StorageGRID® 11.2

S3 (Simple Storage Service) Implementation Guide

August 2019 | 215-13585_2019-08_en-us
doccomments@netapp.com

NetApp®

Contents

Support for the S3 REST API

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-premises object storage that uses the StorageGRID system. This requires minimal changes to a client application's current use of S3 REST API calls.

Changes to S3 REST API support

You should be aware of changes to the StorageGRID system's support for the S3 REST API.

Release	Comments
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.
11.1	Added support for Cross-Origin Resource Sharing (CORS), HTTP for S3 client connections to grid nodes, and compliance settings on buckets.
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.

Supported versions

StorageGRID supports the following specific versions of S3 and HTTP.

Item	Version
S3 specification	*Simple Storage Service API Reference* 2006-03-01

Item	Version
HTTP	1.1 For more information about HTTP, see HTTP/1.1 (RFC 2616). **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

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
Notifications	Sends notifications about events in a source StorageGRID bucket to a configured Simple Notification Service (SNS) endpoint.	PUT Bucket notification
Search integration	Sends object metadata for objects stored in a StorageGRID bucket to a configured Elasticsearch index.	PUT Bucket metadata notification **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. 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.

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 theStorageGRID 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.

Related concepts

Operations on buckets on page 20

Related references

PUT Bucket metadata notification configuration request on page 51

Related information

Using tenant accounts
Administering StorageGRID

Configuring tenant accounts and connections

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

Steps

1. Creating and configuring S3 tenant accounts on page 8
2. Identifying IP addresses for API Gateway Nodes and Storage Nodes on page 9
3. Deciding to use HTTPS or HTTP connections on page 9
4. Identifying port numbers for API Gateway Nodes and Storage Nodes on page 10
5. Endpoint domain names for S3 requests on page 10
6. Testing your S3 REST API configuration on page 11

Creating and configuring 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 and users, and containers and objects.

S3 tenant accounts are created by a StorageGRID grid administrator using the Grid Manager or the Grid Management API. When creating an S3 tenant account, the grid administrator specifies the following information:

- Display name for the tenant (the tenant's account ID is assigned automatically and cannot be changed)
- Whether the use of platform services will be allowed for the account
- Optionally, a storage quota for the tenant account—the maximum number of gigabytes, terabytes, or petabytes available for the tenant's objects. A tenant's storage quota represents a logical amount (object size), not a physical amount (size on disk).
- If single sign-on (SSO) is not in use for the StorageGRID system, whether the tenant account will use its own identity source or share the grid's identity source, and the initial password for the tenant's local root user.
- If SSO is enabled, which federated group has Root Access permission to configure the tenant account.

After an S3 tenant account is created, tenant users can access the Tenant Manager to perform tasks such as the following:

- Setting up identity federation (unless the identity source is shared with the grid), and creating local groups and users
- Managing S3 access keys
- Creating and managing S3 buckets
- Using platform services (if enabled)
- Monitoring storage usage

Attention: S3 tenant users can create and manage S3 buckets with the Tenant Manager, but they must have S3 access keys and use the S3 REST API to ingest and manage objects.

Related information

Administering StorageGRID
Using tenant accounts

Identifying IP addresses for API Gateway Nodes and Storage Nodes

Client applications use the IP address of a Storage Node or API Gateway Node to connect to StorageGRID.

About this task

Client applications connect directly to StorageGRID Storage Nodes or API Gateway Nodes to store and retrieve objects. API Gateway Nodes allow ingests to be load balanced across Storage Nodes.

You can use the Grid Manager to look up the IP address of a specific Storage Node or API Gateway Node.

Steps

1. Sign in to the Grid Manager using a supported browser.
2. Select **Nodes**.
3. Select the Storage Node or API Gateway Node to which you want to connect.
4. Select the **Overview** tab.
5. In the Node Information section, note the IP addresses for the node.
6. Click **Show more** to view IPv6 addresses and interface mappings.

   You can establish connections from client applications to any of the IP addresses in the list:

   - **eth0:** Grid Network
   - **eth1:** Admin Network (optional)
   - **eth2:** Client Network (optional)

   Note: IPv6 is only supported for client application connections through the API Gateway Node.

Related information

Administering StorageGRID

Deciding to use HTTPS or HTTP connections

Client applications can use encrypted HTTPS connections or less-secure HTTP connections when they connect to the Storage Nodes or API Gateway Nodes.

By default, client applications use HTTPS for all connections with Storage Nodes and API Gateway Nodes. Optionally, the **HTTP** option can be enabled from the Grid Manager to also allow client applications to use HTTP. For example, a client application might use HTTP when testing the connection to a Storage Node in a non-production environment.

Attention: HTTP mode is intended for use in testing and debugging environments. Be careful when enabling HTTP for a production grid since requests will be sent unencrypted.

To learn how to enable this option, see information about administering StorageGRID. If the **HTTP** option has been enabled for the grid, you must use different ports for HTTP communications than for HTTPS communications.

Related concepts

Identifying port numbers for API Gateway Nodes and Storage Nodes on page 10

Benefits of active, idle, and concurrent HTTP connections on page 80

Related information

Administering StorageGRID

Identifying port numbers for API Gateway Nodes and Storage Nodes

You must know which port number to use when connecting the S3 client to the Storage Node or API Gateway Node.

The following ports are used by S3 clients to connect to the StorageGRID system. You use different ports to connect to Storage Nodes than to connect to API Gateway Nodes. The port number also depends on whether the **HTTP** grid option has been enabled and you are using an HTTP connection.

Grid node	Use	Port number
API Gateway Node	S3 port for HTTPS	8082
	S3 port for HTTP	8084
Storage Node	S3 port for HTTPS	18082
	S3 port for HTTP	18084

Related concepts

Deciding to use HTTPS or HTTP connections on page 9

Related information

Administering StorageGRID

Endpoint domain names for S3 requests

A StorageGRID administrator must provide you with the endpoint domain names to use for S3 path-style and S3 virtual hosted-style requests. If the StorageGRID system uses an uncommon root CA, your grid administrator will also provide the root CA certificate to be imported into your S3 client application's truststore.

About this task

A grid administrator must configure S3 endpoints using the Grid Manager. See the instructions for administering StorageGRID for complete instructions.

For example, if the endpoint is `s3.company.com`, the grid administrator will obtain a custom server certificate that includes the `s3.company.com` endpoint and the endpoint's wildcard SAN:

`*.s3.company.com`. The administrator will also confirm that the DNS server supports the endpoint and the wildcard SAN.

Once the endpoint has been configured, you can use virtual hosted-style requests (for example, `bucket.s3.company.com`). The DNS server will resolve to the correct endpoint and the certificate will authenticate the endpoint as expected.

