09 GMT
Transfer-Encoding: chunked
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<ObjectLockConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <ObjectLockEnabled>Enabled</ObjectLockEnabled>
    <Rule>
        <DefaultRetention>
            <Mode>COMPLIANCE</Mode>
            <Years>6</Years>
        </DefaultRetention>
    </Rule>
</ObjectLockConfiguration>
```

How to specify retention settings for an object

A bucket with S3 Object Lock enabled can contain a combination of objects with and without S3 Object Lock retention settings.

Object-level retention settings are specified using the S3 REST API. The retention settings for an object override any default retention settings for the bucket.

You can specify the following settings for each object:

- **Retention mode**: Either COMPLIANCE or GOVERNANCE.
- **Retain-until-date**: A date specifying how long the object version must be retained by StorageGRID.
    - In COMPLIANCE mode, if the retain-until-date is in the future, the object can be retrieved, but it can’t be modified or deleted. The retain-until-date can be increased, but this date can’t be decreased or

removed.

    - In GOVERNANCE mode, users with special permission can bypass the retain-until-date setting. They can delete an object version before its retention period has elapsed. They can also increase, decrease, or even remove the retain-until-date.

- **Legal hold**: Applying a legal hold to an object version immediately locks that object. For example, you might need to put a legal hold on an object that is related to an investigation or legal dispute. A legal hold has no expiration date, but remains in place until it is explicitly removed.

  The legal hold setting for an object is independent of the retention mode and the retain-until-date. If an object version is under a legal hold, no one can delete that version.

To specify S3 Object Lock settings when adding an object version to a bucket, issue a PUT Object, PUT Object - Copy, or Initiate Multipart Upload request.

You can use the following:

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

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

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

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

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

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

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

You can use the `s3:object-lock-remaining-retention-days` policy condition key to limit the minimum and maximum allowable retention periods for your objects.

How to update retention settings for an object

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

- `PUT Object legal-hold`

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

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

How to use GOVERNANCE mode

Users who have the `s3:BypassGovernanceRetention` permission can bypass the active retention settings of an object that uses GOVERNANCE mode. Any DELETE or PUT Object retention operations must include the `x-amz-bypass-governance-retention:true` request header. These users can perform these additional operations:

- Perform DELETE Object or DELETE Multiple Objects operations to delete an object version before its retention period has elapsed.

  Objects that are under a legal hold can't be deleted. Legal hold must be OFF.

- Perform PUT Object retention operations that change an object version's mode from GOVERNANCE to COMPLIANCE before the object's retention period has elapsed.

  Changing the mode from COMPLIANCE to GOVERNANCE is never allowed.

- Perform PUT Object retention operations to increase, decrease, or remove an object version's retention period.

Related information

- Manage objects with S3 Object Lock
- Use S3 Object Lock to retain objects
- Amazon Simple Storage Service User Guide: Using S3 Object Lock

Create S3 lifecycle configuration

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

The simple example in this section illustrates how an S3 lifecycle configuration can control when certain objects are deleted (expired) from specific S3 buckets. The example in this section is for illustration purposes only. For complete details on creating S3 lifecycle configurations, see Amazon Simple Storage Service Developer Guide: Object lifecycle management. Note that StorageGRID only supports Expiration actions; it does not support Transition actions.

What lifecycle configuration is

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

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

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

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

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

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

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

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

Create lifecycle configuration

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

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

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

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

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

Apply lifecycle configuration to bucket

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

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

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

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

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

A successful response lists the lifecycle configuration you just applied.

Validate that bucket lifecycle expiration applies to object

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

> ⓘ Because bucket lifecycle overrides ILM, the `expiry-date` shown is the actual date the object will be deleted. For details, see How object retention is determined.

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

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

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

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

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

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

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

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

Recommendations for implementing S3 REST API

You should follow these recommendations when implementing the S3 REST API for use with StorageGRID.

Recommendations for HEADs to non-existent objects

If your application routinely checks to see if an object exists at a path where you don’t expect the object to actually exist, you should use the “Available” consistency control. For example, you should use the “Available” consistency control if your application HEADs a location before PUT-ing to it.

Otherwise, if the HEAD operation does not find the object, you might receive a high number of 500 Internal Server errors if one or more Storage Nodes are unavailable.

You can set the “Available” consistency control for each bucket using the PUT Bucket consistency request, or you can specify the consistency control in the request header for an individual API operation.

Recommendations for object keys

Follow these recommendations for object key names, based on when the bucket was first created.

Buckets created in StorageGRID 11.4 or earlier

- Don’t use random values as the first four characters of object keys. This is in contrast to the former AWS recommendation for key prefixes. Instead, use non-random, non-unique prefixes, such as `image`.
- If you do follow the former AWS recommendation to use random and unique characters in key prefixes, prefix the object keys with a directory name. That is, use this format:

  ```
  mybucket/mydir/f8e3-image3132.jpg
  ```

  Instead of this format:

```
mybucket/f8e3-image3132.jpg
```

Buckets created in StorageGRID 11.4 or later

Restricting object key names to meet performance best practices is not required. In most cases, you can use random values for the first four characters of object key names.

> An exception to this is an S3 workload that continuously removes all objects after a short period of time. To minimize the performance impact for this use case, vary a leading portion of the key name every several thousand objects with something like the date. For example, suppose an S3 client typically writes 2,000 objects/second and the ILM or bucket lifecycle policy removes all objects after three days. To minimize the performance impact, you might name keys using a pattern like this: `/mybucket/mydir/`*`yyyymmddhhmmss-random_UUID`*`.jpg`

Recommendations for “range reads”

If the global option to compress stored objects is enabled, S3 client applications should avoid performing GET Object operations that specify a range of bytes be returned. These “range read” operations are inefficient because StorageGRID must effectively uncompress the objects to access the requested bytes. GET Object operations that request a small range of bytes from a very large object are especially inefficient; for example, it is inefficient to read a 10 MB range from a 50 GB compressed object.

If ranges are read from compressed objects, client requests can time out.

> If you need to compress objects and your client application must use range reads, increase the read timeout for the application.

Related information

- Consistency controls
- PUT Bucket consistency
- Administer StorageGRID

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: - The SHA256 value is not calculated for the body of the request. The user-submitted value is accepted without validation, as if the value `UNSIGNED-PAYLOAD` had been provided for the `x-amz-content-sha256` header.
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
GET Service `(ListBuckets)`	Implemented with all Amazon S3 REST API behavior. Subject to change without notice.
GET Storage Usage	The GET Storage Usage request tells you the total amount of storage in use by an account, and for each bucket associated with the account. This is an operation on the service with a path of / and a custom query parameter (`?x-ntap-sg-usage`) added.
OPTIONS /	Client applications can issue `OPTIONS /` requests to the S3 port on a Storage Node, without providing S3 authentication credentials, to determine whether the Storage Node is available. You can use this request for monitoring, or to allow external load balancers to identify when a Storage Node is down.

Related information

GET Storage Usage

Operations on buckets

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

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

See the following for more information:

- Amazon Web Services (AWS) Documentation: Bucket Restrictions and Limitations
- Configure S3 endpoint domain names

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

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

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

Operation	Implementation
DELETE Bucket	This operation deletes the bucket.
DELETE Bucket cors	This operation deletes the CORS configuration for the bucket.
DELETE Bucket encryption	This operation deletes the default encryption from the bucket. Existing encrypted objects remain encrypted, but any new objects added to the bucket aren't encrypted.
DELETE Bucket lifecycle	This operation deletes the lifecycle configuration from the bucket. See Create S3 lifecycle configuration.
DELETE Bucket policy	This operation deletes the policy attached to the bucket.
DELETE Bucket replication	This operation deletes the replication configuration attached to the bucket.
DELETE Bucket tagging	This operation uses the `tagging` subresource to remove all tags from a bucket.
GET Bucket (ListObjects) (ListObjectsV2)	This operation returns some or all (up to 1,000) of the objects in a bucket. The Storage Class for objects can have either of two values, even if the object was ingested with the `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.
GET Bucket Object versions (ListObjectVersions)	With READ access on a bucket, this operation with the `versions` subresource lists metadata of all of the versions of objects in the bucket.
GET Bucket acl	This operation returns a positive response and the ID, DisplayName, and Permission of the bucket owner, indicating that the owner has full access to the bucket.
GET Bucket cors	This operation returns the `cors` configuration for the bucket.
GET Bucket encryption	This operation returns the default encryption configuration for the bucket.

Operation	Implementation
GET Bucket lifecycle (GetBucketLifecycleConfiguration)	This operation returns the lifecycle configuration for the bucket. See Create S3 lifecycle configuration.
GET Bucket location	This operation returns the region that was set using the `LocationConstraint` element in the PUT Bucket request. If the bucket's region is `us-east-1`, an empty string is returned for the region.
GET Bucket notification (GetBucketNotificationConfiguration)	This operation returns the notification configuration attached to the bucket.
GET Bucket policy	This operation returns the policy attached to the bucket.
GET Bucket replication	This operation returns the replication configuration attached to the bucket.
GET Bucket tagging	This operation uses the `tagging` subresource to return all tags for a bucket.
GET Bucket versioning	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
GET Object Lock Configuration	This operation returns the bucket default retention mode and default retention period, if configured. See Use S3 REST API to configure S3 Object Lock.
HEAD Bucket	This operation 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.

Operation	Implementation
PUT Bucket	This operation 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 PUT Bucket 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.
PUT Bucket cors	This operation sets the CORS configuration for a bucket so that the bucket can service cross-origin requests. Cross-origin resource sharing (CORS) is a security mechanism that allows client web applications in one domain to access resources in a different domain. For example, suppose you use an S3 bucket named `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
PUT Bucket encryption	This operation sets the default encryption state of an existing bucket. When bucket-level encryption is enabled, any new objects added to the bucket are encrypted.StorageGRID supports server-side encryption with StorageGRID-managed keys. When specifying the server-side encryption configuration rule, set the `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).
PUT Bucket lifecycle (PutBucketLifecycleConfiguration)	This operation creates a new lifecycle configuration for the bucket or replaces an existing lifecycle configuration. StorageGRID supports up to 1,000 lifecycle rules in a lifecycle configuration. Each rule can include the following XML elements: • Expiration (Days, Date) • NoncurrentVersionExpiration (NoncurrentDays) • Filter (Prefix, Tag) • Status • ID StorageGRID does not support these actions: • AbortIncompleteMultipartUpload • ExpiredObjectDeleteMarker • 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

PUT Bucket notification
(PutBucketNotificationConfiguration)
	This operation configures notifications for the bucket using the notification configuration XML included in the request body. You should be aware of the following implementation details:
	• StorageGRID supports Simple Notification Service (SNS) 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`

PUT Bucket policy	This operation sets the policy attached to the bucket.

Operation	Implementation
PUT Bucket replication	This operation configures StorageGRID CloudMirror replication for the bucket using the replication configuration XML provided in the request body. For CloudMirror replication, you should be aware of the following implementation details: • 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 the Amazon S3 documentation on 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.
PUT Bucket tagging	This operation uses the `tagging` subresource to add or update a set of tags for a bucket. When adding bucket tags, be aware of the following limitations: • 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.

Operation	Implementation
PUT Bucket versioning	This implementation uses the `versioning` subresource to set the versioning state of an existing bucket. You can set the versioning state with one of the following values: • 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`.
PUT Object Lock Configuration	This operation 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.

Related information

Consistency controls

GET Bucket last access time

Use bucket and group access policies

S3 operations tracked in audit logs

Custom operations on buckets

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

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

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

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

Related information

S3 operations tracked in the audit logs

Operations on objects

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

The following conditions apply to all object operations:

- StorageGRID consistency controls are supported by all operations on objects, with the exception of the following:
  - GET Object ACL
  - `OPTIONS /`
  - PUT Object legal hold
  - PUT Object retention
  - SELECT Object content

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

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

Operation	Implementation
DELETE Object	Multi-Factor Authentication (MFA) and the response header `x-amz-mfa` aren’t supported. When processing a DELETE Object request, StorageGRID attempts to immediately remove all copies of the object from all stored locations. If successful, StorageGRID returns a response to the client immediately. If all copies 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 version enabled bucket, it results in the generation of a delete marker. The `versionId` for the delete marker is returned using the `x-amz-version-id` response header, and the `x-amz-delete-marker` response header is returned set to `true`. • If an object is deleted without the `versionId` subresource on a version suspended bucket, it results in a permanent deletion of an already existing 'null' version or a 'null' delete marker, and the generation of a new 'null' delete marker. The `x-amz-delete-marker` response header is returned set to `true`. **Note**: In certain cases, multiple delete markers might exist for an object. See Use S3 REST API to configure S3 Object Lock to learn how to delete object versions in GOVERNANCE mode.

Operation	Implementation
DELETE Multiple Objects (DeleteObjects)	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.
DELETE Object tagging	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`.
GET Object	GET Object
GET Object ACL	If the necessary access credentials are provided for the account, the operation returns a positive response and the ID, DisplayName, and Permission of the object owner, indicating that the owner has full access to the object.
GET Object legal hold	Use S3 REST API to configure S3 Object Lock
GET Object retention	Use S3 REST API to configure S3 Object Lock
GET Object tagging	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`.
HEAD Object	HEAD Object
POST Object restore	POST Object restore
PUT Object	PUT Object
PUT Object - Copy	PUT Object - Copy
PUT Object legal hold	Use S3 REST API to configure S3 Object Lock

Operation	Implementation
PUT Object retention	Use S3 REST API to configure S3 Object Lock
PUT Object tagging	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 PUT Object tagging to update an object’s tags, StorageGRID does not re-ingest the object. This means that the option for Ingest Behavior specified in the matching ILM rule is not used. Any changes to object placement that are triggered by the update are made when ILM is re-evaluated by normal background ILM processes. This means that if the ILM rule uses the Strict option for ingest behavior, no action is taken if the required object placements 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

Related information

S3 operations tracked in audit logs

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.

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

Use SSE

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

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

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

- PUT Object
- PUT Object - Copy
- Initiate Multipart Upload

Use SSE-C

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

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

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

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

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

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

- You must use https.

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

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

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

- If 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 Developer Guide: Protecting Data Using Server-Side Encryption with Customer-Provided Encryption Keys (SSE-C)

GET Object

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

GET object and multipart objects

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

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

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.

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 GET Object for Cloud Storage Pool objects

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

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

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

Multipart or segmented objects in a Cloud Storage Pool

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

In these cases:

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

GET Object 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 GET Object 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	• **SUCCESS**: 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.

Related information

S3 operations tracked in audit logs

HEAD Object

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

HEAD object and multipart objects

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

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

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.

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 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.

HEAD Object 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 HEAD object
Object ingested into StorageGRID but not yet evaluated by ILM, or object stored in a traditional storage pool or using erasure coding	`200 OK` (No special response header is returned.)

Object in Cloud Storage Pool but not yet transitioned to a non-retrievable state	`200 OK`
	`x-amz-storage-class: GLACIER`
	`x-amz-restore: ongoing-request="false", expiry-date="Sat, 23 July 20 2030 00:00:00 GMT"`
	Until the object is transitioned to a non-retrievable state, the value for `expiry-date` is set to some distant time in the future. The exact time of transition is not controlled by the StorageGRID system.

State of object	Response to HEAD object

Object has transitioned to non-retrievable state, but at least one copy also exists on the grid	`200 OK`
	`x-amz-storage-class: GLACIER`
	`x-amz-restore: ongoing-request="false", expiry-date="Sat, 23 July 20 2030 00:00:00 GMT"`
	The value for `expiry-date` is set to some distant time in the future.
	Note: If the copy on the grid is not available (for example, a Storage Node is down), you must issue a POST Object restore request to restore the copy from the Cloud Storage Pool before you can successfully retrieve the object.

Object transitioned to a non-retrievable state, and no copy exists on the grid	`200 OK`
	`x-amz-storage-class: GLACIER`

Object in process of being restored from a non-retrievable state	`200 OK`
	`x-amz-storage-class: GLACIER`
	`x-amz-restore: ongoing-request="true"`

Object fully restored to the Cloud Storage Pool	`200 OK`
	`x-amz-storage-class: GLACIER`
	`x-amz-restore: ongoing-request="false", expiry-date="Sat, 23 July 20 2018 00:00:00 GMT"`
	The `expiry-date` indicates when the object in the Cloud Storage Pool will be returned to a non-retrievable state.

Multipart or segmented objects in Cloud Storage Pool

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

HEAD Object 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 HEAD Object 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	• **SUCCESS**: 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.

Related information

S3 operations tracked in audit logs

POST Object restore

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

Supported request type

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

Versioning

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

Behavior of POST Object restore on Cloud Storage Pool objects

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

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

State of object	Behavior of POST Object restore
Object ingested into StorageGRID but not yet evaluated by ILM, or object is not in a Cloud Storage Pool	`403 Forbidden`, `InvalidObjectState`
Object in Cloud Storage Pool but not yet transitioned to a non-retrievable state	`200 OK` No changes are made. **Note**: Before an object has been transitioned to a non-retrievable state, you can't change its `expiry-date`.
Object transitioned to a non-retrievable state	`202 Accepted` Restores a retrievable copy of the object to the Cloud Storage Pool for the number of days specified in the request body. At the end of this period, the object is returned to a non-retrievable state. Optionally, use the `Tier` request element to determine how long the restore job will take to finish (`Expedited`, `Standard`, or `Bulk`). If you 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 POST Object restore request with a new value for `Days`. The restoration date is updated relative to the time of the request.

Related information

Manage objects with ILM

HEAD Object

S3 operations tracked in audit logs

PUT Object

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

Resolve conflicts

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

Object size

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

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

User metadata size

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

UTF-8 characters in user metadata

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

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

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

Object tag limits

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

Object ownership

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

Supported request headers

The following request headers are supported:

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

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

  - StorageGRID does not verify the `chunk-signature` against the chunk data.

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

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

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

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

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

