Use Swift REST API (deprecated)

StorageGRID 11.8

NetApp
May 17, 2024

Table of Contents

Use Swift REST API (deprecated)

Use Swift REST API: Overview

Client applications can use the OpenStack Swift API to interface with the StorageGRID system.

Support for Swift client applications has been deprecated and will be removed in a future release.

StorageGRID supports the following specific versions of Swift and HTTP.

Item	Version
Swift specification	OpenStack Swift Object Storage API v1 as of November 2015
HTTP	1.1 For more information about HTTP, see HTTP/1.1 (RFCs 7230-35). **Note**: StorageGRID does not support HTTP/1.1 pipelining.

Related information

OpenStack: Object Storage API

History of Swift API support in StorageGRID

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

Release	Comments
11.8	
11.7	Support for Swift client applications has been deprecated and will be removed in a future release.
11.6	Minor editorial changes.
11.5	Removed Weak consistency. The Available consistency will be used instead.
11.4	Added support for TLS 1.3. Added description of interrelationship between ILM and consistency.

Release	Comments
11.3	Updated PUT Object operations to describe the impact of ILM rules that use synchronous placement at ingest (the Balanced and Strict options for Ingest Behavior). Added description of client connections that use load balancer endpoints or high availability groups. TLS 1.1 ciphers are no longer supported.
11.2	Minor editorial changes to document.
11.1	Added support for using HTTP for Swift client connections to grid nodes. Updated the definitions of consistency values.
11.0	Added support for 1,000 containers for each tenant account.
10.3	Administrative updates and corrections to the document. Removed sections for configuring custom server certificates.
10.2	Initial support of the Swift API by the StorageGRID system. The currently supported version is OpenStack Swift Object Storage API v1.

How StorageGRID implements Swift REST API

A client application can use Swift REST API calls to connect to Storage Nodes and Gateway Nodes to create containers and to store and retrieve objects. This enables service-oriented applications developed for OpenStack Swift to connect with on-premise object storage provided by the StorageGRID system.

Swift object management

After Swift objects have been ingested in the StorageGRID system, they are managed by the information lifecycle management (ILM) rules in the active ILM policies. ILM rules and ILM policies determine how StorageGRID creates and distributes copies of object data and how it manages those copies over time. For example, an ILM rule might apply to objects in specific Swift containers and might specify that multiple object copies be saved to several data centers for a certain number of years.

Contact your NetApp Professional Services consultant or StorageGRID administrator if you need to understand how the grid's ILM rules and policies will affect the objects in your Swift tenant account.

Conflicting client requests

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

Consistency guarantees and controls

By default, StorageGRID provides read-after-write consistency for newly created objects and eventual consistency for object updates and HEAD operations. 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.

StorageGRID also allows you to control consistency on a per container basis. Consistency values provide a balance between the availability of the objects and the consistency of those objects across different Storage Nodes and sites, as required by your application.

Recommendations for implementing Swift REST API

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

Recommendations for HEADs to non-existent objects

If your application routinely checks to see if an object exists at a path where you don't expect the object to actually exist, you should use the "Available" consistency. For example, you should use the "Available" consistency if your application performs a HEAD operation to a location before performing a PUT operation to that location.

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 for each container using the PUT container consistency request. You view set the "Available" consistency for each container using the GET container consistency request.

Recommendations for object names

For containers that are created in StorageGRID 11.4 or later, restricting object names to meet performance best practices is no longer required. For example, you can now use random values for the first four characters of object names.

For containers that were created in releases earlier than StorageGRID 11.4, continue to follow these recommendations for object names:

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

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

  Instead of this format:

  ```
  mycontainer/f8e3-image3132.jpg
  ```

Recommendations for "range reads"

If the global option to compress stored objects is enabled, Swift 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.

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

Test Swift REST API configuration

You can use the Swift CLI to test your connection to the StorageGRID system and to verify that you can read and write objects.

Before you begin

- You have downloaded and installed the Swift command-line client: SwiftStack: python-swiftclient
- Optionally, you have created a load balancer endpoint. Otherwise, you know the IP address of the Storage Node you want to connect to and the port number to use. See IP addresses and ports for client connections.
- You have created a Swift tenant account.
- You have signed in to the tenant account and created at least one group and user. See Create groups for a Swift tenant.

