Use Swift REST API (deprecated)

StorageGRID 11.7

NetApp
April 12, 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
11.7	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 control. The Available consistency level will be used instead.
11.4	Added support for TLS 1.3. Added description of interrelationship between ILM and consistency setting.

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 controls.
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 system's active ILM policy. The ILM rules and ILM policy 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 controls provide a balance between the availability of the objects and the consistency of those objects across different Storage Nodes and sites, as required by your application.

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 control. For example, you should use the "Available" consistency control 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 control for each container using the PUT container consistency request. You view set the "Available" consistency control 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.

Configure tenant accounts and connections

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

Create and configure Swift tenant accounts

A Swift tenant account is required before Swift API clients can store and retrieve objects on StorageGRID. Each tenant account has its own account ID, groups and users, and containers and objects.

Swift tenant accounts are created by a StorageGRID grid administrator using the Grid Manager or the Grid Management API.

When creating a Swift tenant account, the grid administrator specifies the following information:

- Display name for the tenant (the tenant's account ID is assigned automatically and can't be changed)
- 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 a Swift tenant account is created, users with the Root access permission 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
- Monitoring storage usage

> Swift users must have the Root access permission to access the Tenant Manager. However, the Root access permission does not allow users to authenticate into the Swift REST API to create containers and ingest objects. Users must have the Swift Administrator permission to authenticate into the Swift REST API.

How client connections can be configured

A grid administrator makes configuration choices that affect how Swift clients connect to StorageGRID to store and retrieve data. The specific information you need to make a connection depends upon the configuration that was chosen.

Client applications can store or retrieve objects by connecting to the Load Balancer service on Admin Nodes or Gateway Nodes, or optionally, the virtual IP address of a high availability (HA) group of Admin Nodes or Gateway Nodes.

> All applications that depend on StorageGRID to provide load balancing should connect using the Load Balancer service.

- Storage Nodes, with or without an external load balancer

When configuring StorageGRID, a grid administrator can use the Grid Manager or the Grid Management API to perform the following steps, all of which are optional:

1. Configure endpoints for the Load Balancer service.

   You must configure endpoints to use the Load Balancer service. The Load Balancer service on Admin Nodes or Gateway Nodes distributes incoming network connections from client applications to Storage Nodes. When creating a load balancer endpoint, the StorageGRID administrator specifies a port number, whether the endpoint accepts HTTP or HTTPS connections, the type of client (S3 or Swift) that will use the endpoint, and the certificate to be used for HTTPS connections (if applicable). Swift supports these endpoint types.

2. Configure Untrusted Client Networks.

   If a StorageGRID administrator configures a node's Client Network to be untrusted, the node only accepts inbound connections on the Client Network on ports that are explicitly configured as load balancer endpoints.

3. Configure high availability groups.

   If an administrator creates an HA group, the network interfaces of multiple Admin Nodes or Gateway Nodes are placed into an active-backup configuration. Client connections are made using the virtual IP address of the HA group.

See Configuration options for HA groups for more information.

Summary: IP addresses and ports for client connections

Client applications connect to StorageGRID using the IP address of a grid node and the port number of a service on that node. If high availability (HA) groups are configured, client applications can connect using the virtual IP address of the HA group.

Information required to make client connections

The table summarizes the different ways that clients can connect to StorageGRID and the IP addresses and ports that are used for each type of connection. See IP addresses and ports for client connections or contact your StorageGRID administrator for more information.

Where connection is made	Service that client connects to	IP address	Port
HA group	Load Balancer	Virtual IP address of an HA group	• Load balancer endpoint port

Where connection is made	Service that client connects to	IP address	Port
Admin Node	Load Balancer	IP address of the Admin Node	• Load balancer endpoint port
Gateway Node	Load Balancer	IP address of the Gateway Node	• Load balancer endpoint port
Storage Node	LDR	IP address of Storage Node	Default Swift ports: • HTTPS: 18083 • HTTP: 18085

Example

To connect a Swift client to the Load Balancer endpoint of an HA group of Gateway Nodes, use a URL structured as shown below:

- `https://VIP-of-HA-group:LB-endpoint-port`

For example, if the virtual IP address of the HA group is 192.0.2.6 and the port number of a Swift Load Balancer endpoint is 10444, then a Swift client could use the following URL to connect to StorageGRID:

- `https://192.0.2.6:10444`

It is possible to configure a DNS name for the IP address that clients use to connect to StorageGRID. Contact your local network administrator.

Decide to use HTTPS or HTTP connections

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

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

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

If the **Enable HTTP for Storage Node connections** option is selected, clients must use different ports for HTTP than they use for HTTPS.

Test your connection in Swift 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 to the system.

Before you begin

- You must have downloaded and installed python-swiftclient, the Swift command-line client.

  SwiftStack: python-swiftclient

- You must have a Swift tenant account in the StorageGRID system.

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
```

Related information

Create and configure Swift tenant accounts

Configure security for the REST API

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

Monitor and audit operations

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

Monitor and audit operations

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 policy 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

Swift error name	HTTP status
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 controls 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 level 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.
`x-ntap-sg-consistency`	The consistency control level 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 level to apply to operations performed on a container. By default, new containers are created using the “read-after-new-write” consistency level.

Request

Request HTTP Header	Description
`X-Auth-Token`	The Swift authentication token for the account to use for the request.
`x-ntap-sg-consistency`	The consistency control level 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 controls and ILM rules interact to affect data protection

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

For example, the consistency control used when an object is stored affects the initial placement of object metadata, while the ingest behavior selected for the ILM rule affects the initial placement of object copies.

Because StorageGRID requires access to both an object's metadata and its data to fulfill client requests, selecting matching levels of protection for the consistency level and ingest behavior can provide better initial data protection and more predictable system responses.

Example of how consistency control and ILM rule can interact

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

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

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

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

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

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

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
```

Configure security for the REST API

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

How StorageGRID provides security for the REST API

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

StorageGRID uses the following security measures.

