StorageGRID® Webscale 11.0

S3 (Simple Storage Service) Implementation Guide

January 2018 | 215-12408_B0
doccomments@netapp.com

NetApp®

Contents

Support for the S3 REST API 5
Changes to S3 REST API support 5
Supported versions 5
Support for StorageGRID Webscale platform services 6
How client applications use HTTP connections 7
S3 tenant accounts in StorageGRID Webscale 7
Specifying S3 API endpoint domain names 7
Identifying IP addresses for API Gateway Nodes and Storage Nodes 9
Port numbers for API Gateway Nodes and Storage Nodes 9
Testing your S3 REST API configuration 9
How StorageGRID Webscale implements the S3 REST API 11
How StorageGRID Webscale ILM rules manage objects 12
Object versioning 13
S3 REST API supported operations and limitations 15
Error responses 15
Date handling 16
Common request headers 17
Common response headers 17
Authenticating requests 17
Operations on the service 18
Operations on buckets 18
Custom operations on buckets 23
Operations on objects 24
Operations for multipart uploads 29
Operations tracked in the audit logs 34
StorageGRID Webscale S3 REST API operations 36
GET Bucket consistency request 36
PUT Bucket consistency request 37
GET Storage Usage request 38
GET Bucket last access time request 40
PUT Bucket last access time request 41
DELETE Bucket metadata notification configuration request 42
GET Bucket metadata notification configuration request 42
PUT Bucket metadata notification configuration request 45
Object metadata included in metadata notifications 48
JSON generated by the search integration service 48
Configuring security for the REST API 49
How the StorageGRID Webscale system implements security for the REST API ... 49
Bucket and group access policies 50
Policy examples 60
How client applications use certificates for security with REST APIs 64

Supported hashing and encryption algorithms for TLS libraries 64
Write-once-read-many (WORM) protection 65
Monitoring and auditing operations 67
Viewing transactions for S3 objects 67
Accessing and reviewing audit logs 67
Benefits of active, idle, and concurrent HTTP connections 68
Benefits of different types of HTTP connections 68
Benefits of keeping idle HTTP connections open 68
Benefits of active HTTP connections 68
Benefits of concurrent HTTP connections 69
Separation of HTTP connection pools for read and write operations 70
Copyright information 71
Trademark information 72
How to send comments about documentation and receive update notifications 73
Index 74

Support for the S3 REST API

The StorageGRID Webscale system supports the storage and retrieval of objects from client applications that interface with the StorageGRID Webscale system by using the Simple Storage Service (S3) Representational State Transfer Application Programming Interface (REST API).

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 Webscale 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 Webscale system support for the S3 REST API.

The following table lists changes to StorageGRID Webscale system support for the S3 REST API:

Date	Release	Comments
September 2014	10.0	Initial support of the S3 REST API by the StorageGRID Webscale system. The currently supported version of the *Simple Storage Service API Reference* is 2006-03-01.
April 2015	10.1	Added support for multipart upload, virtual hosted-style requests, and v4 authentication.
December 2015	10.2	Added support for group and bucket access policies, and for multipart copy (Upload Part - Copy).
June 2016	10.3	Added support for versioning.
April 2017	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.
October 2017	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.

Supported versions

StorageGRID Webscale supports the following specific versions of S3 and HTTP.

Item	Version
S3 specification	*Simple Storage Service API Reference* 2006-03-01
HTTP	1.1 For more information about HTTP, see HTTP/1.1 (RFC 2616).

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 Webscale platform services

StorageGRID Webscale platform services enable you 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.

Attention: StorageGRID Webscale 11.0 includes the initial release of platform services. CloudMirror replication, notifications, and search integration are currently appropriate only for specific situations and workloads. You must contact your NetApp representative if you want to use the initial release of these services.

The *Tenant Administrator Guide* includes a complete description of platform services, including instructions on how to create an endpoint that represents the remote service in your tenant account, which is required before a service can be configured. This guide discusses the StorageGRID Webscale implementation of the configuration APIs.

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

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

A grid administrator must enable the use of platform services for a tenant account before they can be used, as described in the *Administrator Guide*.

Related concepts

Operations on buckets on page 18

Related references

PUT Bucket metadata notification configuration request on page 45

Related information

Administering tenant accounts
Administering StorageGRID Webscale

How client applications use HTTP connections

Client applications use HTTP connections to access and communicate with the StorageGRID Webscale system.

Client applications connect directly to an API Gateway Node or Storage Node to store and retrieve objects. To load balance ingests across Storage Nodes, you can connect to an API Gateway Node, which handles the load balancing for you. Otherwise, you can connect directly to a Storage Node.

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

Client applications can issue "OPTIONS /" HTTPS requests to the S3 port on a Storage Node, without providing S3 authentication credentials, to determine whether the LDR Service is available. You can use this request for monitoring, or to allow external load balancers to identify when a Storage Node is down.

Setting up the connection to client applications involves the following tasks:

- Creating an S3 tenant account
- Identifying IP addresses for API Gateway Nodes and Storage Nodes
- Identifying S3 port numbers for API Gateway Nodes and Storage Nodes
- Copying the system's certificate authority (CA) certificate for client applications that require server validation

Related information

Administering StorageGRID Webscale

S3 tenant accounts in StorageGRID Webscale

You can configure the StorageGRID Webscale system to accept connections from client applications originally developed to use S3 web services by creating an S3 tenant account.

S3 tenant accounts are created and deleted using the **Tenants** menu option for the Grid Management Interface or by using the management API. S3 groups and users, including S3 access keys, S3 bucket settings, and S3 group policies, are managed using the Tenant Management Interface or by using the Tenant API. When you configure an S3 client, you must provide the S3 account information, which is used in the authentication process. For more information, see the *Administrator Guide* and the *Tenant Administrator Guide*.

Related information

Administering StorageGRID Webscale

Administering tenant accounts

Specifying S3 API endpoint domain names

To support S3 virtual-hosted style requests, you must configure the list of endpoint domain names that S3 clients will be connecting to.

Before you begin

- You must be signed in to the Grid Management Interface using a supported browser.

- To perform this task, you need specific access permissions. For details, see information about controlling system access with administration user accounts and groups.

About this task

API endpoint domain names are configured after you create the fully-qualified domain names on the DNS server, depending on the grid nodes that S3 clients will be connecting to:

- If S3 clients are connecting to one or more API Gateway Nodes, you must include the domain name of each API Gateway Node.
- If S3 clients are connecting to one or more Storage Nodes, you must include the domain name of each Storage Node.
- If S3 clients are connecting through an external load balancer, you must include the domain name of the load balancer.

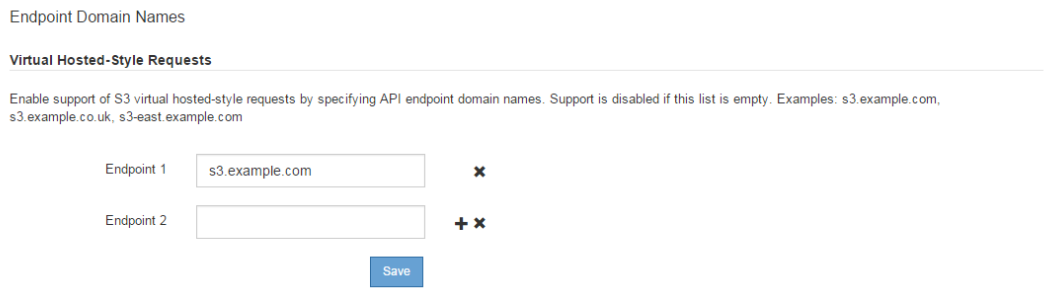
If this list is empty, support for S3 virtual hosted-style requests is disabled.

You must also configure a custom server certificate for the StorageGRID Webscale system with a wildcard Subject Alternative Name (SAN) for each endpoint, and possibly each IP address. These steps are required to validate the SSL certificate and verify the hostname when API client applications connect to the endpoint.

Steps

1. Select **Configuration > Domain Names**.

   The Endpoint Domain Names page appears.



2. Using the (+) icon to add additional fields, enter the list of S3 API endpoint domain names in the **Endpoint *number*** fields and click **Save**.

   Warning: Do not make any changes to the domain name configuration when a grid upgrade is in progress.