Related concepts

Configuring security for the REST API on page 74

Related information

Administering StorageGRID

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

Steps

1. Configure the Amazon Web Services settings to use the account you created in the StorageGRID system:
   a. Enter configuration mode:
      ```
      aws configure
      ```
   b. Enter the AWS Access Key ID for the account you created.
   c. Enter the AWS 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.
   ```
   aws s3api --endpoint-url https://10.96.101.17:8082
   --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://10.96.101.17:8082 --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://10.96.101.17:8082 --no-verify-ssl
list-objects --bucket testbucket
```

5. Delete the object.

```
aws s3api --endpoint-url https://10.96.101.17:8082 --no-verify-ssl
delete-object --bucket testbucket --key s3.pdf
```

6. Delete the bucket.

```
aws s3api --endpoint-url https://10.96.101.17:8082 --no-verify-ssl
delete-bucket --bucket testbucket
```

How StorageGRID implements the S3 REST API

A client application can use S3 REST API calls to connect to Storage Nodes and API Gateway Nodes, to create buckets, and to store and retrieve objects.

Conflicting client requests

Conflicting client requests, such as a 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 let you make a trade-off 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

You can set the consistency control for a bucket or an API operation to 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. Matches AWS S3 consistency guarantees. **Note:** If your application uses HEAD requests on objects that do not exist, you might receive a high number of 500 Internal Server errors if one or more Storage Nodes are unavailable. To prevent these errors, set the consistency control to “available” unless you require consistency guarantees similar to AWS S3.
available (eventual consistency for HEAD operations)	Behaves the same as the “read-after-new-write” consistency level, but only provides eventual consistency for HEAD operations. Offers higher availability for HEAD operations than “read-after-new-write” if Storage Nodes are unavailable. Differs from AWS S3 consistency guarantees for HEAD operations only.

Consistency control	Description
weak	Provides eventual consistency and high availability, with minimal data protection guarantees, especially if a Storage Node fails or is unavailable. Suitable only for write-heavy workloads that require high availability, do not require read-after-write consistency, and can tolerate the potential loss of data if a node fails.

Using the “read-after-new-write” and “available” consistency controls

When a HEAD or GET operation uses the “read-after-new-write” consistency control or a GET operation uses the “available” 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 the highest consistency level, “all,” which requires all copies of the object metadata to be available.

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 the “all” consistency level. Because this consistency level requires all copies of the object metadata to be available, you can receive a high number of 500 Internal Server errors if one or more Storage Nodes are unavailable.

Unless you require consistency guarantees similar to AWS S3, you can prevent these errors for HEAD operations by setting the consistency control to “available.” When a HEAD operation uses the “available” consistency control, StorageGRID provides eventual consistency only. It does not retry a failed operation until it reaches the “all” consistency level, so it does not require that all copies of the object metadata be available.

Specifying the consistency control for an 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
```

Note: You must use the same consistency control for both the PUT Object and GET Object operations. For example, using “weak” to write an object and then using “strong-global” to read the same object back does not provide strong consistency across all sites.

Specifying the consistency control for a 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 are not 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.

Related references

GET Bucket consistency request on page 45
PUT Bucket consistency request on page 46

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 within the StorageGRID system.

Storage grade

The type of storage used to store object data (disk or archival media).

Loss protection

How copies are made: 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.

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

  Note: 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. You can enable last access time updates using the PUT Bucket last access time request, the **S3 > Buckets > Configure Last Access Time** check box in the Tenant Manager, or using 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 references

PUT Bucket last access time request on page 47

Related information

Using tenant accounts
Administering StorageGRID

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.

In StorageGRID, object versioning can be combined with Information Lifecycle Management (ILM), rather than Amazon's Object Lifecycle Management, which is not supported. 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.

Note: 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 creation of ILM rules that use a noncurrent reference time. 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.

Recommendations for implementing the 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 do not 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

You should not use random values as the first four characters of object keys. This is in contrast to AWS recommendations for key prefixes. Instead, you should use non-random, non-unique prefixes, such as `image`.

If you do follow the AWS recommendation to use random and unique characters in key prefixes, you should 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
```

Recommendations for “range reads”

If the **Stored Object Compression** grid option is enabled for StorageGRID, 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 very 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.

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

Related concepts

Consistency controls on page 13

Related references

PUT Bucket consistency request on page 46

Related information

Administering StorageGRID

S3 REST API supported operations and limitations

The StorageGRID system implements the *Simple Storage Service API Reference* (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.

Related information

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

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 common request headers defined by the *Simple Storage Service API Reference*, with several exceptions.

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.
x-amz-copy-source-server-side-encryption-customer-key	Not implemented.
x-amz-copy-source-server-side-encryption-customer-key-MD5	Not implemented.
x-amz-security-token	Not implemented. Returns `XNotImplemented`.
x-amz-server-side-encryption-customer-algorithm	Not implemented.

Related information

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

Common response headers

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

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

Related information

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

Authenticating 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.

Using 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.

Using 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 do not need to know the secret access key in order 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	Implemented with all Amazon S3 REST API behavior.
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.

Operation	Implementation
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 references

GET Storage Usage request on page 55

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 in order to support S3 virtual hosted-style requests.

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	Implemented with all Amazon S3 REST API behavior.
DELETE Bucket cors	This operation deletes the CORS configuration for the bucket.
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.
GET Bucket (List Objects), version 1 and version 2	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.
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 location	This operation returns the bucket's region. By default, `us-east-1` is returned unless a region was set using the `LocationConstraint` element in the PUT Bucket request.

Operation	Implementation
GET Bucket Object versions	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 notification	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 versioning	This implementation uses the `versioning` subresource to return the versioning state of a bucket. The versioning state returned indicates if the bucket is “Unversioned” or if the bucket is version “Enabled” or “Suspended.”
HEAD Bucket	This operation determines if a bucket exists and you have permission to access it.
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 between 3 and 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 because 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 do not 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. • When using a PUT Bucket request, you can include the `SGCompliance` XML element in the XML request body. `SGCompliance` is a custom StorageGRID element that allows you to create buckets that are compliant.

Operation	Implementation
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`.
PUT Bucket notification	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 are not 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`. • Notifications on the `s3:ReducedRedundancyLostObject` event are not supported. • Event notification messages use standard values for most keys, except for the following: ◦ `eventSource` returns `sgws:s3` ◦ `awsRegion`: this key is not returned ◦ `x-amz-id-2`: this key is not returned ◦ `arn` returns `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: • 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. 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 do not 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 are not notified. ◦ If you delete the object or bucket after it has been replicated, StorageGRID follows standard Amazon S3 delete behavior for cross-region replication.
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`.