> Swift tenant users must have the Administrator group permission to authenticate into the Swift REST API.

About this task

If you have not configured security, you must add the `--insecure` flag to each of these commands.

Steps

1. Query the info URL for your StorageGRID Swift deployment:

```
swift
-U <Tenant_Account_ID:Account_User_Name>
-K <User_Password>
-A https://<FQDN | IP>:<Port>/info
capabilities
```

   This is sufficient to test that your Swift deployment is functional. To further test account configuration by storing an object, continue with the additional steps.

2. Put an object in the container:

```
touch test_object
swift
-U <Tenant_Account_ID:Account_User_Name>
-K <User_Password>
-A https://<FQDN | IP>:<Port>/auth/v1.0
upload test_container test_object
--object-name test_object
```

3. Get the container to verify the object:

```
swift
-U <Tenant_Account_ID:Account_User_Name>
-K <User_Password>
-A https://<FQDN | IP>:<Port>/auth/v1.0
list test_container
```

4. Delete the object:

```
swift
-U <Tenant_Account_ID:Account_User_Name>
-K <User_Password>
-A https://<FQDN | IP>:<Port>/auth/v1.0
delete test_container test_object
```

5. Delete the container:

```
swift
-U `<_Tenant_Account_ID:Account_User_Name_>`
-K `<_User_Password_>`
-A `\https://<_FQDN_ | _IP_>:<_Port_>/auth/v1.0'
delete test_container
```

Swift REST API supported operations

The StorageGRID system supports most operations in the OpenStack Swift API. Before integrating Swift REST API clients with StorageGRID, review the implementation details for account, container, and object operations.

Operations supported in StorageGRID

The following Swift API operations are supported:

- Account operations
- Container operations
- Object operations

Common response headers for all operations

The StorageGRID system implements all common headers for supported operations as defined by the OpenStack Swift Object Storage API v1.

Related information

OpenStack: Object Storage API

Supported Swift API endpoints

StorageGRID supports the following Swift API endpoints: the info URL, the auth URL, and the storage URL.

info URL

You can determine the capabilities and limitations of the StorageGRID Swift implementation by issuing a GET request to the Swift base URL with the /info path.

```
https://FQDN | Node IP:Swift Port/info/
```

In the request:

- *FQDN* is the fully qualified domain name.
- *Node IP* is the IP address for the Storage Node or the Gateway Node on the StorageGRID network.
- *Swift Port* is the port number used for Swift API connections on the Storage Node or Gateway Node.

For example, the following info URL would request information from a Storage Node with the IP address of 10.99.106.103 and using port 18083.

```
https://10.99.106.103:18083/info/
```

The response includes the capabilities of the Swift implementation as a JSON dictionary. A client tool can parse the JSON response to determine the capabilities of the implementation and use them as constraints for subsequent storage operations.

The StorageGRID implementation of Swift allows unauthenticated access to the info URL.

auth URL

A client can use the Swift auth URL to authenticate as a tenant account user.

```
https://FQDN | Node IP:Swift Port/auth/v1.0/
```

You must provide the tenant account ID, user name, and password as parameters in the `X-Auth-User` and `X-Auth-Key` request headers, as follows:

```
X-Auth-User: Tenant_Account_ID:Username
```

```
X-Auth-Key: Password
```

In the request headers:

- *Tenant_Account_ID* is the account ID assigned by StorageGRID when the Swift tenant was created. This is the same tenant account ID used on the Tenant Manager sign-in page.
- *Username* is the name of a tenant user that has been created in the Tenant Manager. This user must belong to a group that has the Swift Administrator permission. The tenant's root user can't be configured to use the Swift REST API.

  If Identity Federation is enabled for the tenant account, provide the username and password of the federated user from the LDAP server. Alternatively, provide the LDAP user's domain name. For example:

  ```
  X-Auth-User: Tenant_Account_ID:Username@Domain_Name
  ```

- *Password* is the password for the tenant user. User passwords are created and managed in the Tenant Manager.

The response to a successful authentication request returns a storage URL and an auth token, as follows:

```
X-Storage-Url: https://FQDN | Node_IP:Swift_Port/v1/Tenant_Account_ID

X-Auth-Token: token

X-Storage-Token: token
```

By default, the token is valid for 24 hours from generation time.