Related information

Administering StorageGRID Webscale

Identifying IP addresses for API Gateway Nodes and Storage Nodes

You need the grid node's IP address to connect API client applications to StorageGRID Webscale.

Steps

1. Sign in to the Grid Management Interface using a supported browser.
2. Select **Grid**.
3. In the **Grid Topology** tree, locate and expand the Storage Node or API Gateway Node to which you want to connect.

   The services for the selected grid node appear.
4. In the Storage Node or API Gateway Node, select **SSM > Resources**, and then scroll to the **Network Addresses** table.

   You can establish HTTPS connections from API client applications to any of the listed IP addresses.

Port numbers for API Gateway Nodes and Storage Nodes

API Gateway Nodes and Storage Nodes are only available for HTTP connections from client applications to the StorageGRID Webscale system on specific port numbers.

The following ports are used for client applications that interface to the StorageGRID Webscale system through S3:

Grid node	Port number
API Gateway Node (CLB S3 Port)	8082
Storage Node (LDR S3 Port)	18082

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 Webscale system.

Steps

1. Configure the Amazon Web Services settings to use the account you created in the StorageGRID Webscale 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 Webscale 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.

To manage these objects, the StorageGRID Webscale system uses information lifecycle management (ILM) rules.

For information about ILM rules, see the *Administrator Guide*.

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 Webscale system completes a given request, and not on when S3 clients begin an operation.

Consistency guarantees and controls

By default, StorageGRID Webscale 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.

StorageGRID Webscale allows you to control the consistency control used for each bucket or API request. You can change the consistency control to 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.

For information on configuring consistency controls for buckets, see *GET Bucket consistency request* on page 36 and *PUT Bucket consistency request* on page 37.