Related concepts

Consistency controls on page 13
Bucket and group access policies on page 56
S3 operations tracked in the audit logs on page 78

Related references

GET Bucket last access time request on page 47
PUT Bucket request modifications for compliance on page 40

Related information

Amazon Web Services (AWS) Documentation: Cross-Region Replication
Using tenant accounts

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
PUT Bucket modifications for compliance	Creates a compliant bucket and sets the compliance settings for that bucket.	*PUT Bucket request modifications for compliance* on page 40
GET Bucket compliance	Returns the compliance settings currently in effect for a bucket that was created to be compliant.	*GET Bucket compliance request* on page 41
PUT Bucket compliance	Allows you to modify the compliance settings for an existing bucket that was created to be compliant.	*PUT Bucket compliance request* on page 43
GET Bucket consistency	Returns the consistency level being applied to a particular bucket.	*GET Bucket consistency request* on page 45
PUT Bucket consistency	Sets the consistency level applied to a particular bucket.	*PUT Bucket consistency request* on page 46
GET Bucket last access time	Returns whether last access time updates are enabled or disabled for a particular bucket.	*GET Bucket last access time request* on page 47
PUT Bucket last access time	Allows you to enable or disable last access time updates for a particular bucket.	*PUT Bucket last access time request* on page 47
DELETE Bucket metadata notification configuration	Deletes the metadata notification configuration XML associated with a particular bucket.	*DELETE Bucket metadata notification configuration request* on page 48
GET Bucket metadata notification configuration	Returns the metadata notification configuration XML associated with a particular bucket.	*GET Bucket metadata notification configuration request* on page 49

Operation	Description	For more information
PUT Bucket metadata notification configuration	Configures the metadata notification service for a bucket.	*PUT Bucket metadata notification configuration request* on page 51

Related concepts

S3 operations tracked in the audit logs on page 78

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:

- All of the operations on objects, except GET Object ACL and `OPTIONS /`, support StorageGRID consistency controls.
- Conflicting client requests, such as two clients writing to the same key, are resolved on a “latest-wins” basis. The timing for the “latest-wins”evaluation is based on when the StorageGRID system completes a given request, and not on when S3 clients begin an operation.
- All objects in a StorageGRID bucket are owned by the bucket owner, including objects created by an anonymous user, or by another account.
- Data objects ingested to the StorageGRID system through Swift cannot be accessed through S3.

Operation	Implementation
DELETE Object	Multi-Factor Authentication (MFA) and the response header `x-amz-mfa` are not supported. **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`.

Operation	Implementation
DELETE Multiple Objects	Multi-Factor Authentication (MFA) and the response header `x-amz-mfa` are not supported. Multiple objects can be deleted in the same request message. **Note:** The DELETE Multiple Objects request is not supported on versioned buckets.
DELETE Object tagging	Uses the `tagging` subresource to remove all tags from an object. Implemented with all Amazon S3 REST API behavior. **Versioning** If the `versionId` query parameter is not specified in the request, the operation deletes all tags from the most recent version of the object in a versioned bucket. If the current version of the object is a delete marker, a "MethodNotAllowed" status is returned with the `x-amz-delete-marker` response header set to `true`.
GET Object	*GET Object* on page 27
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 tagging	Uses the `tagging` subresource to return all tags for an object. Implemented with all Amazon S3 REST API behavior **Versioning** If the `versionId` query parameter is not specified in the request, the operation returns all tags from the most recent version of the object in a versioned bucket. If the current version of the object is a delete marker, a "MethodNotAllowed" status is returned with the `x-amz-delete-marker` response header set to `true`.
HEAD Object	*HEAD Object* on page 28
POST Object restore	*POST Object restore* on page 29
PUT Object	*PUT Object* on page 30
PUT Object - Copy	*PUT Object - Copy* on page 32
PUT Object tagging	Uses the `tagging` subresource to add a set of tags to an existing object. Implemented with all Amazon S3 REST API behavior **Resolving conflicts** Conflicting client requests, such as two clients writing to the same key, are resolved on a "latest-wins" basis. The timing for the "latest-wins"evaluation is based on when the StorageGRID system completes a given request, and not on when S3 clients begin an operation. **Versioning** If the `versionId` query parameter is not specified in the request, the operation add tags to the most recent version of the object in a versioned bucket. If the current version of the object is a delete marker, a "MethodNotAllowed" status is returned with the `x-amz-delete-marker` response header set to `true`.

Related concepts

Consistency controls on page 13
S3 operations tracked in the audit logs on page 78

GET Object

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

Unsupported request header

The following request header is not supported and returns `XNotImplemented`:

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

Versioning

If a `versionId` subresource is not specified, the operation fetches the most recent version of the object in a versioned bucket. If the current version of the object is a delete marker, a "Not Found" status is returned with the `x-amz-delete-marker` response header set to `true`.

Behavior of GET Object for Cloud Storage Pool objects

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

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 Glacier	`200 OK` A copy of the object is retrieved from the Cloud Storage Pool.
Object transitioned to Glacier from the Cloud Storage Pool bucket	`403 Forbidden`, `InvalidObjectState` Use a POST Object restore request to restore a temporary copy of the object back to the Cloud Storage Pool.
Object in process of being restored from Glacier to the Cloud Storage Pool bucket	`403 Forbidden`, `InvalidObjectState` Wait for the POST Object restore request to complete.
Object fully restored to the Cloud Storage Pool bucket	`200 OK` The response body includes a copy of the restored object.

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 bucket 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 Glacier or when some parts of the object have not yet been restored.

In these cases:

- The GET Object request might return some data but stop midway through the transfer.

- A subsequent GET Object request might return `403 Forbidden`.

Related concepts

HEAD Object on page 28
POST Object restore on page 29
S3 operations tracked in the audit logs on page 78

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.

Unsupported request header

The following request header is not supported and returns `XNotImplemented`:

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

Response headers for Cloud Storage Pool objects

If the object is stored in a Cloud Storage Pool, 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 bucket and optionally transitioned to Glacier 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 Glacier	`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 Glacier, the value for `expiry-date` is set to some distant time in the future. The exact time of transition is not controlled by the StorageGRID system.

Object transitioned to Glacier from the Cloud Storage Pool bucket	`202 OK` `x-amz-storage-class: GLACIER`

Object in process of being restored from Glacier to the Cloud Storage Pool bucket	`200 OK`
	`x-amz-storage-class: GLACIER`
	`x-amz-restore: ongoing-request="true"`

State of object	Response to HEAD object

Object fully restored to the Cloud Storage Pool bucket	`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 temporary copy of the restored object will be deleted from the Cloud Storage Pool bucket.

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 bucket 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 Glacier or when some parts of the object have not yet been restored.

Versioning

If a `versionId` subresource is not specified, the operation fetches the most recent version of the object in a versioned bucket. If the current version of the object is a delete marker, a “Not Found” status is returned with the `x-amz-delete-marker` response header set to `true`.