  ```
  x-amz-meta-name: value
  ```

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

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

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

  An ILM rule can’t use both a **User defined creation time** for the Reference time and the Balanced or Strict options for Ingest Behavior. An error is returned when the ILM rule is created.

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

      If a request is made without these headers, the bucket default retention settings are used to calculate the object version 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:

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

Storage class options

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

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

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

- STANDARD (Default)
  - **Dual commit**: If the ILM rule specifies the Dual commit option for Ingest Behavior, as soon as an object is ingested a second copy of that object is created and distributed to a different Storage Node (dual commit). When the ILM is evaluated, StorageGRID determines if these initial interim copies satisfy the placement instructions in the rule. If they 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 policy, 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`
- **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.

ⓘ 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`.

> ⓘ 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

Operations on buckets

S3 operations tracked in audit logs

How client connections can be configured

PUT Object - Copy

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

Resolve conflicts

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

Object size

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

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

UTF-8 characters in user metadata

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

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

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

Supported request headers

The following request headers are supported:

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

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

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

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

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

    If a request is made without these headers, the bucket default retention settings are used to calculate the object version 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-website-redirect-location`

Storage class options

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

- `STANDARD`

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

- `REDUCED_REDUNDANCY`

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

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

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

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

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

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

  This means that if the ILM rule uses the Strict option for ingest behavior, no action is taken if the required object placements 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 PUT Object - Copy request, so the object can be decrypted and then copied:
  - `x-amz-copy-source-server-side-encryption-customer-algorithm`: Specify `AES256`.
  - `x-amz-copy-source-server-side-encryption-customer-key`: Specify the encryption key you provided when you created the source object.
  - `x-amz-copy-source-server-side-encryption-customer-key-MD5`: Specify the MD5 digest you provided when you created the source object.
- If you want to encrypt the target object (the copy) with a unique key that you provide and manage, include the following three headers:
  - `x-amz-server-side-encryption-customer-algorithm`: Specify `AES256`.
  - `x-amz-server-side-encryption-customer-key`: Specify a new encryption key for the target object.
  - `x-amz-server-side-encryption-customer-key-MD5`: Specify the MD5 digest of the new encryption key.

  > ⓘ 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 PUT Object - Copy request:
  - `x-amz-server-side-encryption`

  > ⓘ 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.

Related information

Manage objects with ILM

S3 operations tracked in audit logs

PUT Object

SelectObjectContent

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

For more information see the AWS documentation for SelectObjectContent.

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.

ⓘ Use of ScanRange is not supported.

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

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 List Multipart Uploads queries for that bucket might return incomplete results.
- StorageGRID enforces AWS size limits for multipart parts. S3 clients must follow these guidelines:
  - Each part in a multipart upload must be between 5 MiB (5,242,880 bytes) and 5 GiB (5,368,709,120 bytes).
  - The last part can be smaller than 5 MiB (5,242,880 bytes).
  - In general, part sizes should be as large as possible. For example, use part sizes of 5 GiB for a 100 GiB object. 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 behavior. You should be aware of how this affects object and part placement:
  - If ILM changes while an S3 multipart upload is in progress, when the multipart upload completes some parts of the object might not meet current ILM requirements. Any part that is not placed correctly is queued for ILM re-evaluation, and is moved to the correct location later.
  - When evaluating ILM for a part, StorageGRID filters on the size of the part, not the size of the object. This means that parts of an object can be stored in locations that 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, at ingest each 1 GB part of a 10-part multipart upload is

stored at DC2. When ILM is evaluated for the object as a whole, all parts of the object are moved to DC1.

- All of the multipart upload operations support StorageGRID consistency controls.
- As required, you can use server-side encryption with multipart uploads. To use SSE (server-side encryption with StorageGRID-managed keys), you include the `x-amz-server-side-encryption` request header in the Initiate Multipart Upload request only. To use SSE-C (server-side encryption with customer-provided keys), you specify the same three encryption key request headers in the Initiate Multipart Upload request and in each subsequent Upload Part request.

Operation	Implementation
List Multipart Uploads	See List Multipart Uploads
Initiate Multipart Upload	See Initiate Multipart Upload
Upload Part	See Upload Part
Upload Part - Copy	See Upload Part - Copy
Complete Multipart Upload	See Complete Multipart Upload
Abort Multipart Upload	Implemented with all Amazon S3 REST API behavior. Subject to change without notice.
List Parts	Implemented with all Amazon S3 REST API behavior. Subject to change without notice.

Related information

- Consistency controls
- Use server-side encryption

List Multipart Uploads

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

The following request parameters are supported:

- `encoding-type`
- `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. When the Complete Multipart Upload operation is performed, that is the point when objects are created (and versioned if applicable).

Initiate Multipart Upload

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

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

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

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

- `STANDARD` (Default)
  - **Dual commit**: If the ILM rule specifies the Dual commit option for Ingest Behavior, as soon as an object is ingested a second copy of that object is created and distributed to a different Storage Node (dual commit). When the ILM is evaluated, StorageGRID determines if these initial interim copies satisfy the placement instructions in the rule. If they 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 policy, 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 the documentation for PUT Object.

Request headers for server-side encryption

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

- **SSE**: Use the following header in the Initiate Multipart Upload request if you want to encrypt the object with a unique key managed by StorageGRID. Don’t specify this header in any of the Upload Part requests.
  - `x-amz-server-side-encryption`
- **SSE-C**: Use all three of these headers in the Initiate Multipart Upload request (and in each subsequent Upload Part request) if you want to encrypt the object with a unique key that you provide and manage.
  - `x-amz-server-side-encryption-customer-algorithm`: Specify `AES256`.
  - `x-amz-server-side-encryption-customer-key`: Specify your encryption key for the new object.
  - `x-amz-server-side-encryption-customer-key-MD5`: Specify the MD5 digest of the new object’s encryption key.

> 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 Complete Multipart Upload operation is performed.

Related information

Manage objects with ILM

PUT Object

Upload Part

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

Supported request headers

The following request headers are supported:

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

Request headers for server-side encryption

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

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

> The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object. Before using customer-provided keys to secure object data, review the considerations in “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 Complete Multipart Upload operation is performed.

Related information

Use server-side encryption

Upload Part - Copy

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

The Upload Part - Copy operation is implemented with all Amazon S3 REST API behavior. 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 Initiate Multipart Upload request, you must also include the following request headers in each Upload Part - Copy request:

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

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

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

> ⓘ The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object. Before using customer-provided keys to secure object data, review the considerations in “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 Complete Multipart Upload operation is performed.

Complete Multipart Upload

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

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 an Ingest Behavior of Dual commit or Balanced.

- `STANDARD`

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

- `REDUCED_REDUNDANCY`

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

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

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

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

Versioning

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

Related information

Manage objects with ILM

Error responses

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

Supported S3 API error codes

Name	HTTP status
AccessDenied	403 Forbidden
BadDigest	400 Bad Request
BucketAlreadyExists	409 Conflict
BucketNotEmpty	409 Conflict
IncompleteBody	400 Bad Request
InternalError	500 Internal Server Error
InvalidAccessKeyId	403 Forbidden
InvalidArgument	400 Bad Request
InvalidBucketName	400 Bad Request
InvalidBucketState	409 Conflict
InvalidDigest	400 Bad Request
InvalidEncryptionAlgorithmError	400 Bad Request
InvalidPart	400 Bad Request
InvalidPartOrder	400 Bad Request
InvalidRange	416 Requested Range Not Satisfiable
InvalidRequest	400 Bad Request
InvalidStorageClass	400 Bad Request
InvalidTag	400 Bad Request
InvalidURI	400 Bad Request
KeyTooLong	400 Bad Request
MalformedXML	400 Bad Request

Name	HTTP status
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
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

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

StorageGRID S3 requests

GET Bucket consistency

The GET Bucket consistency request allows you to determine the consistency level being applied to a particular bucket.

The default consistency controls are set to guarantee read-after-write for newly created objects.

You have the s3:GetBucketConsistency permission, or be account root, to complete this operation.

Request example