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: Sat, 29 Nov 2015 01:02:17 GMT
Authorization: AWS 9MOYPG9ACWPAJA1S72R5:jUGbYkLdBApjCWBgK4TxvOjfock=
Host: test.com
Consistency-Control: strong-site
```

You can set the `Consistency-Control` header to one of the following values:

- **all**: Provides the highest guarantee of read-after-write consistency. 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.
- **default** (read-after-write for new): 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 attempts HEAD operations on keys that do not exist, set the consistency level to **available**, unless you require AWS S3 consistency guarantees. Otherwise, a high number of 500 Internal Server errors can result if one or more Storage Nodes are unavailable.

- **available** (eventual consistency for HEAD operations): Behaves the same as the **default** consistency level, but only provides eventual consistency for HEAD operations. Offers higher availability for HEAD operations than **default** 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.

The following behavior is applied to operations on the bucket:

- Consistency controls only affect objects within a bucket. Setting the consistency control for a bucket does not impact the bucket itself; instead, it sets the consistency control for the S3 operations performed on the objects in the bucket.

  Note: In general, you should use the **default** 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.

- The default behavior of the S3 HEAD operation is to start with a low consistency. If the result of the first HEAD attempt is Not Found, the grid retries the operation with a higher consistency on each retry, until consistency control value **all** is used. If you use the HEAD operation on a key that does not exist, retries will eventually “bubble up” to the **all** consistency control, which requires all nodes to be online. In this case, a high number of 500 Internal Server errors can result if one or more Storage Nodes are unavailable. To prevent these errors, change the consistency control to **available**. This will stop the "bubbling up" of consistency behavior for HEAD operations, although read-after-write consistency on newly created object metadata will no longer be guaranteed.
- The “bubbling up” of consistency behavior also applies to GET operations. However, PUT and DELETE operations do not follow this pattern; they either succeed or fail at the specified consistency control.

Note: You must use the same StorageGRID Webscale `Consistency-Control` header 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.

Related information

Administering StorageGRID Webscale

How StorageGRID Webscale ILM rules manage objects

You can create information lifecycle management (ILM) rules that manage object data ingested into the StorageGRID Webscale system from S3 REST API client applications. You use these ILM rules 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 Webscale 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.

You can set up ILM rules to filter and select objects. For objects ingested using S3, you 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 Webscale 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 Management Interface or the Tenant Management API. When enabling last access time updates, be aware that StorageGRID Webscale performance might be reduced, especially in systems with small objects. See the *Tenant Administrator Guide* for more information.

- Location Constraint
- Object Size
- User Metadata
- Object Tag

For details about ILM rules, see the *Administrator Guide*.

Related information

Administering tenant accounts
Administering StorageGRID Webscale

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 Webscale system implements versioning with support for most features, and with some limitations.

In StorageGRID Webscale, 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 Webscale system.

Multi-factor authentication (MFA) is not supported.

Note: Versioning can be enabled only on buckets created with StorageGRID Webscale 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 that object is updated, this causes its previous versions to become noncurrent. Using a noncurrent time filter allows you to create policies that reduce the storage impact of previous versions of objects.

S3 REST API supported operations and limitations

The StorageGRID Webscale 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 Webscale system supports both virtual hosted-style requests and path-style requests.

Related information

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

Error responses

The StorageGRID Webscale system supports all standard S3 REST API error responses that apply. In addition, the StorageGRID Webscale implementation adds the custom `XNotImplemented` and `NoSuchEndpoint` 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
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
InvalidStorageClass	400 Bad Request
InvalidTag	400 Bad Request
InvalidURI	400 Bad Request
KeyTooLong	400 Bad Request
MalformedXML	400 Bad Request

Name	HTTP Status
MetadataTooLarge	400 Bad Request
MethodNotAllowed	405 Method Not Allowed
MissingContentLength	411 Length Required
MissingRequestBodyError	400 Bad Request
MissingSecurityHeader	400 Bad Request
NoSuchBucket	404 Not Found
NoSuchKey	404 Not Found
NoSuchUpload	404 Not Found
NotImplemented	501 Not Implemented
NoSuchBucketPolicy	404 Not Found
PreconditionFailed	412 Precondition Failed
RequestTimeTooSkewed	403 Forbidden
ServiceUnavailable	503 Service Unavailable
SignatureDoesNotMatch	403 Forbidden
TooManyBuckets	400 Bad Request
UserKeyMustBeSpecified	400 Bad Request

StorageGRID Webscale specific error codes

Error code	Description	HTTP status code
NoSuchEndpoint	The endpoint you are trying to use to configure Bucket replication or Bucket metadata notifications does not exist.	404 Not Found
XNotImplemented	The request you provided implies functionality that is not implemented.	501 Not Implemented

Date handling

The StorageGRID Webscale implementation of the S3 REST API only supports valid HTTP date formats.

The StorageGRID Webscale 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 Webscale system supports common request headers defined by the *Simple Storage Service API Reference*, with a couple of exceptions.

The following list provides request headers that are not supported:

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.

Common response headers

The StorageGRID Webscale system supports common response headers, with some exceptions.

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

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

Authenticating requests

The StorageGRID Webscale 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 Webscale 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 following operations on the service are supported by the StorageGRID Webscale system:

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.
OPTIONS /	Client applications can issue requests to the S3 port on a Storage Node, without providing S3 authentication credentials, to determine whether the LDR Service is available.

Operations on buckets

The StorageGRID Webscale system supports a maximum of 1000 buckets per 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 Webscale consistency controls. For information on using the `Consistency-Control` header, see *How StorageGRID Webscale implements the S3 REST API* on page 11.

You can check whether updates to last access time are enabled or disabled for individual buckets. For more information, see *GET Bucket last access time request* on page 40.

For other bucket operations, see "Custom operations on buckets."

The following table describes which operations on buckets are implemented, and how they are implemented, in the StorageGRID Webscale system.

Operation	Implementation
DELETE Bucket	Implemented with all Amazon S3 REST API behavior.
DELETE Bucket policy	If the necessary access credentials are provided for the account, this operation deletes the policy attached to the bucket. For information about the policy language supported by the StorageGRID Webscale system, see *Bucket and group access policies* on page 50.
DELETE Bucket replication	If the necessary access credentials are provided for the account, the operation deletes the replication configuration attached to the bucket.

Operation	Implementation
GET Bucket (List Objects)	If the necessary access credentials are provided for the account, this operation returns some or all (up to 1,000) of the objects in a bucket. The Storage Class for objects is always listed as STANDARD, even when the object was ingested with the REDUCED_REDUNDANCY storage class specified. When an object is ingested into StorageGRID Webscale with the REDUCED_REDUNDANCY storage class, it means that the object is ingested using a single-commit ingest operation. It does not result in the object being stored at lower levels of redundancy in the StorageGRID Webscale system. **Caution:** Be careful when ingesting objects using REDUCED_REDUNDANCY to create only a single initial copy of the object data. If the single copy is created on a Storage Node that fails, and ILM is not yet satisfied, the result is unrecoverable loss of data. Requests that traverse a large number of keys require special handling. The request might return a truncated response or an empty response to avoid timing out. To get the complete set of results, you need to continue making requests while updating the **`marker`** parameter, as you normally do with a truncated result. Always use **`NextMarker`** if it is present. In some cases, the StorageGRID Webscale implementation of the S3 REST API returns a **`NextMarker`**, when the Amazon S3 REST API would not, because it is a better marker than the last key returned.
GET Bucket acl	If the necessary access credentials are provided for the account, 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 location	If the necessary access credentials are provided for the account, this operation will return 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.
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	If the necessary access credentials are provided for the account, this operation returns the notification configuration attached to the bucket.
GET Bucket policy	If the necessary access credentials are provided for the account, this operation returns the policy attached to the bucket. For information about the policy language supported by the StorageGRID Webscale system, see *Bucket and group access policies* on page 50.
GET Bucket replication	If the necessary access credentials are provided for the account, this operation returns the replication configuration attached to the bucket.

Operation	Implementation
GET Bucket versioning	This implementation uses the `versioning` subresource to return the versioning state of a bucket. To retrieve the versioning state of a bucket, you must be the bucket owner. The versioning state returned indicates if the bucket is "Unversioned" or if the bucket is version "Enabled" or "Suspended".
HEAD Bucket	If the necessary access credentials are provided for the account, this operation determines if a bucket exists and you have permission to access it.
PUT Bucket	If the necessary access credentials are provided for the account, this operation creates a new bucket. By creating the bucket, you become the bucket owner. By default, buckets are created in the us-east-1 region. To specify a different region, use the LocationConstraint request element, and specify the exact name of a region that has been defined using the StorageGRID Webscale Grid Management Interface or the 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 Webscale.

Operation	Implementation
PUT Bucket notification	If the necessary access credentials are provided for the account, this operation configures notifications for the bucket using the notification configuration XML included in the request body. **Attention:** StorageGRID Webscale 11.0 includes the initial release of platform services. CloudMirror replication, notifications, and search integration are currently appropriate only for specific situations and workloads. You must contact your NetApp representative if you want to use the initial release of these services. You should be aware of the following implementation details: • StorageGRID Webscale 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 Webscale endpoint. Endpoints can be created using the Tenant Management Interface 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` Consult the *Tenant Administrator Guide* for more information on implementing notifications on S3 buckets.
PUT Bucket policy	If the necessary access credentials are provided for the account, this operation sets the policy attached to the bucket. For information about the policy language supported by the StorageGRID Webscale system, see *Bucket and group access policies* on page 50.

Operation	Implementation
PUT Bucket replication	If the necessary access credentials are provided for the account, this operation configures StorageGRID Webscale CloudMirror replication for the bucket using the replication configuration XML provided in the request body. **Attention:** StorageGRID Webscale 11.0 includes the initial release of platform services. CloudMirror replication, notifications, and search integration are currently appropriate only for specific situations and workloads. You must contact your NetApp representative if you want to use the initial release of these services. 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 Webscale endpoints as specified in the Tenant Management Interfaceor the Tenant Management API. The endpoint must exist for replication configuration to succeed. If the endpoint does not exist, a `404 Not Found` message is returned with the code `NoSuchEndpoint`. • You do not need to specify a `Role` or `StorageClass` in the configuration XML, as these values are not used by StorageGRID Webscale and will be ignored if submitted. StorageGRID Webscale does not need a role defined to enable it to store replicated objects to the destination bucket, and uses the `STANDARD` storage class by default. Consult the *Tenant Administrator Guide* for more information on using bucket replication to implement StorageGRID Webscale CloudMirror replication.
PUT Bucket versioning	This implementation uses the `versioning` subresource to set the versioning state of an existing bucket. To set the versioning state, you must be the bucket owner. 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 information

Administering tenant accounts

Custom operations on buckets

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

The following table describes which custom operations on buckets are implemented, and how they are implemented, in the StorageGRID Webscale system.

Operation	Implementation
GET Bucket consistency request	If the necessary access credentials are provided for the account, this operation returns the consistency level being applied to a particular bucket. For more information, see *GET Bucket consistency request* on page 36.
PUT Bucket consistency request	If the necessary access credentials are provided for the account, this operation sets the consistency level applied to a particular bucket. For more information, see *PUT Bucket consistency request* on page 37.
GET Bucket last access time request	If the necessary access credentials are provided for the account, this operation returns whether last access time updates are enabled or disabled for a particular bucket. For more information, see *GET Bucket last access time request* on page 40.
PUT Bucket last access time request	If the necessary access credentials are provided for the account, this operation allows you to enable or disable last access time updates for a particular bucket. Last access time is disabled as the default setting for all buckets created with version 10.3.0, or later. For more information, see *PUT Bucket last access time request* on page 41.
DELETE Bucket metadata notification configuration	If the necessary access credentials are provided for the account, this operation deletes the metadata notification configuration XML associated with a particular bucket. For more information, see *DELETE Bucket metadata notification configuration request* on page 42.
GET Bucket metadata notification configuration	If the necessary access credentials are provided for the account, this operation returns the metadata notification configuration XML associated with a particular bucket. For more information, see *GET Bucket metadata notification configuration request* on page 42.

Operation	Implementation
PUT Bucket metadata notification configuration	If the necessary access credentials are provided for the account, this operation configures the metadata notification service for a bucket using the configuration XML included in the request body. See the *Tenant Administrator Guide* for more information on implementing the search integration service. For more information on the PUT metadata notification configuration API, see *PUT Bucket metadata notification configuration request* on page 45.

Related information

Administering tenant accounts

Operations on objects

Data objects ingested to the StorageGRID Webscale system through Swift cannot be accessed through S3.

All of the operations on objects, except GET Object ACL and OPTIONS /, support StorageGRID Webscale consistency controls. For information on using the `Consistency-Control` header, see *How StorageGRID Webscale implements the S3 REST API* on page 11.

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 Webscale system completes a given request, and not on when S3 clients begin an operation.

All objects in a StorageGRID Webscale bucket are owned by the bucket owner, including objects created by an anonymous user, or by another account.

The following operations on objects are supported by the StorageGRID Webscale system.

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. Unlike the PUT Object operation, the DELETE Multiple Objects operation does not support chunked transfer encoding and the content encoding gzip attributes.
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	The following request headers are not supported and return `XNotImplemented`: • `x-amz-restore` • `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`.
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`.

Operation	Implementation
HEAD Object	The following request headers are not supported and return `XNotImplemented`: • `x-amz-restore` • `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`.
PUT Object	**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 Webscale system completes a given request, and not on when S3 clients begin an operation. **Object size** StorageGRID Webscale supports objects up to 5 TB in size. **Object ownership** In StorageGRID Webscale, all objects are owned by the bucket owner account, including objects created by a non-owner account or an anonymous user.
PUT Object (*continued*)	**Storage class options** 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:** The `REDUCED_REDUNDANCY` storage class is an option used to limit redundant storage for data that is better replicated elsewhere, such as with ILM policies. Therefore, specifying the `REDUCED_REDUNDANCY` value does not affect the specified ILM policy, and it does not result in data being stored at lower levels of redundancy in the StorageGRID Webscale system. **Caution:** Be careful when ingesting objects using REDUCED_REDUNDANCY to create only a single initial copy of the object data. If the single copy is created on a Storage Node that fails, and ILM is not yet satisfied, the result is unrecoverable loss of data.

Operation	Implementation
PUT Object (*continued*)	**Request headers** The following request headers are supported: • `x-amz-tagging` • `x-amz-server-side-encryption` • `x-amz-meta-` name-value pairs for user-defined metadata To record the object creation time, so that you can use the User Defined Creation Time option for the reference time in an ILM rule, you need to store the value in a user-defined header named `x-amz-meta-creation-time`. For example: `x-amz-meta-creation-time`=1443399726. This field is evaluated as seconds since Jan 1, 1970. For more information, see "Reference time" in the *Administrator Guide*. The following request headers are supported only with the following values: • `Transfer-Encoding:chunked` • `Content-Encoding:aws-chunked` **Note:** Submitting other values for `Content-Encoding` can lead to unexpected results, or failures due to unrecognized coding schemes or MD5 verification failures. The following request headers are not supported: • `Expires` • `x-amz-acl` 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`
PUT Object (*continued*)	**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. For more information on versioning, see the "PUT Bucket versioning" and "GET Bucket versioning" entries in *Operations on buckets* on page 18.

Operation	Implementation
PUT Object - Copy	**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 Webscale system completes a given request, and not on when S3 clients begin an operation. **Request headers** The following request headers are supported: • `x-amz-meta-` name-value pairs for user-defined metadata To record the object creation time, so that you can use the User Defined Creation Time option for the reference time in an ILM rule, you need to store the value in a user-defined header named `x-amz-meta-creation-time`. For example: `x-amz-meta-creation-time`=1443399726. This field is evaluated as seconds since Jan 1, 1970. For more information, see "Reference time" in the *Administrator Guide*. • `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.

Operation	Implementation
PUT Object - Copy (*continued*)	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`.
PUT Object - Copy (*continued*)	**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.
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 Webscale 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`.

Operations for multipart uploads

Note: 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.

Note: Each multipart part except the last must be between 5 MB and 5 GB. The last part can be less than 5 MB. This client must follow these limits as it is not enforced by StorageGRID Webscale.

All of the multipart upload operations support StorageGRID Webscale consistency controls. For information on using the `Consistency-Control` header, see *How StorageGRID Webscale implements the S3 REST API* on page 11.

Operation	Implementation
List Multipart Uploads	The following request headers 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).

Operation	Implementation
Initiate Multipart Upload	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:** The `REDUCED_REDUNDANCY` storage class is an option used to limit redundant storage for data that is better replicated elsewhere, such as using ILM policies. Therefore, specifying the `REDUCED_REDUNDANCY` value does not affect the specified ILM policy, and does not result in data being stored at lower levels of redundancy in the StorageGRID Webscale system. **Caution:** Be careful when ingesting objects using REDUCED_REDUNDANCY to create only a single initial copy of the object data. If the single copy is created on a Storage Node that fails, and ILM is not yet satisfied, the result is unrecoverable loss of data. The following request headers are supported: • `Content-Type` • `x-amz-meta-` name-value pairs for user-defined metadata To record the object creation time, so that you can use the User Defined Creation Time option for the reference time in an ILM rule, you need to store the value in a user-defined header named `x-amz-meta-creation-time`. For example: `x-amz-meta-creation-time=1443399726`. This field is evaluated as seconds since Jan 1, 1970. For more information, see "Reference time" in the *Administrator Guide*. The `x-amz-server-side-encryption` header is not supported directly for Initiate Multipart Upload requests. If you require server-side encryption for a multipart upload, you need to specify the `x-amz-server-side-encryption` header for each of the upload parts, but you cannot specify it as part of the Initiate Multipart Upload header or the request fails. The following request headers are not supported and return `XNotImplemented`: • `x-amz-website-redirect-location` • `x-amz-server-side-encryption-customer-key` **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).

Operation	Implementation
Upload Part	The following request headers are supported: • `Content-Length` • `x-amz-server-side-encryption` If you need to specify server-side encryption for a multipart upload, you need to specify the `x-amz-server-side-encryption` header for each of the individual upload parts. 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	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 Webscale 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).

Operation	Implementation
Complete Multipart Upload	**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 Webscale system completes a given request, and not on when S3 clients begin an operation. **Request headers** The `x-amz-storage-class` request header is supported with the following enumerated values: • `STANDARD` Default. Specifies a dual-commit ingest operation. • `REDUCED_REDUNDANCY` Specifies a single-commit ingest operation. **Note:** The `REDUCED_REDUNDANCY` storage class is an option used to limit redundant storage for data that is better replicated elsewhere, such as with ILM policies. Therefore, specifying the `REDUCED_REDUNDANCY` value does not affect the specified ILM policy, and it does not result in data being stored at lower levels of redundancy in the StorageGRID Webscale system. **Caution:** Be very careful when configuring ILM policies for data configured with `REDUCED_REDUNDANCY` to replicate only a single copy of object data before the ILM policy is satisfied. If the ILM policy is not yet satisfied, and the single copy is created on a Storage Node that fails, the result is unrecoverable loss of data. **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.

Operation	Implementation
Complete Multipart Upload (continued)	**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 Management Interface on Total Events (SMTT). The Last Event message at **Grid** > ***site*** > ***Storage Node*** > **SSM** > **Events** displays " Failed to publish notifications for *`bucket-name object key`* " for the last object whose notification failed. 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. They can resubmit the existing values to avoid making unwanted changes.
Abort Multipart Upload	Implemented with all Amazon S3 REST API behavior.
List Parts	Implemented with all Amazon S3 REST API behavior.

Related information

Administering StorageGRID Webscale

Operations tracked in the audit logs

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

Bucket operations tracked in the audit logs
DELETE Bucket
DELETE Multiple Objects
GET Bucket (List Objects)

Bucket operations tracked in the audit logs
GET Bucket Object versions
HEAD Bucket
PUT Bucket
PUT Bucket Versioning

Object operations tracked in the audit logs
Complete Multipart Upload
DELETE Object
GET Object
HEAD Object
PUT Object
PUT Object - Copy

StorageGRID Webscale S3 REST API operations

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

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. For more information on the `Consistency-Control` header, see information about how StorageGRID Webscale implements the S3 REST API.

You must have the s3:GetBucketConsistency permission to complete this operation.

Request

Request HTTP Header	Description
Authorization	Specifies the AWS signature and the access key ID for the account to use for the request.
Date	The date and time of the request.
Host	The host name to which the request is directed.

Request example