Related concepts

POST Object restore on page 29
S3 operations tracked in the audit logs on page 78

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`.

Behavior of POST Object restore on Cloud Storage Pool objects

If an object has been moved to a Cloud Storage Pool, a POST Object restore request has the following behavior, based on the state of the object. See “HEAD Object” for more details.

State of object	Behavior of POST Object restore
Object ingested into StorageGRID but not yet evaluated by ILM, or object stored in a traditional storage pool or using erasure coding	`403 Forbidden`, `InvalidObjectState`
Object in Cloud Storage Pool but not yet transitioned to Glacier	`200 OK` No changes are made. **Note:** Before an object has been transitioned to Glacier, you cannot change its `expiry-date`.

State of object	Behavior of POST Object restore
Object transitioned to Glacier from the Cloud Storage Pool bucket	`202 Accepted` A temporary copy of the object is restored to the Cloud Storage Pool bucket for the number of days specified in the request body. After the specified period, the temporary copy is deleted, but the object remains in Glacier. Optionally, specify `versionId` to restore a specific version of an object in a versioned bucket. If you do not specify `versionId`, the most recent version of the object is restored. Optionally, use the `Tier` request element to determine how long the restore job will take to finish (Expedited, Standard, or Bulk). If you do not specify `Tier`, the `Standard` tier is used.
Object in process of being restored from Glacier to the Cloud Storage Pool bucket	`409 Conflict`, `RestoreAlreadyInProgress`
Object fully restored to the Cloud Storage Pool bucket	`200 OK` **Note:** If a temporary copy of an object has been restored, 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 concepts

HEAD Object on page 28
S3 operations tracked in the audit logs on page 78

PUT Object

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

Resolving conflicts

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

Object size

StorageGRID supports objects up to 5 TB in size.

Note: The maximum size for objects that can be replicated to a destination bucket by the CloudMirror replication service is 625 GiB (671,088,640,000 bytes). Objects that match the filtering criteria in the CloudMirror replication configuration XML will not be ingested if they are larger than this size.

User metadata size

AWS 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.

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 AWS 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.

Storage class options

The `x-amz-storage-class` request header is supported with the following values:

- `STANDARD`
  (Default) Specifies a dual-commit ingest operation. As soon as an object is ingested, a second copy of that object is created and distributed to a different Storage Node. When the object is matched by an ILM rule in the active policy, StorageGRID determines if the initial, dual-commit copies satisfy the placement instructions in the rule. If they do not, the object is added to the ILM evaluation queue. When the object is re-evaluated, new object copies might need to be made in different locations, and the initial dual-commit copies might need to be deleted.

- `REDUCED_REDUNDANCY`
  Specifies a single-commit ingest operation. Specify `REDUCED_REDUNDANCY` only if you need to limit redundant storage at the time of ingest and only if you are willing to risk the loss of object data. For example, you can lose data if the single copy is initially stored on a Storage Node that fails before ILM evaluation can occur.
  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. Using `REDUCED_REDUNDANCY` does not result in data being stored at lower levels of redundancy in the StorageGRID system.

  Note: You cannot use the `REDUCED_REDUNDANCY` option if you are ingesting an object into an S3 compliant bucket. This is to ensure that compliance requirements are satisfied (two copies of each object exist) before the object is evaluated by the active ILM policy. See the instructions for administering StorageGRID.

Request headers

The following request headers are supported:

- `x-amz-tagging`
- `x-amz-server-side-encryption`
- `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.

> **Note:** Adding `creation-time` as user-defined metadata is not supported if you are adding an object to a S3 compliant bucket. An error will be returned.

The following request headers are not supported:

- `Expires`
- `x-amz-acl`

The following request headers are not supported and return `XNotImplemented`:

- `Transfer-Encoding`
- `x-amz-server-side-encryption-customer-algorithm`
- `x-amz-server-side-encryption-customer-key`
- `x-amz-server-side-encryption-customer-key-MD5`
- `x-amz-website-redirect-location`

Versioning

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

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

Related concepts

Operations on buckets on page 20
S3 operations tracked in the audit logs on page 78

Related information

Administering StorageGRID

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 copy operation is the same as performing a GET and then a PUT.

Resolving conflicts

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

Object size

StorageGRID supports objects up to 5 TB in size.

> **Note:** The maximum size for objects that can be replicated to a destination bucket by the CloudMirror replication service is 625 GiB (671,088,640,000 bytes). Objects that match the filtering criteria in the CloudMirror replication configuration XML will not be ingested if they are larger than this size.

Storage class options

The `x-amz-storage-class` request header is supported with the following values:

- `STANDARD`
  (Default) Specifies a dual-commit ingest operation.
- `REDUCED_REDUNDANCY`
  Specifies a single-commit ingest operation.

  Note: You cannot use the `REDUCED_REDUNDANCY` option if you are ingesting an object into an S3 compliant bucket. This is to ensure that compliance requirements are satisfied (two copies of each object exist) before the object is evaluated by the active ILM policy. See the instructions for administering StorageGRID.

Request headers

The following request headers are supported:

- `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-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-server-side-encryption`
- `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.

The following request headers are not supported and return `XNotImplemented`:

- `x-amz-server-side-encryption-customer-algorithm`
- `x-amz-server-side-encryption-customer-key`
- `x-amz-server-side-encryption-customer-key-MD5`
- `x-amz-website-redirect-location`

If the source bucket and key, specified in the `x-amz-copy-source` header, is 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.

Note: The `server-side-encryption` value of the object cannot be updated. Instead, make a copy with a new server-side-encryption value using `x-amz-metadata-directive`: `REPLACE`.

Versioning

If the source bucket is versioned, you can use the `x-amz-copy-source` header to copy the latest version of an object. To copy a specific version of an object, you must explicitly specify the version to copy using the `versionId` subresource. If the destination bucket is versioned, the generated version is returned in the `x-amz-version-id` response header. If versioning is suspended for the target bucket, then `x-amz-version-id` returns a "null" value.

Related concepts

S3 operations tracked in the audit logs on page 78

Related information

Administering StorageGRID

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.
- Each multipart part except the last must be between 5 MB and 5 GB. The last part can be less than 5 MB. The client must follow these limits as it is not enforced by StorageGRID.
- All of the multipart upload operations support StorageGRID consistency controls.

Operation	Implementation
List Multipart Uploads	See *List Multipart Uploads* on page 34.
Initiate Multipart Upload	See *Initiate Multipart Upload* on page 35.
Upload Part	See *Upload Part* on page 36
Upload Part - Copy	See *Upload Part - Copy* on page 36
Complete Multipart Upload	See *Complete Multipart Upload* on page 37
Abort Multipart Upload	Implemented with all Amazon S3 REST API behavior.
List Parts	Implemented with all Amazon S3 REST API behavior.