```
GET /bucket?x-ntap-sg-consistency HTTP/1.1
Date: date
Authorization: authorization string
Host: host
```

Response

In the response XML, `<Consistency>` will return one of the following values:

Consistency control	Description
all	All nodes receive the data immediately, or the request will fail.
strong-global	Guarantees read-after-write consistency for all client requests across all sites.
strong-site	Guarantees read-after-write consistency for all client requests within a site.
read-after-new-write	(Default) Provides read-after-write consistency for new objects and eventual consistency for object updates. Offers high availability and data protection guarantees. Recommended for most cases.
available	Provides eventual consistency for both new objects and object updates. For S3 buckets, use only as required (for example, for a bucket that contains log values that are rarely read, or for HEAD or GET operations on keys that don't exist). Not supported for S3 FabricPool buckets.

Response example

```
HTTP/1.1 200 OK
Date: Fri, 18 Sep 2020 01:02:18 GMT
Connection: CLOSE
Server: StorageGRID/11.5.0
x-amz-request-id: 12345
Content-Length: 127
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<Consistency xmlns="http://s3.storagegrid.com/doc/2015-02-01/">read-after-
new-write</Consistency>
```

Related information

Consistency controls

PUT Bucket consistency

The PUT Bucket consistency request allows you to specify the consistency level to apply to operations performed on a bucket.

The default consistency controls are set to guarantee read-after-write for newly created objects.

Before you begin

You have the s3:PutBucketConsistency permission, or be account root, to complete this operation.

Request

The `x-ntap-sg-consistency` parameter must contain one of the following values:

Consistency control	Description
all	All nodes receive the data immediately, or the request will fail.
strong-global	Guarantees read-after-write consistency for all client requests across all sites.
strong-site	Guarantees read-after-write consistency for all client requests within a site.
read-after-new-write	(Default) Provides read-after-write consistency for new objects and eventual consistency for object updates. Offers high availability and data protection guarantees. Recommended for most cases.
available	Provides eventual consistency for both new objects and object updates. For S3 buckets, use only as required (for example, for a bucket that contains log values that are rarely read, or for HEAD or GET operations on keys that don't exist). Not supported for S3 FabricPool buckets.

Note: In general, you should use the "read-after-new-write" consistency control value. If requests aren't working correctly, change the application client behavior if possible. Or, configure the client to specify the consistency control for each API request. Set the consistency control at the bucket level only as a last resort.

Request example

```
PUT /bucket?x-ntap-sg-consistency=strong-global HTTP/1.1
Date: date
Authorization: authorization string
Host: host
```

Related information

Consistency controls

GET Bucket last access time

The GET Bucket last access time request allows you to determine if last access time updates are enabled or disabled for individual buckets.

You have the s3:GetBucketLastAccessTime permission, or be account root, to complete this operation.

Request example

```
GET /bucket?x-ntap-sg-lastaccesstime HTTP/1.1
Date: date
Authorization: authorization string
Host: host
```

Response example

This example shows that last access time updates are enabled for the bucket.

```
HTTP/1.1 200 OK
Date: Sat, 29 Nov 2015 01:02:18 GMT
Connection: CLOSE
Server: StorageGRID/10.3.0
x-amz-request-id: 12345
Content-Length: 127
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<LastAccessTime xmlns="http://s3.storagegrid.com/doc/2015-02-01/">enabled
</LastAccessTime>
```

PUT Bucket last access time

The PUT Bucket last access time request allows you to enable or disable last access time updates for individual buckets. Disabling last access time updates improves performance, and is the default setting for all buckets created with version 10.3.0, or later.

You have the s3:PutBucketLastAccessTime permission for a bucket, or be account root, to complete this operation.

> Starting with StorageGRID version 10.3, updates to last access time are disabled by default for all new buckets. If you have buckets that were created using an earlier version of StorageGRID and you want to match the new default behavior, you must explicitly disable last access time updates for each of those earlier buckets. You can enable or disable updates to last access time using the PUT Bucket last access time request, the **S3** > **Buckets** > **Change Last Access Setting** checkbox in the Tenant Manager, or the Tenant Management API.

If last access time updates are disabled for a bucket, the following behavior is applied to operations on the bucket:

- GET Object, GET Object ACL, GET Object Tagging, and HEAD Object requests don’t update last access time. The object is not added to queues for information lifecycle management (ILM) evaluation.
- PUT Object - Copy and PUT Object Tagging requests that update only the metadata also update last access time. The object is added to queues for ILM evaluation.
- If updates to last access time are disabled for the source bucket, PUT Object - Copy requests don’t update last access time for the source bucket. The object that was copied is not added to queues for ILM

evaluation for the source bucket. However, for the destination, PUT Object - Copy requests always update last access time. The copy of the object is added to queues for ILM evaluation.

- Complete Multipart Upload requests update last access time. The completed object is added to queues for ILM evaluation.

Request examples

This example enables last access time for a bucket.

```
PUT /bucket?x-ntap-sg-lastaccesstime=enabled HTTP/1.1
Date: date
Authorization: authorization string
Host: host
```

This example disables last access time for a bucket.

```
PUT /bucket?x-ntap-sg-lastaccesstime=disabled HTTP/1.1
Date: date
Authorization: authorization string
Host: host
```

Related information

Use a tenant account

DELETE Bucket metadata notification configuration

The DELETE Bucket metadata notification configuration request allows you to disable the search integration service for individual buckets by deleting the configuration XML.

You have the s3:DeleteBucketMetadataNotification permission for a bucket, or be account root, to complete this operation.

Request example

This example shows disabling the search integration service for a bucket.

```
DELETE /test1?x-ntap-sg-metadata-notification HTTP/1.1
Date: date
Authorization: authorization string
Host: host
```

GET Bucket metadata notification configuration

The GET Bucket metadata notification configuration request allows you to retrieve the configuration XML used to configure search integration for individual buckets.

You have the s3:GetBucketMetadataNotification permission, or be account root, to complete this operation.

Request example

This request retrieves the metadata notification configuration for the bucket named `bucket`.

```
GET /bucket?x-ntap-sg-metadata-notification HTTP/1.1
Date: date
Authorization: authorization string
Host: host
```

Response

The response body includes the metadata notification configuration for the bucket. The metadata notification configuration lets you determine how the bucket is configured for search integration. That is, it allows you to determine which objects are indexed, and which endpoints their object metadata is being sent to.

```
<MetadataNotificationConfiguration>
    <Rule>
        <ID>Rule-1</ID>
        <Status>rule-status</Status>
        <Prefix>key-prefix</Prefix>
        <Destination>
           <Urn>arn:aws:es:_region:account-
ID_:domain/_mydomain/myindex/mytype_</Urn>
        </Destination>
    </Rule>
    <Rule>
        <ID>Rule-2</ID>
         ...
    </Rule>
     ...
</MetadataNotificationConfiguration>
```

Each metadata notification configuration includes one or more rules. Each rule specifies the objects that it applies to and the destination where StorageGRID should send object metadata. Destinations must be specified using the URN of a StorageGRID endpoint.

Name	Description	Required
MetadataNotificationConfiguration	Container tag for rules used to specify the objects and destination for metadata notifications. Contains one or more Rule elements.	Yes

Name	Description	Required
Rule	Container tag for a rule that identifies the objects whose metadata should be added to a specified index. Rules with overlapping prefixes are rejected. Included in the MetadataNotificationConfiguration element.	Yes
ID	Unique identifier for the rule. Included in the Rule element.	No
Status	Status can be 'Enabled' or 'Disabled'. No action is taken for rules that are disabled. Included in the Rule element.	Yes
Prefix	Objects that match the prefix are affected by the rule, and their metadata is sent to the specified destination. To match all objects, specify an empty prefix. Included in the Rule element.	Yes
Destination	Container tag for the destination of a rule. Included in the Rule element.	Yes

Name	Description	Required
Urn	URN of the destination where object metadata is sent. Must be the URN of a StorageGRID endpoint with the following properties: • `es` must be the third element. • The URN must end with the index and type where the metadata is stored, in the form `domain-name/myindex/mytype`. Endpoints are configured using the Tenant Manager or Tenant Management API. They take the following form: • `arn:aws:es:_region:account-ID_:domain/mydomain/myindex/mytype` • `urn:mysite:es:::mydomain/myindex/mytype` The endpoint must be configured before the configuration XML is submitted, or configuration will fail with a 404 error. Urn is included in the Destination element.	Yes

Response example

The XML included between the `<MetadataNotificationConfiguration></MetadataNotificationConfiguration>` tags shows how integration with a search integration endpoint is configured for the bucket. In this example, object metadata is being sent to an Elasticsearch index named `current` and type named `2017` that is hosted in an AWS domain named `records`.

```
HTTP/1.1 200 OK
Date: Thu, 20 Jul 2017 18:24:05 GMT
Connection: KEEP-ALIVE
Server: StorageGRID/11.0.0
x-amz-request-id: 3832973499
Content-Length: 264
Content-Type: application/xml

<MetadataNotificationConfiguration>
    <Rule>
        <ID>Rule-1</ID>
        <Status>Enabled</Status>
        <Prefix>2017</Prefix>
        <Destination>
           <Urn>arn:aws:es:us-east-
1:3333333:domain/records/current/2017</Urn>
        </Destination>
    </Rule>
</MetadataNotificationConfiguration>
```

Related information

Use a tenant account

PUT Bucket metadata notification configuration

The PUT Bucket metadata notification configuration request allows you to enable the search integration service for individual buckets. The metadata notification configuration XML that you supply in the request body specifies the objects whose metadata is sent to the destination search index.

You have the s3:PutBucketMetadataNotification permission for a bucket, or be account root, to complete this operation.

Request

The request must include the metadata notification configuration in the request body. Each metadata notification configuration includes one or more rules. Each rule specifies the objects that it applies to, and the destination where StorageGRID should send object metadata.

Objects can be filtered on the prefix of the object name. For example, you could send metadata for objects with the prefix `/images` to one destination, and objects with the prefix `/videos` to another.

Configurations that have overlapping prefixes aren’t valid, and are rejected when they are submitted. For example, a configuration that included one rule for for objects with the prefix `test` and a second rule for objects with the prefix `test2` would not be allowed.

Destinations must be specified using the URN of a StorageGRID endpoint. The endpoint must exist when the metadata notification configuration is submitted, or the request fails as a `400 Bad Request`. The error message states: `Unable to save the metadata notification (search) policy. The specified endpoint URN does not exist:` *`URN`*`.`

```
<MetadataNotificationConfiguration>
    <Rule>
        <ID>Rule-1</ID>
        <Status>rule-status</Status>
        <Prefix>key-prefix</Prefix>
        <Destination>
           <Urn>arn:aws:es:region:account-
ID:domain/mydomain/myindex/mytype</Urn>
        </Destination>
    </Rule>
    <Rule>
        <ID>Rule-2</ID>
         ...
    </Rule>
     ...
</MetadataNotificationConfiguration>
```

The table describes the elements in the metadata notification configuration XML.

Name	Description	Required
MetadataNotificationConfiguration	Container tag for rules used to specify the objects and destination for metadata notifications. Contains one or more Rule elements.	Yes
Rule	Container tag for a rule that identifies the objects whose metadata should be added to a specified index. Rules with overlapping prefixes are rejected. Included in the MetadataNotificationConfiguration element.	Yes
ID	Unique identifier for the rule. Included in the Rule element.	No
Status	Status can be 'Enabled' or 'Disabled'. No action is taken for rules that are disabled. Included in the Rule element.	Yes

Name	Description	Required
Prefix	Objects that match the prefix are affected by the rule, and their metadata is sent to the specified destination. To match all objects, specify an empty prefix. Included in the Rule element.	Yes
Destination	Container tag for the destination of a rule. Included in the Rule element.	Yes
Urn	URN of the destination where object metadata is sent. Must be the URN of a StorageGRID endpoint with the following properties: • `es` must be the third element. • The URN must end with the index and type where the metadata is stored, in the form `domain-name/myindex/mytype`. Endpoints are configured using the Tenant Manager or Tenant Management API. They take the following form: • `arn:aws:es:region:account-ID:domain/mydomain/myindex/mytype` • `urn:mysite:es:::mydomain/myindex/mytype` The endpoint must be configured before the configuration XML is submitted, or configuration will fail with a 404 error. Urn is included in the Destination element.	Yes

Request examples

This example shows enabling search integration for a bucket. In this example, object metadata for all objects is sent to the same destination.