Tokens are generated for a specific tenant account. A valid token for one account does not authorize a user to access another account.

storage URL

A client application can issue Swift REST API calls to perform supported account, container, and object operations against a Gateway Node or Storage Node. Storage requests are addressed to the storage URL returned in the authentication response. The request must also include the X-Auth-Token header and value returned from the auth request.

```
https://FQDN | IP:Swift_Port/v1/Tenant_Account_ID

[/container][/object]

X-Auth-Token: token
```

Some storage response headers that contain usage statistics might not reflect accurate numbers for recently modified objects. It might take a few minutes for accurate numbers to appear in these headers.

The following response headers for account and container operations are examples of those that contain usage statistics:

- `X-Account-Bytes-Used`
- `X-Account-Object-Count`

- `X-Container-Bytes-Used`
- `X-Container-Object-Count`

Related information

Configure tenant accounts and connections

Account operations

Container operations

Object operations

Account operations

The following Swift API operations are performed on accounts.

GET account

This operation retrieves the container list associated with the account and account usage statistics.

The following request parameter is required:

- `Account`

The following request header is required:

- `X-Auth-Token`

The following supported request query parameters are optional:

- `Delimiter`
- `End_marker`
- `Format`
- `Limit`
- `Marker`
- `Prefix`

A successful execution returns the following headers with an "HTTP/1.1 204 No Content" response if the account is found and has no containers or the container list is empty; or an "HTTP/1.1 200 OK" response if the account is found and the container list is not empty:

- `Accept-Ranges`
- `Content-Length`
- `Content-Type`
- `Date`
- `X-Account-Bytes-Used`
- `X-Account-Container-Count`

- `X-Account-Object-Count`
- `X-Timestamp`
- `X-Trans-Id`

HEAD account

This operation retrieves account information and statistics from a Swift account.

The following request parameter is required:

- `Account`

The following request header is required:

- `X-Auth-Token`

A successful execution returns the following headers with an "HTTP/1.1 204 No Content" response:

- `Accept-Ranges`
- `Content-Length`
- `Date`
- `X-Account-Bytes-Used`
- `X-Account-Container-Count`
- `X-Account-Object-Count`
- `X-Timestamp`
- `X-Trans-Id`

Related information

Swift operations tracked in the audit logs

Container operations

StorageGRID supports a maximum of 1,000 containers per Swift account. The following Swift API operations are performed on containers.

DELETE container

This operation removes an empty container from a Swift account in a StorageGRID system.

The following request parameters are required:

- `Account`
- `Container`

The following request header is required:

- `X-Auth-Token`

A successful execution returns the following headers with an "HTTP/1.1 204 No Content" response:

- `Content-Length`
- `Content-Type`
- `Date`
- `X-Trans-Id`

GET container

This operation retrieves the object list associated with the container along with container statistics and metadata in a StorageGRID system.

The following request parameters are required:

- `Account`
- `Container`

The following request header is required:

- `X-Auth-Token`

The following supported request query parameters are optional:

- `Delimiter`
- `End_marker`
- `Format`
- `Limit`
- `Marker`
- `Path`
- `Prefix`

A successful execution returns the following headers with an "HTTP/1.1 200 Success" or a "HTTP/1.1 204 No Content" response:

- `Accept-Ranges`
- `Content-Length`
- `Content-Type`
- `Date`
- `X-Container-Bytes-Used`
- `X-Container-Object-Count`
- `X-Timestamp`
- `X-Trans-Id`

HEAD container

This operation retrieves container statistics and metadata from a StorageGRID system.

The following request parameters are required:

- `Account`
- `Container`

The following request header is required:

- `X-Auth-Token`

A successful execution returns the following headers with an "HTTP/1.1 204 No Content" response:

- `Accept-Ranges`
- `Content-Length`
- `Date`
- `X-Container-Bytes-Used`
- `X-Container-Object-Count`
- `X-Timestamp`
- `X-Trans-Id`

PUT container

This operation creates a container for an account in a StorageGRID system.