Related concepts

Consistency controls on page 13

List Multipart Uploads

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

The following request parameters are supported:

- `encoding-type`
- `max-uploads`
- `key-marker`

- `prefix`
- `upload-id-marker`

The `delimiter` request parameter is not supported.

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 operation initiates a multipart upload for an object, and returns an upload ID.

The `x-amz-storage-class` request header is supported with the following enumerated values:

- `STANDARD`
  (Default) Specifies a dual-commit ingest operation. As soon as an object is ingested, a second copy of that object is created and distributed to a different Storage Node. When the object is matched by an ILM rule in the active policy, StorageGRID determines if the initial, dual-commit copies satisfy the placement instructions in the rule. If they do not, the object is added to the ILM evaluation queue. When the object is re-evaluated, new object copies might need to be made in different locations, and the initial dual-commit copies might need to be deleted.
- `REDUCED_REDUNDANCY`
  Specifies a single-commit ingest operation. Specify `REDUCED_REDUNDANCY` only if you need to limit redundant storage at the time of ingest and only if you are willing to risk the loss of object data. For example, you might lose data if the single copy was initially stored on a Storage Node that failed before ILM evaluation could occur.
  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 it is evaluated by the active ILM policy. Using `REDUCED_REDUNDANCY` does not result in data being stored at lower levels of redundancy in the StorageGRID system.

  Note: You cannot use the `REDUCED_REDUNDANCY` option if you are ingesting an object into an S3 compliant bucket. This is to ensure that compliance requirements are satisfied (two copies of each object exist) before the object is evaluated by the active ILM policy. See the instructions for administering StorageGRID.

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.

> **Note:** Adding `creation-time` as user-defined metadata is not supported if you are adding an object to a S3 compliant bucket. An error will be returned.

- `x-amz-server-side-encryption` (Previously, you could specify the `x-amz-server-side-encryption` header for each of the upload parts; starting with the StorageGRID 11.2 release, this implementation is no longer supported.)

The following request headers are not supported and return `XNotImplemented`:

- `x-amz-server-side-encryption-customer-algorithm`
- `x-amz-server-side-encryption-customer-key`
- `x-amz-server-side-encryption-customer-key-MD5`
- `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. When the Complete Multipart Upload operation is performed, that is the point when objects are created (and versioned if applicable).

Related information

Administering StorageGRID

Upload Part

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

The following request header is supported:

- `Content-Length`

The following request headers are not supported and return `XNotImplemented`:

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

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).

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.

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`

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).

Complete Multipart Upload

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

Resolving conflicts

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

Object size

StorageGRID supports objects up to 5 TB in size.

Note: The maximum size for objects that can be replicated to a destination bucket by the CloudMirror replication service is 625 GiB (671,088,640,000 bytes). Objects that match the filtering criteria in the CloudMirror replication configuration XML will not be ingested if they are larger than this size.

Request headers

The `x-amz-storage-class` request header is supported with the following enumerated values:

- `STANDARD`
  (Default) Specifies a dual-commit ingest operation.
- `REDUCED_REDUNDANCY`
  Specifies a single-commit ingest operation.

  Note: You cannot use the `REDUCED_REDUNDANCY` option if you are ingesting an object into an S3 compliant bucket. This is to ensure that compliance requirements are satisfied (two copies of each object exist) before the object is evaluated by the active ILM policy. See the instructions for administering StorageGRID.

Attention: 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.

Note: 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 upon 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.

Note: 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-name object 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/bycasterr.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.

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
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

Name	HTTP status
InvalidStorageClass	400 Bad Request
InvalidTag	400 Bad Request
InvalidURI	400 Bad Request
KeyTooLong	400 Bad Request
MalformedXML	400 Bad Request
MetadataTooLarge	400 Bad Request
MethodNotAllowed	405 Method Not Allowed
MissingContentLength	411 Length Required
MissingRequestBodyError	400 Bad Request
MissingSecurityHeader	400 Bad Request
NoSuchBucket	404 Not Found
NoSuchKey	404 Not Found
NoSuchUpload	404 Not Found
NotImplemented	501 Not Implemented
NoSuchBucketPolicy	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
XNoSuchBucketCompliance	The specified bucket does not have compliance enabled.	404 Not Found
XNotImplemented	The request you provided implies functionality that is not implemented.	501 Not Implemented

StorageGRID S3 REST API operations

There are operations added on to the S3 REST API that are specific to StorageGRID system.

PUT Bucket request modifications for compliance

The SGCompliance XML element is a StorageGRID custom element that you can include in the optional XML request body of PUT Bucket requests to create compliant buckets and set compliance settings.

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

If the PUT Bucket request body does not include the SGCompliance XML element, compliance will not be enabled for the bucket. If the PUT Bucket request body includes the SGCompliance XML element, you must specify a value for all child elements.

When compliance is enabled for a bucket, you cannot:

- Enable bucket versioning
- Modify an existing object's data, user-defined metadata, or S3 object tagging

When an object's age is within the retention period, or when legal hold is enabled for the bucket, you cannot:

- Delete the object

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 do not 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.

For more information about configuring compliance and to learn how compliance settings affect information lifecycle management (ILM) rules, see information about administering StorageGRID.

Request example

This example request creates a compliant bucket named `mybucket` in the default region, `us-east-1`. In this example, objects in `mybucket` will be retained for one year (525,600 minutes), starting from when each object is ingested into the grid. There is no legal hold on this bucket. Each object will be automatically deleted after one year.

```
PUT /mybucket HTTP/1.1
Date: date
Authorization: authorization name
Host: host
Content-Length: 272

<CreateBucketConfiguration>
  <LocationConstraint>us-east-1</LocationConstraint>
  <SGCompliance>
    <RetentionPeriodMinutes>525600</RetentionPeriodMinutes>
    <LegalHold>false</LegalHold>
    <AutoDelete>true</AutoDelete>
  </SGCompliance>
</CreateBucketConfiguration>
```

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. You can specify any non-zero positive integer as the value. **Attention:** After the bucket's retention period is set, you cannot decrease that value; you can only increase it. **Note:** If you want to test compliance settings before implementing this feature in a production-level grid, you can set RetentionPeriodMinutes to a small value. However, be aware that it will take longer than an hour to automatically delete objects in compliant buckets, even if the retention period is very short. This is because of the delays due to ILM scanning.
LegalHold	• True: This bucket is currently under a legal hold. Objects in this bucket cannot 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 a PUT Bucket request attempts to remove the compliance settings of an existing compliant bucket or add compliance settings to an existing non-compliant bucket, the HTTP status code for the response is 409 Conflict.