```
PUT /test1?x-ntap-sg-metadata-notification HTTP/1.1
Date: date
Authorization: authorization string
Host: host

<MetadataNotificationConfiguration>
    <Rule>
        <ID>Rule-1</ID>
        <Status>Enabled</Status>
        <Prefix></Prefix>
        <Destination>
           <Urn>urn:sgws:es:::sgws-notifications/test1/all</Urn>
        </Destination>
    </Rule>
</MetadataNotificationConfiguration>
```

In this example, object metadata for objects that match the prefix `/images` is sent to one destination, while object metadata for objects that match the prefix `/videos` is sent to a second destination.

```
PUT /graphics?x-ntap-sg-metadata-notification HTTP/1.1
Date: date
Authorization: authorization string
Host: host

<MetadataNotificationConfiguration>
    <Rule>
        <ID>Images-rule</ID>
        <Status>Enabled</Status>
        <Prefix>/images</Prefix>
        <Destination>
           <Urn>arn:aws:es:us-east-1:3333333:domain/es-
domain/graphics/imagetype</Urn>
        </Destination>
    </Rule>
    <Rule>
        <ID>Videos-rule</ID>
        <Status>Enabled</Status>
        <Prefix>/videos</Prefix>
        <Destination>
           <Urn>arn:aws:es:us-west-1:22222222:domain/es-
domain/graphics/videotype</Urn>
        </Destination>
    </Rule>
</MetadataNotificationConfiguration>
```

JSON generated by the search integration service

When you enable the search integration service for a bucket, a JSON document is generated and sent to the destination endpoint each time object metadata or tags are added, updated, or deleted.

This example shows an example of the JSON that could be generated when an object with the key `SGWS/Tagging.txt` is created in a bucket named `test`. The `test` bucket is not versioned, so the `versionId` tag is empty.

```
{
  "bucket": "test",
  "key": "SGWS/Tagging.txt",
  "versionId": "",
  "accountId": "86928401983529626822",
  "size": 38,
  "md5": "3d6c7634a85436eee06d43415012855",
  "region":"us-east-1"
  "metadata": {
    "age": "25"
  },
  "tags": {
    "color": "yellow"
  }
}
```

Object metadata included in metadata notifications

The table lists all the fields that are included in the JSON document that is sent to the destination endpoint when search integration is enabled.

The document name includes the bucket name, object name, and version ID if present.

Type	Item name	Description
Bucket and object information	bucket	Name of the bucket
Bucket and object information	key	Object key name
Bucket and object information	versionID	Object version, for objects in versioned buckets
Bucket and object information	region	Bucket region, for example `us-east-1`
System metadata	size	Object size (in bytes) as visible to an HTTP client

Type	Item name	Description
System metadata	md5	Object hash
User metadata	metadata *key:value*	All user metadata for the object, as key-value pairs
Tags	tags *key:value*	All object tags defined for the object, as key-value pairs

For tags and user metadata, StorageGRID passes dates and numbers to Elasticsearch as strings or as S3 event notifications. To configure Elasticsearch to interpret these strings as dates or numbers, follow the Elasticsearch instructions for dynamic field mapping and for mapping date formats. You must enable the dynamic field mappings on the index before you configure the search integration service. After a document is indexed, you can't edit the document's field types in the index.

Related information

Use a tenant account

GET Storage Usage request

The GET Storage Usage request tells you the total amount of storage in use by an account, and for each bucket associated with the account.

The amount of storage used by an account and its buckets can be obtained by a modified GET Service request with the `x-ntap-sg-usage` query parameter. Bucket storage usage is tracked separately from the PUT and DELETE requests processed by the system. There might be some delay before the usage values match the expected values based on the processing of requests, particularly if the system is under heavy load.

By default, StorageGRID attempts to retrieve usage information using strong-global consistency. If strong-global consistency can't be achieved, StorageGRID attempts to retrieve the usage information at a strong-site consistency.

You have the s3:ListAllMyBuckets permission, or be account root, to complete this operation.

Request example

```
GET /?x-ntap-sg-usage HTTP/1.1
Date: date
Authorization: authorization string
Host: host
```

Response example

This example shows an account that has four objects and 12 bytes of data in two buckets. Each bucket contains two objects and six bytes of data.

```
HTTP/1.1 200 OK
Date: Sat, 29 Nov 2015 00:49:05 GMT
Connection: KEEP-ALIVE
Server: StorageGRID/10.2.0
x-amz-request-id: 727237123
Content-Length: 427
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<UsageResult xmlns="http://s3.storagegrid.com/doc/2015-02-01">
<CalculationTime>2014-11-19T05:30:11.000000Z</CalculationTime>
<ObjectCount>4</ObjectCount>
<DataBytes>12</DataBytes>
<Buckets>
<Bucket>
<Name>bucket1</Name>
<ObjectCount>2</ObjectCount>
<DataBytes>6</DataBytes>
</Bucket>
<Bucket>
<Name>bucket2</Name>
<ObjectCount>2</ObjectCount>
<DataBytes>6</DataBytes>
</Bucket>
</Buckets>
</UsageResult>
```

Versioning

Every object version stored will contribute to the `ObjectCount` and `DataBytes` values in the response. Delete markers aren’t added to the `ObjectCount` total.

Related information

Consistency controls

Deprecated bucket requests for legacy Compliance

You might need to use the StorageGRID S3 REST API to manage buckets that were created using the legacy Compliance feature.

Compliance feature deprecated

The StorageGRID Compliance feature that was available in previous StorageGRID versions is deprecated and has been replaced by S3 Object Lock.

If you previously enabled the global Compliance setting, the global S3 Object Lock setting is enabled in StorageGRID 11.6. You can no longer create new buckets with Compliance enabled; however, as required, you can use the StorageGRID S3 REST API to manage any existing legacy Compliant buckets.

- Use S3 REST API to configure S3 Object Lock
- Manage objects with ILM
- NetApp Knowledge Base: How to manage legacy Compliant buckets in StorageGRID 11.5

Deprecated compliance requests:

- Deprecated - PUT Bucket request modifications for compliance

  The SGCompliance XML element is deprecated. Previously, you could include this StorageGRID custom element in the optional XML request body of PUT Bucket requests to create a Compliant bucket.

- Deprecated - GET Bucket compliance

  The GET Bucket compliance request is deprecated. However, you can continue to use this request to determine the compliance settings currently in effect for an existing legacy Compliant bucket.

- Deprecated - PUT Bucket compliance

  The PUT Bucket compliance request is deprecated. However, you can continue to use this request to modify the compliance settings for an existing legacy Compliant bucket. For example, you can place an existing bucket on legal hold or increase its retention period.

Deprecated: PUT Bucket request modifications for compliance

The SGCompliance XML element is deprecated. Previously, you could include this StorageGRID custom element in the optional XML request body of PUT Bucket requests to create a Compliant bucket.

> ⓘ The StorageGRID Compliance feature that was available in previous StorageGRID versions is deprecated and has been replaced by S3 Object Lock.

Use S3 REST API to configure S3 Object Lock

Manage objects with ILM

NetApp Knowledge Base: How to manage legacy Compliant buckets in StorageGRID 11.5

You can no longer create new buckets with Compliance enabled. The following error message is returned if you attempt to use the PUT Bucket request modifications for compliance to create a new Compliant bucket:

```
The Compliance feature is deprecated.
Contact your StorageGRID administrator if you need to create new Compliant
buckets.
```

Deprecated: GET Bucket compliance request

The GET Bucket compliance request is deprecated. However, you can continue to use this request to determine the compliance settings currently in effect for an existing legacy

Compliant bucket.

> The StorageGRID Compliance feature that was available in previous StorageGRID versions is deprecated and has been replaced by S3 Object Lock.

Use S3 REST API to configure S3 Object Lock

Manage objects with ILM

NetApp Knowledge Base: How to manage legacy Compliant buckets in StorageGRID 11.5

You have the s3:GetBucketCompliance permission, or be account root, to complete this operation.

Request example

This example request allows you to determine the compliance settings for the bucket named `mybucket`.

```
GET /mybucket/?x-ntap-sg-compliance HTTP/1.1
Date: date
Authorization: authorization string
Host: host
```

Response example

In the response XML, `<SGCompliance>` lists the compliance settings in effect for the bucket. This example response shows the compliance settings for a bucket in which each object will be retained for one year (525,600 minutes), starting from when the object is ingested into the grid. There is currently no legal hold on this bucket. Each object will be automatically deleted after one year.

```
HTTP/1.1 200 OK
Date: date
Connection: connection
Server: StorageGRID/11.1.0
x-amz-request-id: request ID
Content-Length: length
Content-Type: application/xml

<SGCompliance>
  <RetentionPeriodMinutes>525600</RetentionPeriodMinutes>
  <LegalHold>false</LegalHold>
  <AutoDelete>true</AutoDelete>
</SGCompliance>
```

Name	Description
RetentionPeriodMinutes	The length of the retention period for objects added to this bucket, in minutes. The retention period starts when the object is ingested into the grid.
LegalHold	• True: This bucket is currently under a legal hold. Objects in this bucket can’t be deleted until the legal hold is lifted, even if their retention period has expired. • False: This bucket is not currently under a legal hold. Objects in this bucket can be deleted when their retention period expires.
AutoDelete	• True: The objects in this bucket will be deleted automatically when their retention period expires, unless the bucket is under a legal hold. • False: The objects in this bucket will not be deleted automatically when the retention period expires. You must delete these objects manually if you need to delete them.

Error responses

If the bucket was not created to be compliant, the HTTP status code for the response is `404 Not Found`, with an S3 error code of `XNoSuchBucketCompliance`.

Deprecated: PUT Bucket compliance request

The PUT Bucket compliance request is deprecated. However, you can continue to use this request to modify the compliance settings for an existing legacy Compliant bucket. For example, you can place an existing bucket on legal hold or increase its retention period.

ⓘ The StorageGRID Compliance feature that was available in previous StorageGRID versions is deprecated and has been replaced by S3 Object Lock.

Use S3 REST API to configure S3 Object Lock

Manage objects with ILM

NetApp Knowledge Base: How to manage legacy Compliant buckets in StorageGRID 11.5

You have the s3:PutBucketCompliance permission, or be account root, to complete this operation.

You must specify a value for every field of the compliance settings when issuing a PUT Bucket compliance request.

Request example

This example request modifies the compliance settings for the bucket named `mybucket`. In this example, objects in `mybucket` will now be retained for two years (1,051,200 minutes) instead of one year, starting from when the object is ingested into the grid. There is no legal hold on this bucket. Each object will be automatically deleted after two years.