```
GET /bucket?x-ntap-sg-consistency HTTP/1.1
Date: Sat, 29 Nov 2015 01:02:17 GMT
Authorization: AWS 9MOYPG9ACWPAJA1S72R5:jUGbYkLdBApjCWBgK4TxvOjfock=
Host: test.com
```

Response

Response HTTP Header	Description
Connection	Specifies whether the connection to the server is open or closed.
Content-Length	The length of the response body.
Content-Type	The Multipurpose Internet Mail Extensions (MIME) type of the response body.
Date	The date and time of the response.
Server	The server that created the response.
x-amz-request-id	The identifier that uniquely identifies the request. Created by the S3 API.

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

- **all**: Provides the highest guarantee of read-after-write consistency. 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.
- **default** (read-after-write for new): 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 attempts HEAD operations on keys that do not exist, set the consistency level to **available**, unless you require AWS S3 consistency guarantees. Otherwise, a high number of 500 Internal Server errors can result if one or more Storage Nodes are unavailable.

- **available** (eventual consistency for HEAD operations): Behaves the same as the **default** consistency level, but only provides eventual consistency for HEAD operations. Offers higher availability for HEAD operations than **default** 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

How StorageGRID Webscale implements the S3 REST API on page 11

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. For more information on the `Consistency-Control` header, see information about how StorageGRID Webscale implements the S3 REST API.

You must have the s3:PutBucketConsistency permission to complete this operation.

Request

Request HTTP Header	Description
Authorization	Specifies the AWS signature and the access key ID for the account to use for the request.
Date	The date and time of the request.

Request HTTP Header	Description
Host	The host name to which the request is directed.

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

- **all**: Provides the highest guarantee of read-after-write consistency. 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.
- **default** (read-after-write for new): 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 attempts HEAD operations on keys that do not exist, set the consistency level to **available**, unless you require AWS S3 consistency guarantees. Otherwise, a high number of 500 Internal Server errors can result if one or more Storage Nodes are unavailable.

- **available** (eventual consistency for HEAD operations): Behaves the same as the **default** consistency level, but only provides eventual consistency for HEAD operations. Offers higher availability for HEAD operations than **default** 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 **default** 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: Sat, 29 Nov 2015 01:02:17 GMT
Authorization: AWS 9MOYPG9ACWPAJA1S72R5:jUGbYkLdBApjCWBgK4TxvOjfock=
Host: test.com
```

Related concepts

How StorageGRID Webscale implements the S3 REST API on page 11

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 to complete this operation.