If one of the compliance elements has malformed XML or invalid values, the HTTP status code for the response is 400 Bad Request.

Related information

Administering StorageGRID
Using tenant accounts

GET Bucket compliance request

The GET Bucket compliance request allows you to determine the compliance settings currently in effect for a bucket that was created to be compliant.

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

Note: For more information about configuring compliance and to learn how compliance settings affect ILM rules and policies, see information about administering StorageGRID.

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 cannot 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`.

Related information

Administering StorageGRID
Using tenant accounts

PUT Bucket compliance request

The PUT Bucket compliance request allows you to modify the compliance settings for an existing bucket that was created to be compliant. For example, you might want to place a bucket on legal hold or increase its retention period. You cannot decrease a bucket's retention period.

To create a compliant bucket, you must use the PUT Bucket request with the custom SGCompliance XML request element, the Tenant Manager, or the Tenant Management API.

Note: For more information about configuring compliance and to learn how compliance settings affect ILM rules and policies, see information about administering StorageGRID.

You must 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. **Attention:** 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 cannot decrease that value; you can only increase it.
LegalHold	• True: This bucket is currently under a legal hold. Objects in this bucket cannot 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.

Name	Description
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 cannot 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.

> **Attention:** 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 references

PUT Bucket request modifications for compliance on page 40

Related information

Using tenant accounts
Administering StorageGRID

GET Bucket consistency request

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 must 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. Matches AWS S3 consistency guarantees. **Note:** If your application uses HEAD requests on objects that do not exist, you might receive a high number of 500 Internal Server errors if one or more Storage Nodes are unavailable. To prevent these errors, set the consistency control to “available” unless you require consistency guarantees similar to AWS S3.
available (eventual consistency for HEAD operations)	Behaves the same as the “read-after-new-write” consistency level, but only provides eventual consistency for HEAD operations. Offers higher availability for HEAD operations than “read-after-new-write” if Storage Nodes are unavailable. Differs from AWS S3 consistency guarantees for HEAD operations only.
weak	Provides eventual consistency and high availability, with minimal data protection guarantees, especially if a Storage Node fails or is unavailable. Suitable only for write-heavy workloads that require high availability, do not require read-after-write consistency, and can tolerate the potential loss of data if a node fails.

Response example

```
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"?>
<Consistency xmlns="http://s3.storagegrid.com/doc/2015-02-01/">default</
Consistency>
```

Related concepts

Consistency controls on page 13

PUT Bucket consistency request

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.

You must 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. Matches AWS S3 consistency guarantees. **Note:** If your application uses HEAD requests on objects that do not exist, you might receive a high number of 500 Internal Server errors if one or more Storage Nodes are unavailable. To prevent these errors, set the consistency control to “available” unless you require consistency guarantees similar to AWS S3.
available (eventual consistency for HEAD operations)	Behaves the same as the “read-after-new-write” consistency level, but only provides eventual consistency for HEAD operations. Offers higher availability for HEAD operations than “read-after-new-write” if Storage Nodes are unavailable. Differs from AWS S3 consistency guarantees for HEAD operations only.
weak	Provides eventual consistency and high availability, with minimal data protection guarantees, especially if a Storage Node fails or is unavailable. Suitable only for write-heavy workloads that require high availability, do not require read-after-write consistency, and can tolerate the potential loss of data if a node fails.

Note: In general, you should use the “read-after-new-write” consistency control value. If requests are not 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 concepts

Consistency controls on page 13

GET Bucket last access time request

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

You must 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 request

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

Note: 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** check box 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 do not 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 do not 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

Using tenant accounts

DELETE Bucket metadata notification configuration request

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

You must 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 request

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

You must 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
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

Using tenant accounts

PUT Bucket metadata notification configuration request

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 must 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 are not 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
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

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>
```

Related information

Using tenant accounts

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",
  "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
	key	Object key name
	versionID	Object version, for objects in versioned buckets
System metadata	size	Object size (in bytes) as visible to an HTTP client
	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

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.

You must 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 are not added to the `ObjectCount` total.

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.

Related concepts

S3 bucket policy examples on page 67
S3 group policy examples on page 70

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.

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.

Element	Description
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.
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 are not 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.

Related information

Using tenant accounts

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.

Note: 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.

Using the 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.

Specifying 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.

Related concepts

Specifying variables in a policy on page 65

Specifying 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 do not 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 ("Bob"):

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

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

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

- You can specify an anonymous principal:

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

- If the username `Bob` was deleted upon leaving the organization, and then later on, another Bob joins the organization and was assigned the same username `Bob`, he could have unintentionally inherited the permissions granted to the previous Bob. To avoid such ambiguity, the user UUID can be used instead of the username. For example:

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

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

Specifying 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.

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	
s3:GetBucketAcl	GET Bucket ACL	
s3:GetBucketCompliance	GET Bucket compliance	Yes
s3:GetBucketConsistency	GET Bucket consistency	Yes
s3:GetBucketCORS	GET Bucket cors	
s3:GetBucketLastAccessTime	GET Bucket last access time	Yes
s3:GetBucketLocation	GET Bucket location	
s3:GetBucketMetadataNotification	GET Bucket metadata notification configuration	Yes
s3:GetBucketNotification	GET Bucket notification	

Permissions	S3 REST API operations	Custom for StorageGRID
s3:GetBucketPolicy	GET Bucket policy	
s3:GetReplicationConfiguration	GET Bucket replication	
s3:GetBucketVersioning	GET Bucket versioning	
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	Yes
s3:PutBucketConsistency	PUT Bucket consistency	Yes
s3:PutBucketCORS	DELETE Bucket cors PUT Bucket cors	
s3:PutBucketLastAccessTime	PUT Bucket last access time	Yes
s3:PutBucketMetadataNotification	PUT Bucket metadata notification configuration	Yes
s3:PutBucketNotification	PUT Bucket notification	
s3:PutBucketPolicy	PUT Bucket policy	
s3:PutReplicationConfiguration	PUT Bucket replication	
s3:PutBucketVersioning	PUT Bucket versioning	

Permissions that apply to objects

Permissions	S3 REST API operations	Custom for StorageGRID
s3:AbortMultipartUpload	Abort Multipart Upload, POST Object restore	
s3:DeleteObject	DELETE Object, DELETE Multiple Objects, POST Object restore	
s3:DeleteObjectTagging	DELETE Object Tagging	
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	
s3:GetObjectAcl	GET Object ACL	
s3:GetObjectTagging	GET Object Tagging	

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

Using the 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 cannot 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 cannot 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.

Attention: If the current S3 policy allows overwrite, and the PutOverwriteObject permission is set to Deny, the client cannot overwrite an object's data, user-defined metadata, or object tagging. In addition, if the **Prevent Client Modify** grid option is set to Enabled, that setting overrides the setting of the PutOverwriteObject permission.

Related concepts

S3 group policy examples on page 70

Specifying conditions in a policy

You can use conditions to allow policies to take effect based on request values. 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 equality (case sensitive).
StringNotEquals	Compares a key to a string value based on exact non-equality (case sensitive).
StringEqualsIgnoreCase	Compares a key to a string value based on exact equality (ignores case).
StringNotEqualsIgnoreCase	Compares a key to a string value based on exact non-equality (ignores case).
StringLike	Compares a key to a string value and provides access if there is an exact match (case sensitive). Can include * and ? wildcard characters.
StringNotLike	Compares a key to a string value and provides access to all except the specified string (case sensitive). Can include * and ? wildcard characters.

Condition operators	Description
NumericEquals	Compares a key to a numeric value and provides access if there is an exact match.
NumericNotEquals	Compares a key to a numeric value and provides access to all except the specified value.
NumericGreaterThan	Compares a key to a numeric value and provides access if there is a "greater than" matching.
NumericGreaterThanEquals	Compares a key to a numeric value and provides access if there is a "greater than or equals" matching.
NumericLessThan	Compares a key to a numeric value and provides access if there is a "less than" matching.
NumericLessThanEquals	Compares a key to a numeric value and provides access if there is a "less than or equals" matching.
Bool	Compares a key to a Boolean value and provides access based on a "true or false" matching.
IpAddress	Compares a key to a numeric value and provides access if there is a match to an IP or range of IP addresses.
NotIpAddress	Compares a key to a numeric value and provides access to all addresses except the specified IP or range of IP addresses.
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.
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.
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:max-keys	Will compare to the max-keys parameter specified in a GET Bucket or GET Bucket Object versions request.
	s3:prefix	Will compare to the prefix parameter specified in a GET Bucket or GET Bucket Object versions request.

Specifying 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.

Creating 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, set the **Prevent Client Modify** grid option to Enabled.
- 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.

Caution: 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.

Caution: Even when all of these rules and policies are applied, they do not 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 concepts

Creating policies requiring special handling on page 65
How StorageGRID ILM rules manage objects on page 15
S3 group policy examples on page 70

Related information

Administering StorageGRID

S3 policy examples

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

S3 bucket policy examples

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

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 since 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.

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