```
PUT /mybucket/?x-ntap-sg-compliance HTTP/1.1
Date: date
Authorization: authorization name
Host: host
Content-Length: 152

<SGCompliance>
  <RetentionPeriodMinutes>1051200</RetentionPeriodMinutes>
  <LegalHold>false</LegalHold>
  <AutoDelete>true</AutoDelete>
</SGCompliance>
```

Name	Description
RetentionPeriodMinutes	The length of the retention period for objects added to this bucket, in minutes. The retention period starts when the object is ingested into the grid. **Important** When specifying a new value for RetentionPeriodMinutes, you must specify a value that is equal to or greater than the bucket's current retention period. After the bucket's retention period is set, you can't decrease that value; you can only increase it.
LegalHold	• True: This bucket is currently under a legal hold. Objects in this bucket can't be deleted until the legal hold is lifted, even if their retention period has expired. • False: This bucket is not currently under a legal hold. Objects in this bucket can be deleted when their retention period expires.
AutoDelete	• True: The objects in this bucket will be deleted automatically when their retention period expires, unless the bucket is under a legal hold. • False: The objects in this bucket will not be deleted automatically when the retention period expires. You must delete these objects manually if you need to delete them.

Consistency level for compliance settings

When you update the compliance settings for an S3 bucket with a PUT Bucket compliance request, StorageGRID attempts to update the bucket's metadata across the grid. By default, StorageGRID uses the **strong-global** consistency level to guarantee that all data center sites and all Storage Nodes that contain bucket metadata have read-after-write consistency for the changed compliance settings.

If StorageGRID can't achieve the **strong-global** consistency level because a data center site or multiple Storage Nodes at a site are unavailable, the HTTP status code for the response is `503 Service Unavailable.`

If you receive this response, you must contact the grid administrator to ensure that the required storage services are made available as soon as possible. If the grid administrator is unable to make enough of the Storage Nodes at each site available, technical support might direct you to retry the failed request by forcing the **strong-site** consistency level.

> Never force the **strong-site** consistency level for PUT bucket compliance unless you have been directed to do so by technical support and unless you understand the potential consequences of using this level.

When the consistency level is reduced to **strong-site**, StorageGRID guarantees that updated compliance settings will have read-after-write consistency only for client requests within a site. This means that the StorageGRID system might temporarily have multiple, inconsistent settings for this bucket until all sites and Storage Nodes are available. The inconsistent settings can result in unexpected and undesired behavior. For example, if you are placing a bucket under a legal hold and you force a lower consistency level, the bucket's previous compliance settings (that is, legal hold off) might continue to be in effect at some data center sites. As a result, objects that you think are on legal hold might be deleted when their retention period expires, either by the user or by AutoDelete, if enabled.

To force the use of the **strong-site** consistency level, reissue the PUT Bucket compliance request and include the `Consistency-Control` HTTP request header, as follows:

```
PUT /mybucket/?x-ntap-sg-compliance HTTP/1.1
Consistency-Control: strong-site
```

Error responses

- If the bucket was not created to be compliant, the HTTP status code for the response is `404 Not Found`.
- If `RetentionPeriodMinutes` in the request is less than the bucket's current retention period, the HTTP status code is `400 Bad Request`.

Related information

Deprecated: PUT Bucket request modifications for compliance

Bucket and group access policies

Use bucket and group access policies

StorageGRID uses the Amazon Web Services (AWS) policy language to allow S3 tenants to control access to buckets and objects within those buckets. The StorageGRID system implements a subset of the S3 REST API policy language. Access policies for the S3 API are written in JSON.

Access policy overview

There are two kinds of access policies supported by StorageGRID.

- **Bucket policies**, which are configured using the GET Bucket policy, PUT Bucket policy, and DELETE Bucket policy S3 API operations. Bucket policies are attached to buckets, so they are configured to control access by users in the bucket owner account or other accounts to the bucket and the objects in it. A bucket policy applies to only one bucket and possibly multiple groups.

- **Group policies**, which are configured using the Tenant Manager or Tenant Management API. Group policies are attached to a group in the account, so they are configured to allow that group to access specific resources owned by that account. A group policy applies to only one group and possibly multiple buckets.

ⓘ There is no difference in priority between group and bucket policies.

StorageGRID bucket and group policies follow a specific grammar defined by Amazon. Inside each policy is an array of policy statements, and each statement contains the following elements:

- Statement ID (Sid) (optional)
- Effect
- Principal/NotPrincipal
- Resource/NotResource
- Action/NotAction
- Condition (optional)

Policy statements are built using this structure to specify permissions: Grant <Effect> to allow/deny <Principal> to perform <Action> on <Resource> when <Condition> applies.

Each policy element is used for a specific function:

Element	Description
Sid	The Sid element is optional. The Sid is only intended as a description for the user. It is stored but not interpreted by the StorageGRID system.
Effect	Use the Effect element to establish whether the specified operations are allowed or denied. You must identify operations you allow (or deny) on buckets or objects using the supported Action element keywords.
Principal/NotPrincipal	You can allow users, groups, and accounts to access specific resources and perform specific actions. If no S3 signature is included in the request, anonymous access is allowed by specifying the wildcard character (*) as the principal. By default, only the account root has access to resources owned by the account. You only need to specify the Principal element in a bucket policy. For group policies, the group to which the policy is attached is the implicit Principal element.
Resource/NotResource	The Resource element identifies buckets and objects. You can allow or deny permissions to buckets and objects using the Amazon Resource Name (ARN) to identify the resource.

Element	Description
Action/NotAction	The Action and Effect elements are the two components of permissions. When a group requests a resource, they are either granted or denied access to the resource. Access is denied unless you specifically assign permissions, but you can use explicit deny to override a permission granted by another policy.
Condition	The Condition element is optional. Conditions allow you to build expressions to determine when a policy should be applied.

In the Action element, you can use the wildcard character (*) to specify all operations, or a subset of operations. For example, this Action matches permissions such as s3:GetObject, s3:PutObject, and s3:DeleteObject.

```
s3:*Object
```

In the Resource element, you can use the wildcard characters (*) and (?). While the asterisk (*) matches 0 or more characters, the question mark (?) matches any single character.

In the Principal element, wildcard characters aren’t supported except to set anonymous access, which grants permission to everyone. For example, you set the wildcard (*) as the Principal value.

```
"Principal":"*"
```

In the following example, the statement is using the Effect, Principal, Action, and Resource elements. This example shows a complete bucket policy statement that uses the Effect "Allow" to give the Principals, the admin group `federated-group/admin` and the finance group `federated-group/finance`, permissions to perform the Action `s3:ListBucket` on the bucket named `mybucket` and the Action `s3:GetObject` on all objects inside that bucket.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::27233906934684427525:federated-group/admin",
          "arn:aws:iam::27233906934684427525:federated-group/finance"
        ]
      },
      "Action": [
        "s3:ListBucket",
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:iam:s3:::mybucket",
        "arn:aws:iam:s3:::mybucket/*"
      ]
    }
  ]
}
```

The bucket policy has a size limit of 20,480 bytes, and the group policy has a size limit of 5,120 bytes.

Consistency control settings for policies

By default, any updates you make to group policies are eventually consistent. Once a group policy becomes consistent, the changes can take an additional 15 minutes to take effect, because of policy caching. By default, any updates you make to bucket policies are also eventually consistent.

As required, you can change the consistency guarantees for bucket policy updates. For example, you might want a change to a bucket policy to become effective as soon as possible for security reasons.

In this case, you can either set the `Consistency-Control` header in the PUT Bucket policy request, or you can use the PUT Bucket consistency request. When changing the consistency control for this request, you must use the value **all**, which provides the highest guarantee of read-after-write consistency. If you specify any other consistency control value in a header for the PUT Bucket consistency request, the request will be rejected. If you specify any other value for a PUT Bucket policy request, the value will be ignored. Once a bucket policy becomes consistent, the changes can take an additional 8 seconds to take effect, because of policy caching.

> ⓘ If you set the consistency level to **all** to force a new bucket policy to become effective sooner, be sure to set the bucket-level control back to its original value when you are done. Otherwise, all future bucket requests will use the **all** setting.

Use ARN in policy statements

In policy statements, the ARN is used in Principal and Resource elements.

- Use this syntax to specify the S3 resource ARN:

```
arn:aws:s3:::bucket-name
arn:aws:s3:::bucket-name/object_key
```

- Use this syntax to specify the identity resource ARN (users and groups):

```
arn:aws:iam::account_id:root
arn:aws:iam::account_id:user/user_name
arn:aws:iam::account_id:group/group_name
arn:aws:iam::account_id:federated-user/user_name
arn:aws:iam::account_id:federated-group/group_name
```

Other considerations:

- You can use the asterisk (*) as a wildcard to match zero or more characters inside the object key.
- International characters, which can be specified in the object key, should be encoded using JSON UTF-8 or using JSON \u escape sequences. Percent-encoding is not supported.

  RFC 2141 URN Syntax

  The HTTP request body for the PUT Bucket policy operation must be encoded with charset=UTF-8.

Specify resources in a policy

In policy statements, you can use the Resource element to specify the bucket or object for which permissions are allowed or denied.

- Each policy statement requires a Resource element. In a policy, resources are denoted by the element `Resource`, or alternatively, `NotResource` for exclusion.
- You specify resources with an S3 resource ARN. For example:

```
"Resource": "arn:aws:s3:::mybucket/*"
```

- You can also use policy variables inside the object key. For example:

```
"Resource": "arn:aws:s3:::mybucket/home/${aws:username}/*"
```

- The resource value can specify a bucket that does not yet exist when a group policy is created.

Specify principals in a policy

Use the Principal element to identity the user, group, or tenant account that is allowed/denied access to the resource by the policy statement.

- Each policy statement in a bucket policy must include a Principal element. Policy statements in a group policy don’t need the Principal element because the group is understood to be the principal.
- In a policy, principals are denoted by the element “Principal,” or alternatively “NotPrincipal” for exclusion.
- Account-based identities must be specified using an ID or an ARN:

  ```
  "Principal": { "AWS": "account_id"}
  "Principal": { "AWS": "identity_arn" }
  ```

- This example uses the tenant account ID 27233906934684427525, which includes the account root and all users in the account:

  ```
   "Principal": { "AWS": "27233906934684427525" }
  ```

- You can specify just the account root:

  ```
  "Principal": { "AWS": "arn:aws:iam::27233906934684427525:root" }
  ```

- You can specify a specific federated user ("Alex"):

  ```
  "Principal": { "AWS": "arn:aws:iam::27233906934684427525:federated-
  user/Alex" }
  ```

- You can specify a specific federated group ("Managers"):

  ```
  "Principal": { "AWS": "arn:aws:iam::27233906934684427525:federated-
  group/Managers"  }
  ```

- You can specify an anonymous principal:

  ```
  "Principal": "*"
  ```

- To avoid ambiguity, you can use the user UUID instead of the username:

  ```
  arn:aws:iam::27233906934684427525:user-uuid/de305d54-75b4-431b-adb2-
  eb6b9e546013
  ```

  For example, suppose Alex leaves the organization and the username `Alex` is deleted. If a new Alex joins the organization and is assigned the same `Alex` username, the new user might unintentionally inherit the permissions granted to the original user.

- The principal value can specify a group/user name that does not yet exist when a bucket policy is created.

Specify permissions in a policy

In a policy, the Action element is used to allow/deny permissions to a resource. There are a set of permissions that you can specify in a policy, which are denoted by the element "Action," or alternatively, "NotAction" for exclusion. Each of these elements maps to specific S3 REST API operations.

The tables lists the permissions that apply to buckets and the permissions that apply to objects.

ⓘ Amazon S3 now uses the s3:PutReplicationConfiguration permission for both the PUT and DELETE Bucket replication actions. StorageGRID uses separate permissions for each action, which matches the original Amazon S3 specification.

ⓘ A DELETE is performed when a PUT is used to overwrite an existing value.