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

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

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

  Optionally, enable HTTP for these connections. For example, you might want to use HTTP for testing or other non-production purposes.

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

Security certificates and client applications

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

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

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

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

See configuring load balancer endpoints and adding a single custom server certificate for TLS connections directly to Storage Nodes.

Summary

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

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

Supported hashing and encryption algorithms for TLS libraries

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

Supported versions of TLS

StorageGRID supports TLS 1.2 and TLS 1.3.

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

Related information

Configure tenant accounts and connections

Monitor and audit operations

You can monitor workloads and efficiencies for client operations by viewing transaction trends for the entire grid, or for specific nodes. You can use audit messages to monitor client operations and transactions.

Monitor object ingest and retrieval rates

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

Steps

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

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

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

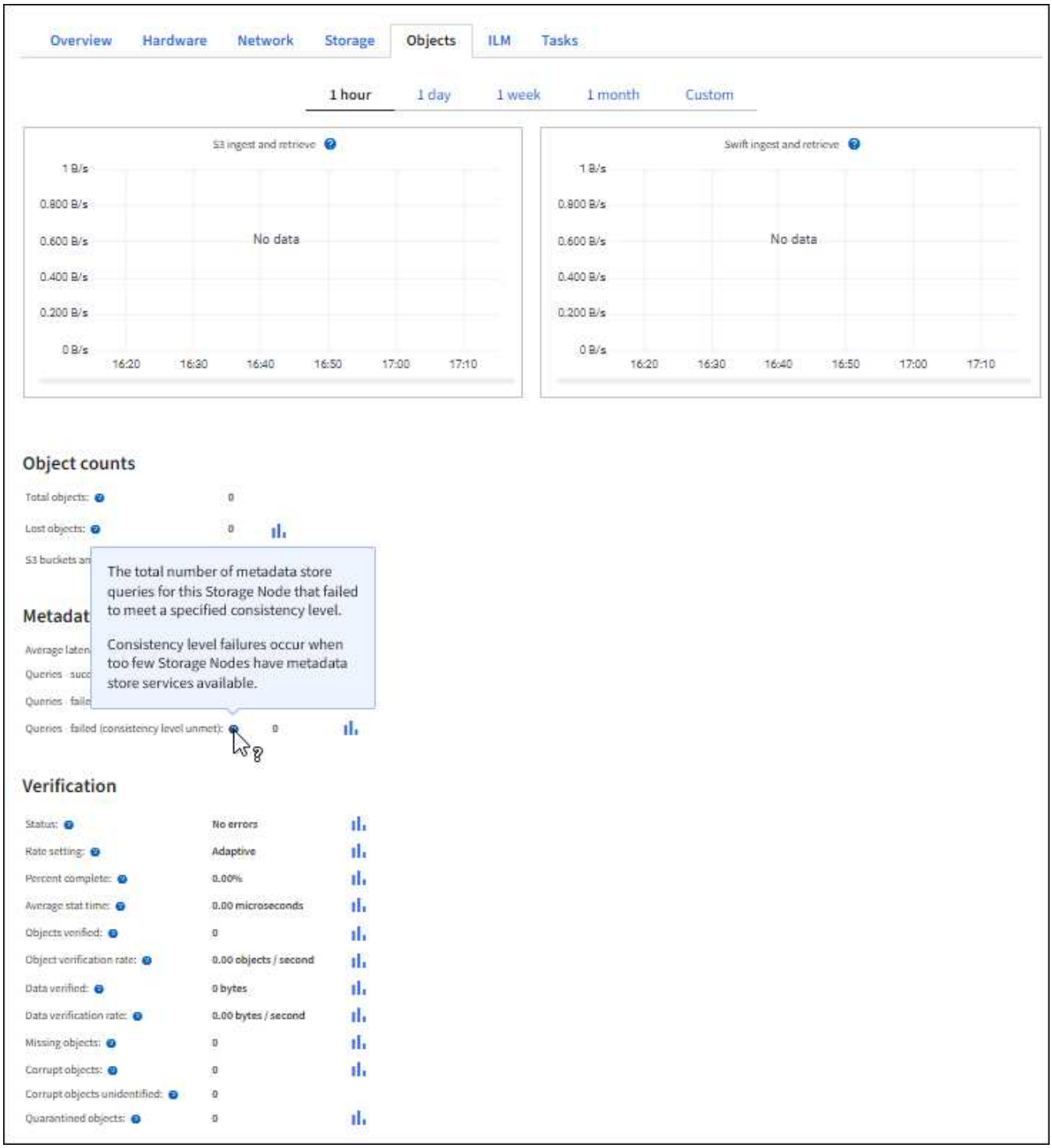
   The charts show trends for all client traffic directed to load balancer endpoints within the grid. You can select a time interval in hours, days, weeks, months, or years, or you can apply a custom interval.

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

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

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

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



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

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

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

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

Access and review audit logs

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

Before you begin

- You must have specific access permissions.
- You must have the `Passwords.txt` 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@primary_Admin_Node_IP`
   b. Enter the password listed in the `Passwords.txt` file.
   c. Enter the following command to switch to root: `su -`
   d. Enter the password listed in the `Passwords.txt` file.

      When you are logged in as root, the prompt changes from `$` to `#`.
2. Go to the directory containing the audit log files:`cd /var/local/audit/export`
3. View the current or a saved audit log file, as required.

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 aren't logged, nor are info, auth, or OPTIONS requests.

Information is tracked for the following Swift operations.

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

Copyright information

Trademark information