```
      ]
    }
  ]
}
```

Related concepts

Operations on buckets on page 20

S3 group policy examples

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

Example: Setting the group policy using the Tenant Manager

When using the Tenant Manager to add or edit a group, you can select how you want to create the group policy that defines which S3 access permissions members of this group will have, as follows:

- **No S3 Access**: Default option. Users in this group do not 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 cannot 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 cannot edit this string.
- **Custom**: Users in the group are granted the permissions you specify in the text box.
  In this example, members of the group are only permitted to list and access their specific folder (key prefix) in the specified bucket.

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}/*"
    }
  ]
}
```

Example: PutOverwriteObject permission

In this example, the `Deny` Effect for PutOverwriteObject and DeleteObject protects the object’s data, user-defined metadata, and S3 object tagging from being deleted or modified.

```
{
  "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/*"
    }
  ]
}
```

Related concepts

Using the PutOverwriteObject permission on page 62
Write-once-read-many (WORM) protection on page 66

Related information

Using tenant accounts

Configuring 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

The StorageGRID system uses Transport Layer Security (TLS) connection security, server authentication, client authentication, and client authorization. When considering security issues, you might find it helpful to understand how the StorageGRID system implements security, authentication, and authorization for the REST API.

The StorageGRID system accepts HTTPS commands submitted over a network connection that uses TLS to provide connection security, application authentication and, optionally, transport encryption. Commands that do not use TLS are rejected. If an object is encrypted when it is ingested, it stays encrypted for the lifetime of the object in the StorageGRID system.

TLS enables the exchange of certificates as entity credentials and allows a negotiation that can use hashing and encryption algorithms.

When a StorageGRID system is installed, a certificate authority (CA) certificate is generated for the system, as well as server certificates for each Storage Node. These server certificates are all signed by the system CA. You need to configure client applications to trust this CA certificate. When a client application connects to any Storage Node using TLS, the application can authenticate the Storage Node by verifying that the server certificate presented by the Storage Node is signed by the trusted system CA.

Alternatively, you can supply a single, custom server certificate that should be used on all Storage Nodes rather than the generated ones. The custom server certificate must be signed by a CA selected by the administrator. The server authentication process by the client application is the same, but uses a different trusted CA.

To configure certificates, see the instructions for administering StorageGRID.

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) **Note:** By request, you can enable OpenStack's Keystone identity service for use with the Swift REST API. If Keystone is enabled, you must use an additional token for validation. To enable Keystone support, contact your NetApp representative.
Client authorization	• S3: Bucket ownership and all applicable access control policies • Swift: Administrator role access

Related information

Administering StorageGRID

Security certificates for client applications

When a client application establishes a TLS session to the StorageGRID system, the system sends a server certificate to the client application for verification to ensure that the HTTPS connection is secure.

The client application loads the grid CA certificate and uses it to verify that the client application is communicating with the expected StorageGRID system. This process protects against man-in-the-middle and impersonation attacks.

Supported hashing and encryption algorithms for TLS libraries

Client applications use the HTTPS protocol to communicate with the StorageGRID system over a network connection that uses Transport Layer Security (TLS). The StorageGRID system supports a limited set of hashing and encryption algorithms from the TLS libraries that client applications can use when establishing a TLS session. When you are setting up the communication processes, it is important for you to know which security algorithms the system uses.

The StorageGRID system supports the following cipher suite security algorithms:

TLS version	Cipher suite	Benefit
v1.1	TLS_RSA_WITH_AES_128_CBC_SHA	**Note:** TLS v1.1 is deprecated. Support for TLS v1.1 will be removed in a future StorageGRID release.
	TLS_RSA_WITH_AES_256_CBC_SHA	
v1.2	TLS_RSA_WITH_AES_128_CBC_SHA	**Note:** Support for CBC ciphers and SHA1 ciphers is deprecated. Support for these ciphers will be removed in a future release.
	TLS_RSA_WITH_AES_256_CBC_SHA	
	TLS_RSA_WITH_AES_128_GCM_SHA256	Provide secure encryption and more efficient processing of large objects.
	TLS_RSA_WITH_AES_256_GCM_SHA384	
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	Support perfect forward secrecy.
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	

The TLS session negotiates the connection, using either AES128 or AES256 based on the client application requirements, and the need to balance performance with encryption security.

Attention: SSLv3 is no longer supported for connections to the Storage Node or API Gateway Node.