Permissions that apply to buckets

Permissions	S3 REST API operations	Custom for StorageGRID
s3:CreateBucket	PUT Bucket	
s3:DeleteBucket	DELETE Bucket	
s3:DeleteBucketMetadataNotification	DELETE Bucket metadata notification configuration	Yes
s3:DeleteBucketPolicy	DELETE Bucket policy	
s3:DeleteReplicationConfiguration	DELETE Bucket replication	Yes, separate permissions for PUT and DELETE*
s3:GetBucketAcl	GET Bucket ACL	
s3:GetBucketCompliance	GET Bucket compliance (deprecated)	Yes
s3:GetBucketConsistency	GET Bucket consistency	Yes
s3:GetBucketCORS	GET Bucket cors	
s3:GetEncryptionConfiguration	GET Bucket encryption	
s3:GetBucketLastAccessTime	GET Bucket last access time	Yes
s3:GetBucketLocation	GET Bucket location	

Permissions	S3 REST API operations	Custom for StorageGRID
s3:GetBucketMetadataNotification	GET Bucket metadata notification configuration	Yes
s3:GetBucketNotification	GET Bucket notification	
s3:GetBucketObjectLockConfiguration	GET Object Lock Configuration	
s3:GetBucketPolicy	GET Bucket policy	
s3:GetBucketTagging	GET Bucket tagging	
s3:GetBucketVersioning	GET Bucket versioning	
s3:GetLifecycleConfiguration	GET Bucket lifecycle	
s3:GetReplicationConfiguration	GET Bucket replication	
s3:ListAllMyBuckets	• GET Service • GET Storage Usage	Yes, for GET Storage Usage
s3:ListBucket	• GET Bucket (List Objects) • HEAD Bucket • POST Object restore	
s3:ListBucketMultipartUploads	• List Multipart Uploads • POST Object restore	
s3:ListBucketVersions	GET Bucket versions	
s3:PutBucketCompliance	PUT Bucket compliance (deprecated)	Yes
s3:PutBucketConsistency	PUT Bucket consistency	Yes
s3:PutBucketCORS	• DELETE Bucket cors† • PUT Bucket cors	
s3:PutEncryptionConfiguration	• DELETE Bucket encryption • PUT Bucket encryption	
s3:PutBucketLastAccessTime	PUT Bucket last access time	Yes

Permissions	S3 REST API operations	Custom for StorageGRID
s3:PutBucketMetadataNotification	PUT Bucket metadata notification configuration	Yes
s3:PutBucketNotification	PUT Bucket notification	
s3:PutBucketObjectLockConfiguration	• PUT Bucket with the `x-amz-bucket-object-lock-enabled: true` request header (also requires the s3:CreateBucket permission) • PUT Object Lock Configuration	
s3:PutBucketPolicy	PUT Bucket policy	
s3:PutBucketTagging	• DELETE Bucket tagging† • PUT Bucket tagging	
s3:PutBucketVersioning	PUT Bucket versioning	
s3:PutLifecycleConfiguration	• DELETE Bucket lifecycle† • PUT Bucket lifecycle	
s3:PutReplicationConfiguration	PUT Bucket replication	Yes, separate permissions for PUT and DELETE*

Permissions that apply to objects

Permissions	S3 REST API operations	Custom for StorageGRID
s3:AbortMultipartUpload	• Abort Multipart Upload • POST Object restore	
s3:BypassGovernanceRetention	• DELETE Object • DELETE Multiple Objects • PUT Object retention	
s3:DeleteObject	• DELETE Object • DELETE Multiple Objects • POST Object restore	
s3:DeleteObjectTagging	DELETE Object Tagging	

Permissions	S3 REST API operations	Custom for StorageGRID
s3:DeleteObjectVersionTagging	DELETE Object Tagging (a specific version of the object)	
s3:DeleteObjectVersion	DELETE Object (a specific version of the object)	
s3:GetObject	• GET Object • HEAD Object • POST Object restore • SELECT Object content	
s3:GetObjectAcl	GET Object ACL	
s3:GetObjectLegalHold	GET Object legal hold	
s3:GetObjectRetention	GET Object retention	
s3:GetObjectTagging	GET Object Tagging	
s3:GetObjectVersionTagging	GET Object Tagging (a specific version of the object)	
s3:GetObjectVersion	GET Object (a specific version of the object)	
s3:ListMultipartUploadParts	List Parts, POST Object restore	
s3:PutObject	• PUT Object • PUT Object - Copy • POST Object restore • Initiate Multipart Upload • Complete Multipart Upload • Upload Part • Upload Part - Copy	
s3:PutObjectLegalHold	PUT Object legal hold	
s3:PutObjectRetention	PUT Object retention	
s3:PutObjectTagging	PUT Object Tagging	

Permissions	S3 REST API operations	Custom for StorageGRID
s3:PutObjectVersionTagging	PUT Object Tagging (a specific version of the object)	
s3:PutOverwriteObject	• PUT Object • PUT Object - Copy • PUT Object tagging • DELETE Object tagging • Complete Multipart Upload	Yes
s3:RestoreObject	POST Object restore	

Use PutOverwriteObject permission

The s3:PutOverwriteObject permission is a custom StorageGRID permission that applies to operations that create or update objects. The setting of this permission determines whether the client can overwrite an object's data, user-defined metadata, or S3 object tagging.

Possible settings for this permission include:

- **Allow**: The client can overwrite an object. This is the default setting.
- **Deny**: The client can't overwrite an object. When set to Deny, the PutOverwriteObject permission works as follows:
  - If an existing object is found at the same path:
    - The object's data, user-defined metadata, or S3 object tagging can't be overwritten.
    - Any ingest operations in progress are cancelled, and an error is returned.
    - If S3 versioning is enabled, the Deny setting prevents PUT Object tagging or DELETE Object tagging operations from modifying the TagSet for an object and its noncurrent versions.
  - If an existing object is not found, this permission has no effect.
- When this permission is not present, the effect is the same as if Allow were set.

> If the current S3 policy allows overwrite, and the PutOverwriteObject permission is set to Deny, the client can't overwrite an object's data, user-defined metadata, or object tagging. In addition, if the **Prevent client modification** checkbox is selected (**CONFIGURATION** > **Security settings** > **Network and objects**), that setting overrides the setting of the PutOverwriteObject permission.

Specify conditions in a policy

Conditions define when a policy will be in effect. Conditions consist of operators and key-value pairs.

Conditions use key-value pairs for evaluation. A Condition element can contain multiple conditions, and each condition can contain multiple key-value pairs. The condition block uses the following format:

```
Condition: {
     condition_type: {
          condition_key: condition_values
```

In the following example, the IpAddress condition uses the SourceIp condition key.

```
"Condition": {
    "IpAddress": {
      "aws:SourceIp": "54.240.143.0/24"
        ...
},
        ...
```

Supported condition operators

Condition operators are categorized as follows:

- String
- Numeric
- Boolean
- IP address
- Null check

Condition operators	Description
StringEquals	Compares a key to a string value based on exact matching (case sensitive).
StringNotEquals	Compares a key to a string value based on negated matching (case sensitive).
StringEqualsIgnoreCase	Compares a key to a string value based on exact matching (ignores case).
StringNotEqualsIgnoreCase	Compares a key to a string value based on negated matching (ignores case).
StringLike	Compares a key to a string value based on exact matching (case sensitive). Can include * and ? wildcard characters.
StringNotLike	Compares a key to a string value based on negated matching (case sensitive). Can include * and ? wildcard characters.
NumericEquals	Compares a key to a numeric value based on exact matching.

Condition operators	Description
NumericNotEquals	Compares a key to a numeric value based on negated matching.
NumericGreaterThan	Compares a key to a numeric value based on “greater than” matching.
NumericGreaterThanEquals	Compares a key to a numeric value based on “greater than or equals” matching.
NumericLessThan	Compares a key to a numeric value based on “less than” matching.
NumericLessThanEquals	Compares a key to a numeric value based on “less than or equals” matching.
Bool	Compares a key to a Boolean value based on “true or false” matching.
IpAddress	Compares a key to an IP address or range of IP addresses.
NotIpAddress	Compares a key to an IP address or range of IP addresses based on negated matching.
Null	Checks if a condition key is present in the current request context.

Supported condition keys

Category	Applicable condition keys	Description
IP operators	aws:SourceIp	Will compare to the IP address from which the request was sent. Can be used for bucket or object operations. **Note:** If the S3 request was sent through the Load Balancer service on Admin Nodes and Gateways Nodes, this will compare to the IP address upstream of the Load Balancer service. **Note**: If a third-party, non-transparent load balancer is used, this will compare to the IP address of that load balancer. Any `X-Forwarded-For` header will be ignored because its validity can’t be ascertained.
Resource/Identity	aws:username	Will compare to the sender’s username from which the request was sent. Can be used for bucket or object operations.

Category	Applicable condition keys	Description
s3:ListBucket and s3:ListBucketVersions permissions	s3:delimiter	Will compare to the delimiter parameter specified in a GET Bucket or GET Bucket Object versions request.
s3:ListBucket and s3:ListBucketVersions permissions	s3:max-keys	Will compare to the max-keys parameter specified in a GET Bucket or GET Bucket Object versions request.
s3:ListBucket and s3:ListBucketVersions permissions	s3:prefix	Will compare to the prefix parameter specified in a GET Bucket or GET Bucket Object versions request.
s3:PutObject	s3:object-lock-remaining-retention-days	Compares to the retain-until-date specified in the `x-amz-object-lock-retain-until-date` request header or computed from the bucket default retention period to make sure that these values are within the allowable range for the following requests: • PUT Object • PUT Object - Copy • Initiate Multipart Upload
s3:PutObjectRetention	s3:object-lock-remaining-retention-days	Compares to the retain-until-date specified in the PUT Object Retention request to ensure that it is within the allowable range.

Specify variables in a policy

You can use variables in policies to populate policy information when it is available. You can use policy variables in the `Resource` element and in string comparisons in the `Condition` element.

In this example, the variable `${aws:username}` is part of the Resource element:

```
"Resource": "arn:aws:s3:::bucket-name/home/${aws:username}/*"
```

In this example, the variable `${aws:username}` is part of the condition value in the condition block:

```
"Condition": {
    "StringLike": {
      "s3:prefix": "${aws:username}/*"
        ...
},
        ...
```

Variable	Description
`${aws:SourceIp}`	Uses the SourceIp key as the provided variable.
`${aws:username}`	Uses the username key as the provided variable.
`${s3:prefix}`	Uses the service-specific prefix key as the provided variable.
`${s3:max-keys}`	Uses the service-specific max-keys key as the provided variable.
`${*}`	Special character. Uses the character as a literal * character.
`${?}`	Special character. Uses the character as a literal ? character.
`${$}`	Special character. Uses the character as a literal $ character.

Create policies requiring special handling

Sometimes a policy can grant permissions that are dangerous for security or dangerous for continued operations, such as locking out the root user of the account. The StorageGRID S3 REST API implementation is less restrictive during policy validation than Amazon, but equally strict during policy evaluation.

Policy description	Policy type	Amazon behavior	StorageGRID behavior
Deny self any permissions to the root account	Bucket	Valid and enforced, but root user account retains permission for all S3 bucket policy operations	Same
Deny self any permissions to user/group	Group	Valid and enforced	Same
Allow a foreign account group any permission	Bucket	Invalid principal	Valid, but permissions for all S3 bucket policy operations return a 405 Method Not Allowed error when allowed by a policy

Policy description	Policy type	Amazon behavior	StorageGRID behavior
Allow a foreign account root or user any permission	Bucket	Valid, but permissions for all S3 bucket policy operations return a 405 Method Not Allowed error when allowed by a policy	Same
Allow everyone permissions to all actions	Bucket	Valid, but permissions for all S3 bucket policy operations return a 405 Method Not Allowed error for the foreign account root and users	Same
Deny everyone permissions to all actions	Bucket	Valid and enforced, but root user account retains permission for all S3 bucket policy operations	Same
Principal is a non-existent user or group	Bucket	Invalid principal	Valid
Resource is a non-existent S3 bucket	Group	Valid	Same
Principal is a local group	Bucket	Invalid principal	Valid
Policy grants a non-owner account (including anonymous accounts) permissions to PUT objects	Bucket	Valid. Objects are owned by the creator account, and the bucket policy does not apply. The creator account must grant access permissions for the object using object ACLs.	Valid. Objects are owned by the bucket owner account. Bucket policy applies.