Request

Request HTTP Header	Description
Authorization	Specifies the AWS signature and the access key ID for the account to use for the request.
Date	The date and time of the request.
Host	The host name to which the request is directed.

Request example

```
GET /?x-ntap-sg-usage HTTP/1.1
Date: Sat, 29 Nov 2015 00:49:04 GMTT
Authorization: AWS 9MOYPG9ACWPAJA1S72R5:jUGbYkLdBApjCWBgK4TxvOjfock=
Host: test.com
```

Response

Response HTTP Header	Description
Connection	Specifies whether the connection to the server is open or closed.
Content-Length	The length of the response body.
Content-Type	The Multipurpose Internet Mail Extensions (MIME) type of the response body.
Date	The date and time of the response.
Server	The server that created the response.
x-amz-request-id	The identifier that uniquely identifies the request. Created by the S3 API.

Response example

```
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.

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 to complete this operation.

Request

Request HTTP Header	Description
Authorization	Specifies the AWS signature and the access key ID for the account to use for the request.
Date	The date and time of the request.
Host	The host name to which the request is directed.

Request example

```
GET /bucket?x-ntap-sg-lastaccesstime HTTP/1.1
Date: Sat, 29 Nov 2015 01:02:17 GMT
Authorization: AWS 9MOYPG9ACWPAJA1S72R5:jUGbYkLdBApjCWBgK4TxvOjfock=
Host: test.com
```

Response

Response HTTP Header	Description
Connection	Specifies whether the connection to the server is open or closed.
Content-Length	The length of the response body.
Content-Type	The Multipurpose Internet Mail Extensions (MIME) type of the response body.
Date	The date and time of the response.
Server	The server that created the response.
x-amz-request-id	The identifier that uniquely identifies the request. Created by the S3 API.

Response example

The value of `<LastAccessTime>` will either be enabled or disabled.

```
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 to enable or disable the last access time configuration settings for the bucket.

> **Note:** Starting with StorageGRID Webscale 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 Webscale 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 Management Interface, 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

Request HTTP Header	Description
Authorization	Specifies the AWS signature and the access key ID for the account to use for the request.
Date	The date and time of the request.
Host	The host name to which the request is directed.

Request examples

Enabling last access time for a bucket

```
PUT /bucket?x-ntap-sg-lastaccesstime=enabled HTTP/1.1
Date: Sat, 29 Nov 2015 01:02:17 GMT
```

```
Authorization: AWS 9MOYPG9ACWPAJA1S72R5:jUGbYkLdBApjCWBgK4TxvOjfock=
Host: test.com
```

Disabling last access time for a bucket

```
PUT /bucket?x-ntap-sg-lastaccesstime=disabled HTTP/1.1
Date: Sat, 29 Nov 2015 01:02:17 GMT
Authorization: AWS 9MOYPG9ACWPAJA1S72R5:jUGbYkLdBApjCWBgK4TxvOjfock=
Host: test.com
```

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 to delete search integration configuration settings for the bucket.

Request

Request HTTP Header	Description
Authorization	Specifies the AWS signature and the access key ID for the account to use for the request.
Date	The date and time of the request.
Host	The host name to which the request is directed.

Request examples

Disabling the search integration service for a bucket

```
DELETE /test1?x-ntap-sg-metadata-notification HTTP/1.1
Host: example.com
Date: Fri, 21 Jul 2017 18:21:34 +0000
Authorization: AWS QYUTN90RX0RXO70QEGU8:y50RN9wUAYL5BnK+eFci4fz0D7U=
```

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 to complete this operation.

Request

Request HTTP Header	Description
Authorization	Specifies the AWS signature and the access key ID for the account to use for the request.
Date	The date and time of the request.
Host	The host name to which the request is directed.

Request example

This request retrieves the metadata notification configuration for the bucket named “bucket”.

```
GET /bucket?x-ntap-sg-metadata-notification HTTP/1.1
Host: example.com
Date: Thu, 20 Jul 2017 18:25:38 +0000
Authorization: AWS QYUTN90RX0RXO70QEGU8:/XpYXJFVGp5pXh0se26ZzxxkuNE=
```

Response

Response HTTP Header	Description
Connection	Specifies whether the connection to the server is open or closed.
Content-Length	The length of the response body.
Content-Type	The Multipurpose Internet Mail Extensions (MIME) type of the response body.
Date	The date and time of the response.
Server	The server that created the response.
x-amz-request-id	The identifier that uniquely identifies the request. Created by the S3 API.

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 Webscale should send object metadata. Destinations must be specified using the URN of a StorageGRID Webscale endpoint. See the *Tenant Administrator Guide* for more information on configuring endpoints, and the search integration service.

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 Webscale 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 Management Interface 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

Administering 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.

Attention: StorageGRID Webscale 11.0 includes the initial release of platform services. CloudMirror replication, notifications, and search integration are currently appropriate only for specific situations and workloads. You must contact your NetApp representative if you want to use the initial release of these services.

You must have the s3:PutBucketMetadataNotification permission for a bucket to enable or disable the metadata notification configuration for the bucket.

Request

Request HTTP Header	Description
Authorization	Specifies the AWS signature and the access key ID for the account to use for the request.
Date	The date and time of the request.
Host	The host name to which the request is directed.

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 Webscale 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 Webscale endpoint. The endpoint must exist when the metadata notification configuration is submitted, or the request fails with a `404 Not Found` message and the `NoSuchEndpoint` error code.

See the *Tenant Administrator Guide* for more information on configuring endpoints and on the search integration service.

```
<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 Webscale 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 Management Interface 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

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
Host: example.com
Date: Thu, 20 Jul 2017 18:21:34 +0000
Authorization: AWS QYUTN90RX0RXO70QEGU8:y50RN9wUAYL5BnK+eFci4fz0D7U=

<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
Host: example.com
Date: Thu, 20 Jul 2017 18:21:34 +0000
Authorization: AWS QYUTN90RX0RXO70QEGU8:y50RN9wUAYL5BnK+eFci4fz0D7U=

<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

Administering tenant accounts

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	md5	Object hash
	size	Object size (in bytes) as visible to an HTTP client
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

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 sample 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"
  }
}
```

Configuring security for the REST API

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

How the StorageGRID Webscale system implements security for the REST API

The StorageGRID Webscale 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 Webscale system implements security, authentication, and authorization for the REST API.

The StorageGRID Webscale 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 Webscale system.

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

When a StorageGRID Webscale 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 choose to 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 in this instance with a different trusted CA. For more information, see “Configuring certificates” in the *Administrator Guide*.

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.

Security issue	Implementation for REST API
Client authorization	• S3: Bucket ownership and all applicable access control policies • Swift: Account admin role access

Related information

Administering StorageGRID Webscale

Bucket and group access policies

The StorageGRID Webscale system implements a subset of the S3 REST API policy language that you can use to control access to buckets and objects within those buckets.

Overview

StorageGRID Webscale uses the Amazon Web Services (AWS) policy language syntax to allow S3 tenants to create access policies to their data. Access policies for the S3 API are written in JSON. There are two kinds of access policies supported by StorageGRID Webscale:

- **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 Management Interface 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 Webscale 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 Webscale system.