Monitoring and auditing operations

You can monitor workloads and efficiencies for client operations by viewing transaction counts for all Storage Nodes or a specific Storage Node. You can use audit messages to monitor client operations and transactions.

Monitoring 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.

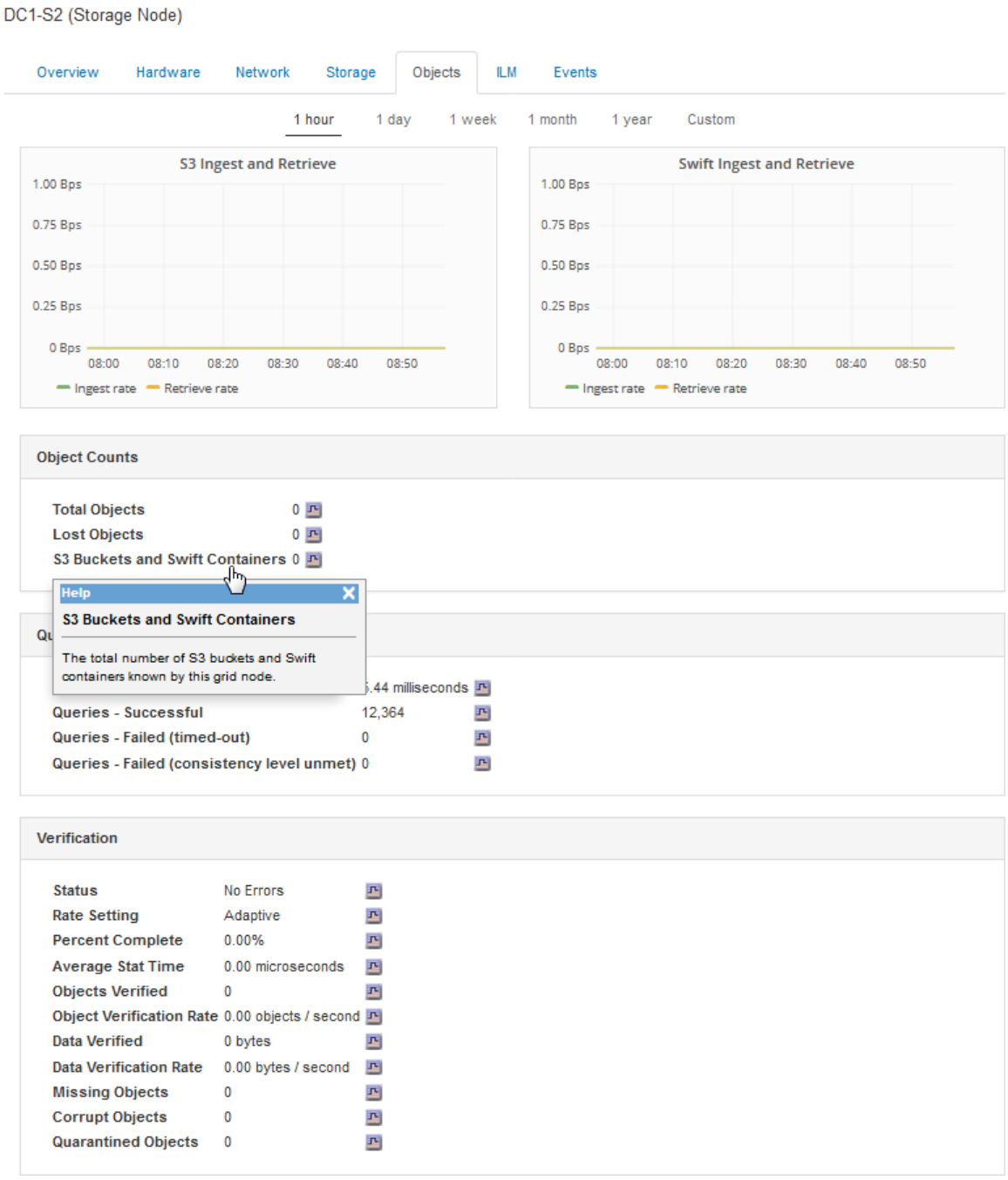
Steps

1. Sign in to the Grid Manager using a supported browser.
2. On the Dashboard, locate the Protocol Operations section.

   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 **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.
5. 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.



6. If you want even more detail:
   a. Select **Support > 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.

Accessing and reviewing 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 must be a StorageGRID administrator with specific access permissions.
- You must have the `Passwords.txt` file. This file is included in the Recovery Package `.zip` file.
- You must 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@`*`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 Multiple Objects

- GET Bucket (List Objects)
- GET Bucket Object versions
- HEAD Bucket
- PUT Bucket
- PUT Bucket compliance
- PUT Bucket Versioning

Object operations tracked in the audit logs

- Complete Multipart Upload
- DELETE Object
- GET Object
- HEAD Object
- POST Object restore
- PUT Object
- PUT Object - Copy

Related concepts

Operations on buckets on page 20
Operations on objects on page 25

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 HTTP connection open for a maximum of 10 minutes. The LDR service 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

You should limit the duration of an active HTTP connection for 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.

Limited active HTTP connections provide the following benefits:

- Enables optimal load balancing across the StorageGRID system
  To optimize load balancing across the StorageGRID system, you should prevent long-lived TCP/IP connections. You should configure client applications to track the duration of each HTTP connection and close the HTTP connection after a set time so that the HTTP connection can be reestablished and rebalanced.
  The StorageGRID system balances its load when a client application establishes an HTTP connection. Over time, an HTTP connection that the StorageGRID system uses for a compute

resource 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 maintenance procedures to start
  Some maintenance procedures start only after all the in-progress HTTP connections are complete.
- Allows client applications to direct HTTP transactions to LDR services that have available space.

Benefits of concurrent HTTP connections

You must keep multiple TCP/IP connections to the StorageGRID system open to allow idle connections to perform transactions as required. The number of client applications also affects how you handle multiple TCP/IP connections.

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, either on a client-by-client basis or on a connection-pool basis. 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

Trademark

How to send comments about documentation and receive update notifications

You can help us to improve the quality of our documentation by sending us your feedback. You can receive automatic notification when production-level (GA/FCS) documentation is initially released or important changes are made to existing production-level documents.

If you have suggestions for improving this document, send us your comments by email.

doccomments@netapp.com

To help us direct your comments to the correct division, include in the subject line the product name, version, and operating system.

If you want to be notified automatically when production-level documentation is released or important changes are made to existing production-level documents, follow Twitter account @NetAppDoc.

You can also contact us in the following ways:

- NetApp, Inc., 1395 Crossman Ave., Sunnyvale, CA 94089 U.S.
- Telephone: +1 (408) 822-6000
- Fax: +1 (408) 822-4501
- Support telephone: +1 (888) 463-8277