The following request parameters are required:

- `Account`
- `Container`

The following request header is required:

- `X-Auth-Token`

A successful execution returns the following headers with an "HTTP/1.1 201 Created" or "HTTP/1.1 202 Accepted" (if the container already exists under this account) response:

- `Content-Length`
- `Date`
- `X-Timestamp`
- `X-Trans-Id`

A container name must be unique in the StorageGRID namespace. If the container exists under another account, the following header is returned: "HTTP/1.1 409 Conflict."

Related information

Object operations

The following Swift API operations are performed on objects. These operations can be tracked in the StorageGRID audit log.

DELETE object

This operation deletes an object's content and metadata from the StorageGRID system.

The following request parameters are required:

- `Account`
- `Container`
- `Object`

The following request header is required:

- `X-Auth-Token`

A successful execution returns the following response headers with an `HTTP/1.1 204 No Content` response:

- `Content-Length`
- `Content-Type`
- `Date`
- `X-Trans-Id`

When processing a DELETE Object request, StorageGRID attempts to immediately remove all copies of the object from all stored locations. If successful, StorageGRID returns a response to the client immediately. If all copies can't be removed within 30 seconds (for example, because a location is temporarily unavailable), StorageGRID queues the copies for removal and then indicates success to the client.

For more information, see How objects are deleted.

GET object

This operation retrieves the object content and gets the object metadata from a StorageGRID system.

The following request parameters are required:

- `Account`
- `Container`
- `Object`

The following request header is required:

- `X-Auth-Token`

The following request headers are optional:

- `Accept-Encoding`
- `If-Match`
- `If-Modified-Since`
- `If-None-Match`
- `If-Unmodified-Since`
- `Range`

A successful execution returns the following headers with an `HTTP/1.1 200 OK` response:

- `Accept-Ranges`
- `Content-Disposition`, returned only if `Content-Disposition` metadata was set
- `Content-Encoding`, returned only if `Content-Encoding` metadata was set
- `Content-Length`
- `Content-Type`
- `Date`
- `ETag`
- `Last-Modified`
- `X-Timestamp`
- `X-Trans-Id`

HEAD object

This operation retrieves metadata and properties of an ingested object from a StorageGRID system.

The following request parameters are required:

- `Account`
- `Container`
- `Object`

The following request header is required:

- `X-Auth-Token`

A successful execution returns the following headers with an "HTTP/1.1 200 OK" response:

- `Accept-Ranges`
- `Content-Disposition`, returned only if `Content-Disposition` metadata was set
- `Content-Encoding`, returned only if `Content-Encoding` metadata was set
- `Content-Length`

- `Content-Type`
- `Date`
- `ETag`
- `Last-Modified`
- `X-Timestamp`
- `X-Trans-Id`

PUT object

This operation creates a new object with data and metadata, or replaces an existing object with data and metadata in a StorageGRID system.

StorageGRID supports objects up to 5 TiB (5,497,558,138,880 bytes) in size.

> 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 Swift clients begin an operation.

The following request parameters are required:

- `Account`
- `Container`
- `Object`

The following request header is required:

- `X-Auth-Token`

The following request headers are optional:

- `Content-Disposition`
- `Content-Encoding`

  Don’t use chunked `Content-Encoding` if the ILM rule that applies to an object filters objects based on size and uses synchronous placement on ingest (the Balanced or Strict options for Ingest Behavior).

- `Transfer-Encoding`

  Don’t use compressed or chunked `Transfer-Encoding` if the ILM rule that applies to an object filters objects based on size and uses synchronous placement on ingest (the Balanced or Strict options for Ingest Behavior).

- `Content-Length`

  If an ILM rule filters objects by size and uses synchronous placement on ingest, you must specify `Content-Length`.

> If you don't follow these guidelines for `Content-Encoding`, `Transfer-Encoding`, and `Content-Length`, StorageGRID must save the object before it can determine object size and apply the ILM rule. In other words, StorageGRID must default to creating interim copies of an object on ingest. That is, StorageGRID must use the Dual Commit option for Ingest Behavior.

For more information about synchronous placement and ILM rules, see Data-protection options for ingest.

- `Content-Type`
- `ETag`
- `X-Object-Meta-<name\>` (object-related metadata)

  If you want to use the **User defined creation time** option as the Reference time for an ILM rule, you must store the value in a user-defined header named `X-Object-Meta-Creation-Time`. For example:

  ```
  X-Object-Meta-Creation-Time: 1443399726
  ```

  This field is evaluated as seconds since January 1, 1970.