Element	Description
Effect	Use the Effect element to establish whether the specified operations are allowed or denied. You must identify operations you allow (or deny) on buckets or objects using the supported Action element keywords.
Principal/ NotPrincipal	You can allow users, groups, and accounts to access specific resources and perform specific actions. If no S3 signature is included in the request, anonymous access is allowed by specifying the wildcard character (*) as the principal. By default, only the account root has access to resources owned by the account. You only need to specify the Principal element in a bucket policy. For group policies, the group to which the policy is attached is the implicit Principal element.
Resource/ NotResource	The Resource element identifies buckets and objects. You can allow or deny permissions to buckets and objects using the uniform resource name (URN) 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": {
        "SGWS": [
          "urn:sgws:identity::27233906934684427525:federated-group/admin",
          "urn:sgws:identity::27233906934684427525:federated-group/finance"
        ]
      },
      "Action": [
        "s3:ListBucket",
        "s3:GetObject"
      ],
      "Resource": [
```

```
        "urn:sgws:s3:::mybucket",
        "urn:sgws:s3:::mybucket/*"
      ]
    }
  ]
}
```

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

Consistency control settings for policies

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

By default, any updates you make to bucket policies are also eventually consistent. However, 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 URN in policy statements

In policy statements, the URN is used in Principal and Resource elements as follows.

- Use this syntax to specify the S3 resource URN:

```
urn:sgws:s3:::bucket_name
urn:sgws:s3:::bucket_name/object_key
```

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

```
urn:sgws:identity::account_id:root
urn:sgws:identity::account_id:user/user_name
urn:sgws:identity::account_id:group/group_name
urn:sgws:identity::account_id:federated-user/user_name
urn:sgws:identity::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, as outlined in *RFC 2141 URN Syntax*, is not supported.
  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/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 URN.
  For example:

```
"Resource": "urn:sgws:s3:::mybucket/*"
```

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

```
"Resource": "urn:sgws:s3:::mybucket/home/${sgws:username}/*"
```

  See *Specifying variables in a policy* on page 58 for a list of available policy variables.

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

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.

- When building a group policy statement, no Principal element is specified because the group is understood to be the principal.
- Each policy statement must include a Principal element (unless it is a group policy). 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 a URN:

```
"Principal": { "SGWS": "account_id"}
"Principal": { "SGWS": "identity_urn" }
```

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

```
 "Principal": { "SGWS": "27233906934684427525" }
```

- You can specify just the account root:

```
"Principal": { "SGWS": "urn:sgws:identity::
27233906934684427525:root" }
```

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

```
"Principal": { "SGWS": "urn:sgws:identity::
27233906934684427525:federated-user/Bob" }
```

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

```
"Principal": { "SGWS": "urn:sgws:identity::
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:

```
urn:sgws:identity::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.

Table 1: Permissions applicable to buckets

Permissions	S3 REST API operations
s3:CreateBucket	PUT Bucket
s3:DeleteBucket	DELETE Bucket
s3:DeleteBucketMetadataNotification	DELETE Bucket metadata notification configuration
s3:DeleteBucketPolicy	DELETE Bucket policy
s3:GetBucketAcl	GET Bucket ACL
s3:GetBucketConsistency	GET Bucket Consistency
s3:GetBucketLastAccessTime	GET Bucket Last Access Time
s3:GetBucketLocation	GET Bucket location
s3:GetBucketMetadataNotification	GET Bucket metadata notification configuration
s3:GetBucketNotification	GET Bucket notification
s3:GetBucketPolicy	GET Bucket policy
s3:GetBucketReplication	GET Bucket replication
s3:GetBucketVersioning	GET Bucket versioning
s3:ListAllMyBuckets	GET Service, GET Storage Usage
s3:ListBucket	GET Bucket (List Objects), HEAD Bucket
s3:ListBucketMultipartUploads	List Multipart Uploads
s3:ListBucketVersions	GET Bucket versions
s3:PutBucketConsistency	PUT Bucket Consistency
s3:PutBucketLastAccessTime	PUT Bucket Last Access Time

Permissions	S3 REST API operations
s3:PutBucketMetadataNotification	PUT Bucket metadata notification configuration
s3:PutBucketNotification	PUT Bucket notification
s3:PutBucketPolicy	PUT Bucket policy
s3:PutBucketReplication	PUT Bucket replication
s3:PutBucketVersioning	PUT Bucket versioning

Table 2: Permissions applicable to objects

Permissions	S3 REST API operations
s3:AbortMultipartUpload	Abort Multipart Upload
s3:DeleteObject	DELETE Object, DELETE Multiple Objects
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
s3:GetObjectAcl	GET Object ACL
s3:GetObjectTagging	GET Object Tagging
s3:GetObjectVersionTagging	GET Object Tagging (a specific version of the object)
s3:GetObjectVersion	GET Object (a specific version of the object)
s3:ListMultipartUploadParts	List Parts
s3:PutObject	PUT Object, PUT Object - Copy, Initiate Multipart Upload, Complete Multipart Upload, uploading parts (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, Complete Multipart Upload

Using the PutOverwriteObject permission

The PutOverwriteObject permission applies to operations that create or update objects (for example, PUT new objects or PUT Copy to update metadata). The setting of this permission determines whether the client can overwrite an object's data or metadata. Possible settings include Allow (client can overwrite an object) or Deny (client cannot overwrite an object). The default setting is Allow. When this permission is not present, the effect is the same as if Allow were set.

When set to Deny, this permission works as follows.

- If an existing object is found at the same path:

  - Any object's data or metadata cannot be overwritten.
  - Any ingest operations in progress are cancelled, and an error is returned.
  - If S3 versioning is enabled, the Deny permission has no effect, since having versioning enabled does not allow data or metadata modification.
- If an existing object is not found, this permission has no effect.

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

For an example using the PutOverwriteObject permission, see *Example: PutOverwriteObject permission* on page 64.

Specifying conditions in a policy

You can use conditions to allow policies to take effect based on request values.

Condition operators are categorized as follows:

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

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": {
      "sgws:SourceIp": "54.240.143.0/24"
        ...
},
       ...
```

Table 3: Supported condition operators

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).

Condition operators	Description
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.
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.

Table 4: Supported condition keys

Category	Applicable condition keys	Description
IP operators	sgws:SourceIp	Will compare to the IP address from which the request was sent. Can be used for bucket or object operations.
Resource/Identity	sgws:username	Will compare to the sender's username from which the request was sent. Can be used for bucket or object operations.

Category	Applicable condition keys	Description
S3:ListBucket and S3:ListBucketVersions permissions	s3:delimiter	Will compare to the delimiter parameter specified in a GET Bucket or GET Bucket Object versions request.
	s3: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 `${sgws:username}` is part of the Resource element:

```
"Resource": "urn:sgws:s3:::bucket-name/home/${sgws:username}/*"
```

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

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

Variable	Description
${sgws:SourceIp}	Uses the SourceIp key as the provided variable.
${sgws: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 Webscale 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
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.

Related concepts

Write-once-read-many (WORM) protection on page 65

Related information

Administering StorageGRID Webscale

Policy examples

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

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 the bucket and perform GetObject 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": ["urn:sgws:s3:::examplebucket","urn:sgws: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 Webscale, 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": {
        "SGWS": "95390887230002558202"
      },
      "Action": "s3:*",
      "Resource": [
        "urn:sgws:s3:::examplebucket",
        "urn:sgws:s3:::examplebucket/*"
      ]
    },
    {
      "Effect": "Allow",
      "Principal": {
        "SGWS": "31181711887329436680"
      },
      "Action": "s3:GetObject",
      "Resource": "urn:sgws:s3:::examplebucket/shared/*"
    },
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "SGWS": "31181711887329436680"
      },
      "Action": "s3:ListBucket",
      "Resource": "urn:sgws: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 GetObject 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": {
        "SGWS": "urn:sgws:identity::95390887230002558202:federated-group/Marketing"
      },
      "Action": "s3:*",
      "Resource": [
        "urn:sgws:s3:::examplebucket",
        "urn:sgws:s3:::examplebucket/*"
      ]
    },
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": ["s3:ListBucket","s3:GetObject"],
      "Resource": [
        "urn:sgws:s3:::examplebucket",
        "urn:sgws: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.

Note: The ‘Condition’ keyword is only supported in the Tenant Management Interface (StorageGRID Webscale version 10.4 or greater).

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

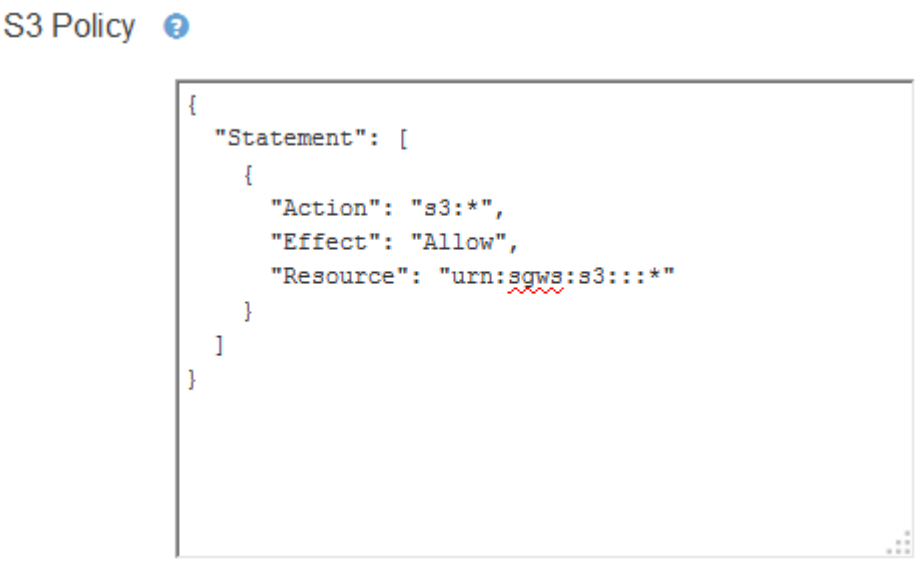
```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "SGWS": "urn:sgws:identity::95390887230002558202:federated-user/Bob"
      },
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "urn:sgws:s3:::examplebucket",
        "urn:sgws:s3:::examplebucket/*"
      ]
    },
    {
      "Effect": "Deny",
      "NotPrincipal": {
        "SGWS": "urn:sgws:identity::95390887230002558202:federated-user/Bob"
      },
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "urn:sgws:s3:::examplebucket",
        "urn:sgws:s3:::examplebucket/*"
      ]
    }
  ]
}
```

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 Management Interface or the API.

Example: Setting the group policy using the Tenant Management Interface

When using the Tenant Management Interface to add or edit a group, you can use the **S3 Policy** dialog box to create and update group policies using valid JSON strings:



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": "urn:sgws:s3:::*"
    }
  ]
}
```