Write-once-read-many (WORM) protection

You can create write-once-read-many (WORM) buckets to protect data, user-defined object metadata, and S3 object tagging. You configure the WORM buckets to allow the creation of new objects and to prevent overwrites or deletion of existing content. Use one of the approaches described here.

To ensure that overwrites are always denied, you can:

- From the Grid Manager, go to **CONFIGURATION** > **Security** > **Security settings** > **Network and objects**, and select the **Prevent client modification** checkbox.
- Apply the following rules and S3 policies:
    - Add a PutOverwriteObject DENY operation to the S3 policy.
    - Add a DeleteObject DENY operation to the S3 policy.
    - Add a PUT Object ALLOW operation to the S3 policy.

Setting DeleteObject to DENY in an S3 policy does not prevent ILM from deleting objects when a rule such as “zero copies after 30 days” exists.

Even when all of these rules and policies are applied, they don’t guard against concurrent writes (see Situation A). They do guard against sequential completed overwrites (see Situation B).

Situation A: Concurrent writes (not guarded against)

```
/mybucket/important.doc
PUT#1 ---> OK
PUT#2 -------> OK
```

Situation B: Sequential completed overwrites (guarded against)

```
/mybucket/important.doc
PUT#1 -------> PUT#2 ---X (denied)
```

Related information

- How StorageGRID ILM rules manage objects
- Example bucket policies
- Example group policies
- Manage objects with ILM
- Use a tenant account

Example bucket policies

Use the examples in this section to build StorageGRID access policies for buckets.

Bucket policies specify the access permissions for the bucket that the policy is attached to. Bucket policies are configured using the S3 PutBucketPolicy API. See Operations on buckets.

A bucket policy can be configured using the AWS CLI as per the following command:

```
> aws s3api put-bucket-policy --bucket examplebucket --policy
file://policy.json
```

Example: Allow everyone read-only access to a bucket

In this example, everyone, including anonymous, is allowed to list objects in the bucket and perform Get Object operations on all objects in the bucket. All other operations will be denied. Note that this policy might not be particularly useful because no one except the account root has permissions to write to the bucket.

```
{
  "Statement": [
    {
      "Sid": "AllowEveryoneReadOnlyAccess",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [ "s3:GetObject", "s3:ListBucket" ],
      "Resource":
["arn:aws:s3:::examplebucket","arn:aws:s3:::examplebucket/*"]
    }
  ]
}
```

Example: Allow everyone in one account full access, and everyone in another account read-only access to a bucket

In this example, everyone in one specified account is allowed full access to a bucket, while everyone in another specified account is only permitted to List the bucket and perform GetObject operations on objects in the bucket beginning with the `shared/` object key prefix.

> ⓘ In StorageGRID, objects created by a non-owner account (including anonymous accounts) are owned by the bucket owner account. The bucket policy applies to these objects.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "95390887230002558202"
      },
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::examplebucket",
        "arn:aws:s3:::examplebucket/*"
      ]
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "31181711887329436680"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::examplebucket/shared/*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "31181711887329436680"
      },
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::examplebucket",
      "Condition": {
        "StringLike": {
          "s3:prefix": "shared/*"
        }
      }
    }
  ]
}
```

Example: Allow everyone read-only access to a bucket and full access by specified group

In this example, everyone including anonymous, is allowed to List the bucket and perform GET Object operations on all objects in the bucket, while only users belonging the group `Marketing` in the specified account are allowed full access.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::95390887230002558202:federated-
group/Marketing"
      },
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::examplebucket",
        "arn:aws:s3:::examplebucket/*"
      ]
    },
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": ["s3:ListBucket","s3:GetObject"],
      "Resource": [
        "arn:aws:s3:::examplebucket",
        "arn:aws:s3:::examplebucket/*"
      ]
    }
  ]
}
```

Example: Allow everyone read and write access to a bucket if client in IP range

In this example, everyone, including anonymous, is allowed to List the bucket and perform any Object operations on all objects in the bucket, provided that the requests come from a specified IP range (54.240.143.0 to 54.240.143.255, except 54.240.143.188). All other operations will be denied, and all requests outside of the IP range will be denied.

```
{
  "Statement": [
    {
      "Sid": "AllowEveryoneReadWriteAccessIfInSourceIpRange",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [ "s3:*Object", "s3:ListBucket" ],
      "Resource":
["arn:aws:s3:::examplebucket","arn:aws:s3:::examplebucket/*"],
      "Condition": {
        "IpAddress": {"aws:SourceIp": "54.240.143.0/24"},
        "NotIpAddress": {"aws:SourceIp": "54.240.143.188"}
      }
    }
  ]
}
```

Example: Allow full access to a bucket exclusively by a specified federated user

In this example, the federated user Alex is allowed full access to the `examplebucket` bucket and its objects. All other users, including ‘root’, are explicitly denied all operations. Note however that ‘root’ is never denied permissions to Put/Get/DeleteBucketPolicy.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::95390887230002558202:federated-user/Alex"
      },
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket",
        "arn:aws:s3:::examplebucket/*"
      ]
    },
    {
      "Effect": "Deny",
      "NotPrincipal": {
        "AWS": "arn:aws:iam::95390887230002558202:federated-user/Alex"
      },
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket",
        "arn:aws:s3:::examplebucket/*"
      ]
    }
  ]
}
```

Example: PutOverwriteObject permission

In this example, the `Deny` Effect for PutOverwriteObject and DeleteObject ensures that no one can overwrite or delete the object’s data, user-defined metadata, and S3 object tagging.

```
{
  "Statement": [
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": [
        "s3:PutOverwriteObject",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion"
      ],
      "Resource": "arn:aws:s3:::wormbucket/*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::95390887230002558202:federated-
group/SomeGroup"

},
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::wormbucket"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::95390887230002558202:federated-
group/SomeGroup"

},
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::wormbucket/*"
    }
  ]
}
```

Example group policies

Use the examples in this section to build StorageGRID access policies for groups.

Group policies specify the access permissions for the group that the policy is attached to. There is no `Principal` element in the policy because it is implicit. Group policies are configured using the Tenant Manager or the API.

Example: Set group policy using Tenant Manager

When you add or edit a group in the Tenant Manager, you can select a group policy to determine which S3 access permissions the members of this group will have. See Create groups for an S3 tenant.

- **No S3 Access**: Default option. Users in this group don't have access to S3 resources, unless access is granted with a bucket policy. If you select this option, only the root user will have access to S3 resources by default.
- **Read Only Access**: Users in this group have read-only access to S3 resources. For example, users in this group can list objects and read object data, metadata, and tags. When you select this option, the JSON string for a read-only group policy appears in the text box. You can't edit this string.
- **Full Access**: Users in this group have full access to S3 resources, including buckets. When you select this option, the JSON string for a full-access group policy appears in the text box. You can't edit this string.
- **Ransomware Mitigation**: This sample policy applies to all buckets for this tenant. Users in this group can perform common actions, but can't permanently delete objects from buckets that have object versioning enabled.

  Tenant Manager users who have the Manage all buckets permission can override this group policy. Limit the Manage all buckets permission to trusted users, and use Multi-Factor Authentication (MFA) where available.
- **Custom**: Users in the group are granted the permissions you specify in the text box.

Example: Allow group full access to all buckets

In this example, all members of the group are permitted full access to all buckets owned by the tenant account unless explicitly denied by bucket policy.

```
{
  "Statement": [
    {
      "Action": "s3:*",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::*"
    }
  ]
}
```

Example: Allow group read-only access to all buckets

In this example, all members of the group have read-only access to S3 resources, unless explicitly denied by the bucket policy. For example, users in this group can list objects and read object data, metadata, and tags.

```
{
  "Statement": [
    {
      "Sid": "AllowGroupReadOnlyAccess",
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:ListBucket",
        "s3:ListBucketVersions",
        "s3:GetObject",
        "s3:GetObjectTagging",
        "s3:GetObjectVersion",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::*"
    }
  ]
}
```

Example: Allow group members full access to only their “folder” in a bucket

In this example, members of the group are only permitted to list and access their specific folder (key prefix) in the specified bucket. Note that access permissions from other group policies and the bucket policy should be considered when determining the privacy of these folders.