- `X-Storage-Class: reduced_redundancy`

  This header affects how many object copies StorageGRID creates if the ILM rule that matches an ingested object specifies an Ingest Behavior of Dual Commit or Balanced.

  - **Dual commit**: If the ILM rule specifies the Dual commit option for Ingest Behavior, StorageGRID creates a single interim copy as the object is ingested (single commit).
  - **Balanced**: If the ILM rule specifies the Balanced option, StorageGRID makes a single interim copy only if the system can't immediately make all copies specified in the rule. If StorageGRID can perform synchronous placement, this header has no effect.

    The `reduced_redundancy` header is best used when the ILM rule that matches the object creates a single replicated copy. In this case using `reduced_redundancy` eliminates the unnecessary creation and deletion of an extra object copy for every ingest operation.

    Using the `reduced_redundancy` header is not recommended in other circumstances because it increases the risk the loss of object data during ingest. For example, you might lose data if the single copy is initially stored on a Storage Node that fails before ILM evaluation can occur.

    > Having only one replicated copy for any time period puts data at risk of permanent loss. If only one replicated copy of an object exists, that object is lost if a Storage Node fails or has a significant error. You also temporarily lose access to the object during maintenance procedures such as upgrades.

  Note that specifying `reduced_redundancy` only affects how many copies are created when an object is first ingested. It does not affect how many copies of the object are made when the object is evaluated by the active ILM policies and does not result in data being stored at lower levels of redundancy in the StorageGRID system.

A successful execution returns the following headers with an "HTTP/1.1 201 Created" response:

- `Content-Length`
- `Content-Type`
- `Date`
- `ETag`
- `Last-Modified`
- `X-Trans-Id`

OPTIONS request

The OPTIONS request checks the availability of an individual Swift service. The OPTIONS request is processed by the Storage Node or Gateway Node specified in the URL.

OPTIONS method

For example, client applications can issue an OPTIONS request to the Swift port on a Storage Node, without providing Swift 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.

When used with the info URL or the storage URL, the OPTIONS method returns a list of supported verbs for the given URL (for example, HEAD, GET, OPTIONS, and PUT). The OPTIONS method can’t be used with the auth URL.

The following request parameter is required:

- `Account`

The following request parameters are optional:

- `Container`
- `Object`

A successful execution returns the following headers with an "HTTP/1.1 204 No Content" response. The OPTIONS request to the storage URL does not require that the target exists.

- `Allow` (a list of supported verbs for the given URL, for example, HEAD, GET, OPTIONS, and PUT)
- `Content-Length`
- `Content-Type`
- `Date`
- `X-Trans-Id`

Related information

Supported Swift API endpoints

Error responses to Swift API operations

Understanding the possible error responses can help you troubleshoot operations.

The following HTTP status codes might be returned when errors occur during an operation:

Swift error name	HTTP status
AccountNameTooLong, ContainerNameTooLong, HeaderTooBig, InvalidContainerName, InvalidRequest, InvalidURI, MetadataNameTooLong, MetadataValueTooBig, MissingSecurityHeader, ObjectNameTooLong, TooManyContainers, TooManyMetadataItems, TotalMetadataTooLarge	400 Bad Request
AccessDenied	403 Forbidden
ContainerNotEmpty, ContainerAlreadyExists	409 Conflict
InternalError	500 Internal Server Error
InvalidRange	416 Requested Range Not Satisfiable
MethodNotAllowed	405 Method Not Allowed
MissingContentLength	411 Length Required
NotFound	404 Not Found
NotImplemented	501 Not Implemented
PreconditionFailed	412 Precondition Failed
ResourceNotFound	404 Not Found
Unauthorized	401 Unauthorized
UnprocessableEntity	422 Unprocessable Entity

StorageGRID Swift REST API operations

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

GET container consistency request

Consistency values provide a balance between the availability of the objects and the consistency of those

objects across different Storage Nodes and sites. The GET container consistency request allows you to determine the consistency being applied to a particular container.