Example: Allow group read-only access to all buckets

In this example, all members of the group are permitted read-only access to all buckets unless explicitly denied by bucket policy. Access to buckets owned by this account would be allowed unless explicitly denied by the target bucket policy.

```
{
  "Statement": [
    {
      "Sid": "AllowGroupReadOnlyAccess",
      "Effect": "Allow",
      "Action": [ "s3:ListAllMyBuckets", "s3:ListBucket", "s3:GetObject" ],
      "Resource": "urn:sgws: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.

Note: The ‘Condition’ keyword and *`sgws:username`* variable are only supported in the Tenant Management Interface (StorageGRID Webscale version 10.4 or later).

```
{
  "Statement": [
    {
      "Sid": "AllowListBucketOfASpecificUserPrefix",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "urn:sgws:s3:::department_bucket",
      "Condition": {
        "StringLike": {
          "s3:prefix": "${sgws:username}/*"
        }
      }
    },
    {
      "Sid": "AllowUserSpecificActionsOnlyInTheSpecificUserPrefix",
      "Effect": "Allow",
      "Action": "s3:*Object",
      "Resource": "urn:sgws:s3:::department_bucket/${sgws:username}/*"
    }
  ]
}
```

Example: PutOverwriteObject permission

In this example, the `Deny` Effect for PutOverwriteObject and DeleteObject protects the object's data and metadata from being deleted or modified.

For more information, see *Using the PutOverwriteObject permission* on page 55 and *Write-once-read-many (WORM) protection* on page 65.