```
{
  "Statement": [
    {
      "Sid": "AllowListBucketOfASpecificUserPrefix",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::department-bucket",
      "Condition": {
        "StringLike": {
          "s3:prefix": "${aws:username}/*"
        }
      }
    },
    {
      "Sid": "AllowUserSpecificActionsOnlyInTheSpecificUserPrefix",
      "Effect": "Allow",
      "Action": "s3:*Object",
      "Resource": "arn:aws:s3:::department-bucket/${aws:username}/*"
    }
  ]
}
```

Configure security for the REST API

You should review the security measures implemented for the REST API and understand how to secure your system.

How StorageGRID provides security for the REST API

You should understand how the StorageGRID system implements security, authentication, and authorization for the REST API.

StorageGRID uses the following security measures.

- Client communications with the Load Balancer service use HTTPS if HTTPS is configured for the load balancer endpoint.

  When you configure a load balancer endpoint, HTTP can optionally be enabled. For example, you might want to use HTTP for testing or other non-production purposes. See the instructions for administering StorageGRID for more information.

- By default, StorageGRID uses HTTPS for client communications with Storage Nodes.

  HTTP can optionally be enabled for these connections. For example, you might want to use HTTP for testing or other non-production purposes. See the instructions for administering StorageGRID for more information.

- Communications between StorageGRID and the client are encrypted using TLS.

- Communications between the Load Balancer service and Storage Nodes within the grid are encrypted whether the load balancer endpoint is configured to accept HTTP or HTTPS connections.
- Clients must supply HTTP authentication headers to StorageGRID to perform REST API operations.

Security certificates and client applications

Clients can connect to the Load Balancer service on Gateway Nodes or Admin Nodes, directly to Storage Nodes.

In all cases, client applications can make TLS connections using either a custom server certificate uploaded by the grid administrator or a certificate generated by the StorageGRID system:

- When client applications connect to the Load Balancer service, they do so using the certificate that was configured for the specific load balancer endpoint used to make the connection. Each endpoint has its own certificate, which is either a custom server certificate uploaded by the grid administrator or a certificate that the grid administrator generated in StorageGRID when configuring the endpoint.
- When client applications connect directly to a Storage Node, they use either the system-generated server certificates that were generated for Storage Nodes when the StorageGRID system was installed (which are signed by the system certificate authority), or a single custom server certificate that is supplied for the grid by a grid administrator.

Clients should be configured to trust the certificate authority that signed whichever certificate they use to establish TLS connections.

See the instructions for administering StorageGRID for information about configuring load balancer endpoints, and for instructions on adding a single custom server certificate for TLS connections directly to Storage Nodes.

Summary

The following table shows how security issues are implemented in the S3 and Swift REST APIs:

Security issue	Implementation for REST API
Connection security	TLS
Server authentication	X.509 server certificate signed by system CA or custom server certificate supplied by administrator
Client authentication	• S3: S3 account (access key ID and secret access key) • Swift: Swift account (user name and password)
Client authorization	• S3: Bucket ownership and all applicable access control policies • Swift: Administrator role access

Related information

Administer StorageGRID

Supported hashing and encryption algorithms for TLS libraries

The StorageGRID system supports a limited set of cipher suites that client applications can use when establishing a Transport Layer Security (TLS) session. To configure ciphers, go to **CONFIGURATION** > **Security** > **Security settings** and select **TLS and SSH policies**.

Supported versions of TLS

StorageGRID supports TLS 1.2 and TLS 1.3.

> SSLv3 and TLS 1.1 (or earlier versions) are no longer supported.

Related information

Configure tenant accounts and connections

Monitor and audit operations

Monitor object ingest and retrieval rates

You can monitor object ingest and retrieval rates as well as metrics for object counts, queries, and verification. You can view the number of successful and failed attempts by client applications to read, write, and modify objects in the StorageGRID system.

Steps

1. Sign in to the Grid Manager using a supported web browser.
2. On the dashboard, select **Performance** > **S3 operations** or **Performance** > **Swift operations**.

   This section summarizes the number of client operations performed by your StorageGRID system. Protocol rates are averaged over the last two minutes.

3. Select **NODES**.
4. From the Nodes home page (deployment level), click the **Load Balancer** tab.

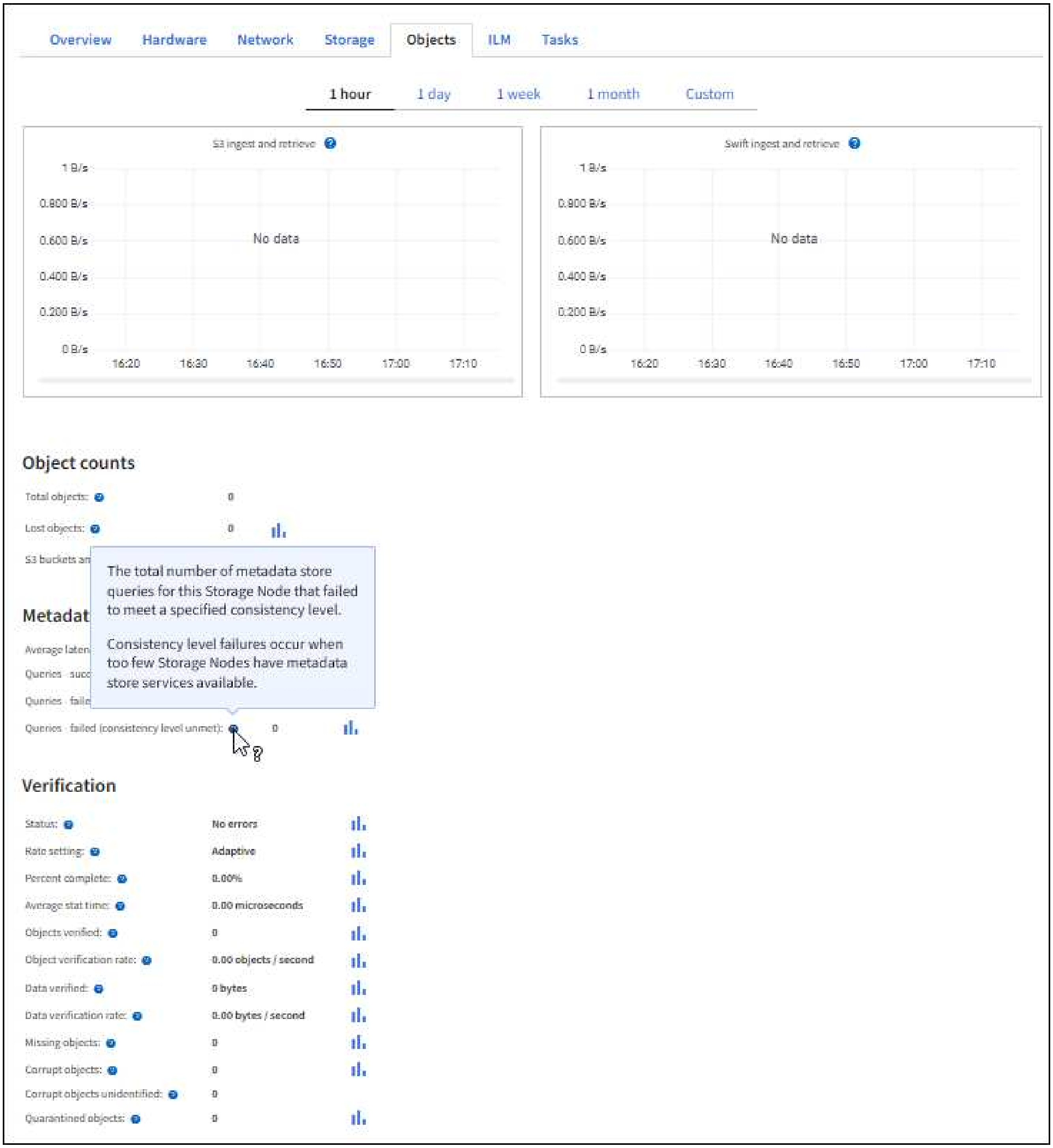
   The charts show trends for all client traffic directed to load balancer endpoints within the grid. You can select a time interval in hours, days, weeks, months, or years, or you can apply a custom interval.

5. From the Nodes home page (deployment level), click the **Objects** tab.

   The chart shows ingest and retrieve rates for your entire StorageGRID system in bytes per second and total bytes. You can select a time interval in hours, days, weeks, months, or years, or you can apply a custom interval.

6. To see information for a particular Storage Node, select the node from the list on the left, and click the **Objects** tab.

   The chart shows the object ingest and retrieval rates for this Storage Node. The tab also includes metrics for object counts, queries, and verification. You can click the labels to see the definitions of these metrics.



Object counts

Total objects: 0

Lost objects: 0

S3 buckets an

Metadat

Average laten

Queries - succ

Queries - faile

Queries - failed (consistency level unmet): 0

The total number of metadata store queries for this Storage Node that failed to meet a specified consistency level.

Consistency level failures occur when too few Storage Nodes have metadata store services available.

Verification

Status:	No errors
Rate setting:	Adaptive
Percent complete:	0.00%
Average stat time:	0.00 microseconds
Objects verified:	0
Object verification rate:	0.00 objects / second
Data verified:	0 bytes
Data verification rate:	0.00 bytes / second
Missing objects:	0
Corrupt objects:	0
Corrupt objects unidentified:	0
Quarantined objects:	0

7. If you want even more detail:
   a. Select **SUPPORT** > **Tools** > **Grid topology**.
   b. Select ***site*** > **Overview** > **Main**.

      The API Operations section displays summary information for the entire grid.

   c. Select ***Storage Node*** > **LDR** > ***client application*** > **Overview** > **Main**

      The Operations section displays summary information for the selected Storage Node.

Access and review audit logs

Audit messages are generated by StorageGRID services and stored in text log files. API-specific audit messages in the audit logs provide critical security, operation, and performance monitoring data that can help you evaluate the health of your system.

Before you begin

- You have specific access permissions.
- You have the `Passwords.txt` file.
- You know the IP address of an Admin Node.

About this task

The active audit log file is named `audit.log`, and it is stored on Admin Nodes.

Once a day, the active audit.log file is saved, and a new `audit.log` file is started. The name of the saved file indicates when it was saved, in the format *`yyyy-mm-dd.txt`*.

After a day, the saved file is compressed and renamed, in the format *`yyyy-mm-dd.txt.gz`*, which preserves the original date.

This example shows the active `audit.log` file, the previous day's file (`2018-04-15.txt`), and the compressed file for the prior day (`2018-04-14.txt.gz`).

```
audit.log
2018-04-15.txt
2018-04-14.txt.gz
```

Steps

1. Log in to an Admin Node:
   a. Enter the following command: `ssh admin@`*`primary_Admin_Node_IP`*
   b. Enter the password listed in the `Passwords.txt` file.
   c. Enter the following command to switch to root: `su -`
   d. Enter the password listed in the `Passwords.txt` file.

      When you are logged in as root, the prompt changes from `$` to `#`.
2. Go to the directory containing the audit log files:

   ```
   cd /var/local/audit/export
   ```
3. View the current or a saved audit log file, as required.

S3 operations tracked in the audit logs

Several bucket operations and object operations are tracked in the StorageGRID audit logs.

Bucket operations tracked in the audit logs

- DELETE Bucket
- DELETE Bucket tagging
- DELETE Multiple Objects
- GET Bucket (List Objects)
- GET Bucket Object versions
- GET Bucket tagging
- HEAD Bucket
- PUT Bucket
- PUT Bucket compliance
- PUT Bucket tagging
- PUT Bucket versioning

Object operations tracked in the audit logs

- Complete Multipart Upload
- Upload Part (when the ILM rule uses the Balanced or Strict ingest behaviors)
- Upload Part - Copy (when the ILM rule uses the Balanced or Strict ingest behaviors)
- DELETE Object
- GET Object
- HEAD Object
- POST Object restore
- PUT Object
- PUT Object - Copy

Related information

Operations on buckets

Operations on objects

Benefits of active, idle, and concurrent HTTP connections

How you configure HTTP connections can impact the performance of the StorageGRID system. Configurations differ depending on whether the HTTP connection is active or idle or you have concurrent multiple connections.

You can identify the performance benefits for the following types of HTTP connections:

- Idle HTTP connections
- Active HTTP connections
- Concurrent HTTP connections

Benefits of keeping idle HTTP connections open

You should keep HTTP connections open even when client applications are idle to allow client applications to perform subsequent transactions over the open connection. Based on system measurements and integration experience, you should keep an idle HTTP connection open for a maximum of 10 minutes. StorageGRID might automatically close an HTTP connection that is kept open and idle for longer than 10 minutes.

Open and idle HTTP connections provide the following benefits:

- Reduced latency from the time that the StorageGRID system determines it has to perform an HTTP transaction to the time that the StorageGRID system can perform the transaction

  Reduced latency is the main advantage, especially for the amount of time required to establish TCP/IP and TLS connections.

- Increased data transfer rate by priming the TCP/IP slow-start algorithm with previously performed transfers
- Instantaneous notification of several classes of fault conditions that interrupt connectivity between the client application and the StorageGRID system

Determining how long to keep an idle connection open is a trade-off between the benefits of slow start that is associated with the existing connection and the ideal allocation of the connection to internal system resources.

Benefits of active HTTP connections

For connections directly to Storage Nodes, you should limit the duration of an active HTTP connection to a maximum of 10 minutes, even if the HTTP connection continuously performs transactions.

Determining the maximum duration that a connection should be held open is a trade-off between the benefits of connection persistence and the ideal allocation of the connection to internal system resources.

For client connections to Storage Nodes, limiting active HTTP connections provides the following benefits:

- Enables optimal load balancing across the StorageGRID system.

  Over time, an HTTP connection might no longer be optimal as load balancing requirements change. The system performs its best load balancing when client applications establish a separate HTTP connection for each transaction, but this negates the much more valuable gains associated with persistent connections.

- Allows client applications to direct HTTP transactions to LDR services that have available space.
- Allows maintenance procedures to start.

  Some maintenance procedures start only after all the in-progress HTTP connections are complete.

For client connections to the Load Balancer service, limiting the duration of open connections can be useful for allowing some maintenance procedures to start promptly. If the duration of client connections is not limited, it might take several minutes for active connections to be automatically terminated.

Benefits of concurrent HTTP connections

You should keep multiple TCP/IP connections to the StorageGRID system open to allow parallelism, which increases performance. The optimal number of parallel connections depends on a variety of factors.

Concurrent HTTP connections provide the following benefits:

- Reduced latency

  Transactions can start immediately instead of waiting for other transactions to be completed.

- Increased throughput

  The StorageGRID system can perform parallel transactions and increase aggregate transaction throughput.

Client applications should establish multiple HTTP connections. When a client application has to perform a transaction, it can select and immediately use any established connection that is not currently processing a transaction.

Each StorageGRID system’s topology has different peak throughput for concurrent transactions and connections before performance begins to degrade. Peak throughput depends on factors such as computing resources, network resources, storage resources, and WAN links. The number of servers and services and the number of applications that the StorageGRID system supports are also factors.

StorageGRID systems often support multiple client applications. You should keep this in mind when you determine the maximum number of concurrent connections used by a client application. If the client application consists of multiple software entities that each establish connections to the StorageGRID system, you should add up all the connections across the entities. You might have to adjust the maximum number of concurrent connections in the following situations:

- The StorageGRID system’s topology affects the maximum number of concurrent transactions and connections that the system can support.
- Client applications that interact with the StorageGRID system over a network with limited bandwidth might have to reduce the degree of concurrency to ensure that individual transactions are completed in a reasonable time.
- When many client applications share the StorageGRID system, you might have to reduce the degree of concurrency to avoid exceeding the limits of the system.

Separation of HTTP connection pools for read and write operations

You can use separate pools of HTTP connections for read and write operations and control how much of a pool to use for each. Separate pools of HTTP connections enable you to better control transactions and balance loads.

Client applications can create loads that are retrieve-dominant (read) or store-dominant (write). With separate pools of HTTP connections for read and write transactions, you can adjust how much of each pool to dedicate for read or write transactions.

Copyright information

Trademark information