Request

Request HTTP Header	Description
`X-Auth-Token`	Specifies the Swift authentication token for the account to use for the request.
`x-ntap-sg-consistency`	Specifies the type of request, where `true` = GET container consistency, and `false` = GET container.
`Host`	The hostname to which the request is directed.

Request example

```
GET /v1/28544923908243208806/Swift container
X-Auth-Token: SGRD_3a877009a2d24cb1801587bfa9050f29
x-ntap-sg-consistency: true
Host: test.com
```

Response

Response HTTP Header	Description
`Date`	The date and time of the response.
`Connection`	Whether the connection to the server is open or closed.
`X-Trans-Id`	The unique transaction identifier for the request.
`Content-Length`	The length of the response body.

Response HTTP Header	Description
`x-ntap-sg-consistency`	The consistency being applied to the container. The following values are supported: **All**: All nodes receive the data immediately or the request will fail. **Strong-global**: Guarantees read-after-write consistency for all client requests across all sites. **Strong-site**: Guarantees read-after-write consistency for all client requests within a site. **Read-after-new-write**: (Default) Provides read-after-write consistency for new objects and eventual consistency for object updates. Offers high availability and data protection guarantees. Recommended for most cases. **Available**: Provides eventual consistency for both new objects and object updates. For S3 buckets, use only as required (for example, for a bucket that contains log values that are rarely read, or for HEAD or GET operations on keys that don't exist). Not supported for S3 FabricPool buckets.

Response example

```
HTTP/1.1 204 No Content
Date: Sat, 29 Nov 2015 01:02:18 GMT
Connection: CLOSE
X-Trans-Id: 1936575373
Content-Length: 0
x-ntap-sg-consistency: strong-site
```

PUT container consistency request

The PUT container consistency request allows you to specify the consistency to apply to operations performed on a container. By default, new containers are created using the "Read-after-new-write" consistency.

Request

Request HTTP Header	Description
`X-Auth-Token`	The Swift authentication token for the account to use for the request.

Request HTTP Header	Description
`x-ntap-sg-consistency`	The consistency to apply to operations on the container. The following values are supported: **All**: All nodes receive the data immediately or the request will fail. **Strong-global**: Guarantees read-after-write consistency for all client requests across all sites. **Strong-site**: Guarantees read-after-write consistency for all client requests within a site. **Read-after-new-write**: (Default) Provides read-after-write consistency for new objects and eventual consistency for object updates. Offers high availability and data protection guarantees. Recommended for most cases. **Available**: Provides eventual consistency for both new objects and object updates. For S3 buckets, use only as required (for example, for a bucket that contains log values that are rarely read, or for HEAD or GET operations on keys that don't exist). Not supported for S3 FabricPool buckets.
`Host`	The hostname to which the request is directed.

How consistency and ILM rules interact to affect data protection

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

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

Example of how consistency and ILM rules can interact

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

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

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

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

If you instead used the same ILM rule and the "Strong-site" consistency, the client might receive a success

message after object data is replicated to the remote site but before object metadata is distributed there. In this case, the level of protection of object metadata does not match the level of protection for object data. If the local site is lost shortly after ingest, object metadata is lost. The object can't be retrieved.

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

Request example

```
PUT /v1/28544923908243208806/_Swift container_
X-Auth-Token: SGRD_3a877009a2d24cb1801587bfa9050f29
x-ntap-sg-consistency: strong-site
Host: test.com
```

Response

Response HTTP Header	Description
`Date`	The date and time of the response.
`Connection`	Whether the connection to the server is open or closed.
`X-Trans-Id`	The unique transaction identifier for the request.
`Content-Length`	The length of the response body.

Response example

```
HTTP/1.1 204 No Content
Date: Sat, 29 Nov 2015 01:02:18 GMT
Connection: CLOSE
X-Trans-Id: 1936575373
Content-Length: 0
```

Swift operations tracked in the audit logs

All successful storage DELETE, GET, HEAD, POST, and PUT operations are tracked in the StorageGRID audit log. Failures and info, auth, or OPTIONS requests are not logged.

Account operations

- GET account
- HEAD account

Container operations

- DELETE container
- GET container
- HEAD container
- PUT container

Object operations

- DELETE object
- GET object
- HEAD object
- PUT object

Related information

- Access audit log file
- Client write audit messages
- Client read audit messages

Copyright information

Trademark information