```
{
"Sid": "WORMExamplePolicy",
"Effect": "Deny",
"Action": ["s3:PutOverwriteObject", "s3:DeleteObject"],
"Resource": ["urn:sgws:s3:::*"],
}
```

How client applications use certificates for security with REST APIs

When a client application establishes a TLS session to the StorageGRID Webscale 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 Webscale 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 Webscale system over a network connection that uses Transport Layer Security (TLS). The StorageGRID Webscale 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 Webscale system supports the following cipher suite security algorithms:

TLS version	Cipher suite	Benefit
v1.0	TLS_RSA_WITH_AES_128_CBC_SHA	Provide secure encryption and efficient processing of objects.
	TLS_RSA_WITH_AES_256_CBC_SHA	
v1.1	TLS_RSA_WITH_AES_128_CBC_SHA	
	TLS_RSA_WITH_AES_256_CBC_SHA	
v1.2	TLS_RSA_WITH_AES_128_CBC_SHA	
	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_DHE_RSA_WITH_AES_128_GCM_SHA256	Support perfect forward secrecy.
	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	
	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 CLB or LDR.

Write-once-read-many (WORM) protection

You can create write-once-read-many (WORM) buckets to protect data and metadata. 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 Management Interface, set the Prevent Client Modify global 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. For more information, see the *Administrator Guide*.

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 (does not guard against)

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

Situation B — Sequential completed overwrites (guards against)

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

For an example using the PutOverwriteObject permission, see *Example: PutOverwriteObject permission* on page 64.

Related concepts

Bucket and group access policies on page 50
How StorageGRID Webscale ILM rules manage objects on page 12
Policy examples on page 60

Monitoring and auditing operations

You can monitor the health of your client application connections to the StorageGRID Webscale system by viewing summary attributes that list transaction counts for the LDR services on all Storage Nodes, or you can view the transactions for a specific Storage Node. Also, you can use audit messages to monitor the operations and transactions of the StorageGRID Webscale system.

Viewing transactions for S3 objects

You can view the number of successful and failed attempts by client applications to read, write, and modify S3 objects in the StorageGRID Webscale system. You can view a summary of all transactions for all LDR services, or you can view the transactions for a specific LDR service. You might want to do this to evaluate the health of the system.

Steps

1. Sign in to the Grid Management Interface using a supported browser.
2. Select **Grid**.
3. Select ***site*** > **Overview** > **Main**, and then view the **API Operations** area.

   The API Operations area displays a summary of information from all of the LDR services that support S3 client applications.
4. Select ***Storage Node*** > **LDR** > **S3** > **Overview** > **Main** to view information for individual LDR services.

Accessing and reviewing audit logs

The StorageGRID Webscale system securely and reliably transports audit messages from each service within the StorageGRID Webscale system to one or more audit repositories. API-specific audit messages provide critical security, operations, and performance monitoring data that can help you evaluate the health of your system.

About this task

The StorageGRID Webscale system compresses audit logs after one day and renames them using the format *YYYY-MM-DD.txt.gz* (where the original date is preserved).

Steps

1. Log in to the server using the user name and password as recorded in the `Passwords.txt` file.
2. Access the audit log directory through a command line of the server that hosts the AMS service.
3. Go to the `/var/local/audit/export/` directory.
4. View the `audit.log` file.

Related information

Understanding audit messages

Benefits of active, idle, and concurrent HTTP connections

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

Benefits of different types of HTTP connections

The type of HTTP connection duration can impact the performance of the StorageGRID Webscale system.

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 Webscale system determines it has to perform an HTTP transaction to the time that the StorageGRID Webscale 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 Webscale 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 Webscale system

To optimize load balancing across the StorageGRID Webscale 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 Webscale system balances its load when a client application establishes an HTTP connection. Over time, an HTTP connection that the StorageGRID Webscale 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 Webscale 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 Webscale 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 Webscale 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 Webscale system supports are also factors.

StorageGRID Webscale 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 Webscale 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 Webscale system's topology affects the maximum number of concurrent transactions and connections that the system can support.

- Client applications that interact with the StorageGRID Webscale 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 Webscale system, you might have to reduce the degree of concurrency to avoid exceeding the limits of the system.

Separation of HTTP connection pools for read and write operations

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

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

Copyright information

Trademark information

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., 495 East Java Drive, Sunnyvale, CA 94089 U.S.
- Telephone: +1 (408) 822-6000
- Fax: +1 (408) 822-4501
- Support telephone: +1 (888) 463-8277

Index

A

access control policies
  description of *50*
  overview of *50*
  supported *50*
algorithms
  encryption *64*, *65*
  hash *64*, *65*
  supported by TLS *64*, *65*
Amazon Web Services Command Line Interface
  testing the S3 REST API configuration *9*
API
  configuring security for *49*
API Gateway Nodes
  IP address of *9*
  IP addresses on CLB service *9*
  port number of *9*
applications, client
  connecting with an S3 tenant account *7*
  monitoring health with audit messages *67*
  viewing transactions for S3 objects *67*
applications, S3 REST API client
  introduction to connecting to the StorageGRID Webscale system *7*
audit logs
  bucket operations tracked *34*
  monitoring health of client applications *67*
  object operations tracked *34*
  reviewing *67*
  S3 REST API *34*
authenticating requests
  described *17*
authentication
  HTTP connections *49*
AWS CLI
  See Amazon Web Services Command Line Interface

B

best practices
  for active HTTP connections *68*
  for concurrent HTTP connections *69*
  for idle HTTP connections *68*
  using separate pools of HTTP connections *70*
bucket names
  restrictions on, AWS *18*
bucket operations
  custom operations *23*
  implementation of DELETE Bucket *18*
  implementation of DELETE Bucket metadata notification configuration *23*
  implementation of GET bucket *18*
  implementation of GET Bucket consistency request *23*
  implementation of GET Bucket last access time *23*
  implementation of GET Bucket metadata notification configuration *23*
  implementation of HEAD Bucket *18*
  implementation of PUT Bucket consistency request *23*
  implementation of PUT Bucket last access time *23*
  implementation of PUT Bucket metadata notification configuration *23*
  maximum number of buckets *18*
  PUT Bucket *18*
  supported operations *18*
  tracked in the audit logs *34*
buckets
  versioning *13*

C

certificate authority (CA) certificates
  how client applications use for security with REST APIs *64*
CLB service
  IP addresses *9*
client applications
  connecting with an S3 tenant account *7*
  how certificates are used for security with REST APIs *64*
  monitoring health with audit messages *67*
  using separate pools of HTTP connections for read and write operations *70*
  viewing transactions for S3 objects *67*
client applications, S3 REST API
  introduction to connecting to the StorageGRID Webscale system *7*
CloudMirror
  implemented using PUT bucket replication *18*
comments
  how to send feedback about documentation *73*
common request headers
  supported by StorageGRID Webscale *17*
common response headers
  supported by StorageGRID Webscale *17*
connecting
  security and TLS in API *49*
connections
  benefits for concurrent HTTP *69*
  benefits for idle HTTP *68*
  best practices for active HTTP *68*
connections, HTTP
  introduction to creating between S3 REST API client applications and the StorageGRID Webscale system *7*

D

dates
  supported formats for StorageGRID Webscale *16*
DELETE Bucket metadata notification configuration
  description of *42*
  request example *42*
  response example *42*
DELETE Object

versioning and 24
DNS resource records 7
documentation
how to receive automatic notification of changes to 73
how to send feedback about 73

E

Elasticsearch
API to configure integration for 45
Elasticsearch integration
implemented using custom bucket APIs 23
encryption algorithms
supported by TLS 64, 65
error responses, S3 REST API
list of 15

F

feedback
how to send comments about documentation 73

G

GET Bucket consistency
description of 36
GET Bucket last access time
description of 40
request example 40
response example 40
GET Bucket metadata notification configuration
description of 42
request example 42
response example 42
GET Service operation
implementation of 18
GET Storage Usage
description of 38
request example 38
response example 38
grid nodes
IP addresses for 9

H

hash algorithms
supported by TLS 64, 65
health of client applications
monitoring using audit messages 67
HTTP Authorization header
described 17
HTTP connections
benefits for concurrent 69
benefits for idle 68
benefits for idle, active, and concurrent 68
benefits of different types 68
best practices for active 68
introduction to creating between S3 REST API client applications and the StorageGRID Webscale system 7
supported versions of 5
using separate pools for read and write operations 70
HTTP date formats
supported, StorageGRID for Webscale 16
HTTP ports
for an API Gateway Node 9
HTTP ports |
for a Storage Node 9
HTTPS connections
how client applications use certificates for security with REST APIs 64
IP address for grid nodes 9

I

ILM
how rules manage objects 12
information
how to send feedback about improving documentation 73
information lifecycle management
See ILM
IP addresses
for API Gateway Nodes 9
for Storage Nodes 9

L

LDR service
IP addresses 9
monitoring transactions 67
lifecycle management, information
See ILM
logs
reviewing audit 67

M

messages, audit
monitoring health of client applications 67
metadata notification configuration
adding 45
deleting 42
format of JSON notifications 48
metadata included in notifications 48
multipart upload operations
Abort Multipart Upload 29
Complete Multipart Upload 29
description of 29
Initiate Multipart Upload 29
List Multipart Uploads 29
List Parts 29
Upload Part 29
Upload Part - Copy 29

O

object operations
implementation of DELETE Multiple Objects 24
implementation of DELETE Object 24
implementation of GET Object 24
implementation of GET Object ACL 24
implementation of PUT Object 24

implementation of PUT Object - Copy *24*
supported operations *18*
tracked in the audit logs *34*
objects
how ILM rules manage objects ingested through the S3 REST API *12*
versioning *13*
objects, S3
viewing transactions for *67*
operations
introduction to supported S3 *15*
operations, bucket
tracked in the audit logs *34*
operations, object
implementation of *24*
tracked in the audit logs *34*

P

platform services
added in v11.0 *5*
APIs *6*
introduction *6*
using Put Bucket metadata notification to configure search integration *45*
using PUT bucket notification to configure event notifications *18*
using PUT bucket replication to configure CloudMirror *18*
policy conditions
examples *60*
port numbers
for an API Gateway Node *9*
for Storage Nodes *9*
PUT Bucket consistency
description of *37*
PUT Bucket last access time
description of *41*
request example *41*
response example *41*
PUT Bucket metadata notification configuration
description of *45*
request example *45*
response example *45*

Q

query parameters
described *17*

R

request headers
supported by StorageGRID Webscale *17*
request headers, HTTP
list of *36*, *38*, *40*, *42*
response headers
supported by StorageGRID Webscale *17*
response headers, HTTP
list of *36*, *38*, *40*, *42*
REST API
configuring security *49*
revision history
changes to support for S3 REST API *5*
root domain names
specifying *7*
rules, ILM
how they manage objects *12*

S

S3 objects
viewing transactions for *67*
S3 operations
introduction to supported *15*
S3 REST API
changes to system support of *5*
common request headers *17*
common response headers *17*
error responses *15*
implementation by the StorageGRID Webscale system *11*
introduction to *5*
support for *5*
supported versions of *5*
testing the connection using the Amazon Web Services Command Line Interface *9*
S3 tenant accounts
creating in StorageGRID Webscale *7*
maximum number of buckets *18*
search integration service
configured using Put Bucket metadata notification *45*
disabling *42*
enabling *45*
format of JSON documents *48*
GET metadata notification configuration *42*
object metadata sent to index *48*
PUT and GET Bucket metadata notification configuration *23*
security
configuring for REST API *49*
how client applications use certificates for with REST APIs *64*
how client applications use certificates with S3 *64*
how client applications use certificates with Swift *64*
Transport Layer Security *49*
server authentication *49*
server certificates
how client applications use for security with REST APIs *64*
service operations
implementation of GET Service operation *18*
Simple Storage Service API Reference
See S3 REST API
Storage Nodes
IP address of *9*
IP addresses on LDR service *9*
monitoring transactions *67*
port number of *9*
storage use
requests for discovering *38*, *40*
StorageGRID Webscale
introduction to supported S3 operations *15*
S3 REST API implementation *11*

security for REST API *49*
StorageGRID Webscale systems
introduction to HTTP connections from S3 REST API client applications *7*
suggestions
how to send feedback about documentation *73*
supported versions
changes to support for S3 REST API *5*
of HTTP *5*
of S3 REST API *5*

T

tenant accounts, S3
creating in StorageGRID Webscale *7*
TLS
how client applications use certificates for security with REST APIs *64*
security in API *49*
supported hashing algorithms *64*, *65*
transactions
viewing for client applications *67*
transactions, S3 objects
viewing for client applications *67*
Transport Layer Security
See TLS
troubleshooting
using audit logs *67*
Twitter
how to receive automatic notification of documentation changes *73*

V

versioning
enabling for buckets *13*
objects *13*
versions supported
of HTTP *5*
of S3 REST API *5*
versions, supported
changes to support for S3 REST API *5*
viewing list of
downloading list of *7*

X

x-amz-date
option in date request headers *16*