



StorageGRID® 11.3

# S3 (Simple Storage Service) Implementation Guide

November 2019 | 215-14206\_2019-11\_en-us  
[doccomments@netapp.com](mailto:doccomments@netapp.com)



# Contents

<b>Support for the S3 REST API .....</b>	<b>5</b>
Changes to S3 REST API support .....	5
Supported versions .....	6
Support for StorageGRID platform services .....	6
<b>Configuring tenant accounts and connections .....</b>	<b>8</b>
Creating and configuring S3 tenant accounts .....	8
Configuring client connections .....	9
Identifying IP addresses for client connections .....	10
Deciding to use HTTPS or HTTP connections .....	11
Endpoint domain names for S3 requests .....	11
Testing your S3 REST API configuration .....	12
<b>How StorageGRID implements the S3 REST API .....</b>	<b>14</b>
Conflicting client requests .....	14
Consistency controls .....	14
How StorageGRID ILM rules manage objects .....	16
Object versioning .....	17
Recommendations for implementing the S3 REST API .....	18
<b>S3 REST API supported operations and limitations .....</b>	<b>19</b>
Date handling .....	19
Common request headers .....	19
Common response headers .....	19
Authenticating requests .....	20
Operations on the service .....	20
Operations on buckets .....	21
Custom operations on buckets .....	26
Operations on objects .....	27
Using server-side encryption .....	30
GET Object .....	31
HEAD Object .....	33
POST Object restore .....	35
PUT Object .....	36
PUT Object - Copy .....	40
Operations for multipart uploads .....	43
List Multipart Uploads .....	44
Initiate Multipart Upload .....	44
Upload Part .....	46
Upload Part - Copy .....	47
Complete Multipart Upload .....	48
Error responses .....	49
<b>StorageGRID S3 REST API operations .....</b>	<b>52</b>
PUT Bucket request modifications for compliance .....	52

GET Bucket compliance request .....	53
PUT Bucket compliance request .....	55
GET Bucket consistency request .....	57
PUT Bucket consistency request .....	58
GET Bucket last access time request .....	59
PUT Bucket last access time request .....	59
DELETE Bucket metadata notification configuration request .....	60
GET Bucket metadata notification configuration request .....	61
PUT Bucket metadata notification configuration request .....	63
JSON generated by the search integration service .....	66
Object metadata included in metadata notifications .....	66
GET Storage Usage request .....	67
<b>Bucket and group access policies .....</b>	<b>68</b>
Access policy overview .....	68
Consistency control settings for policies .....	70
Using the ARN in policy statements .....	70
Specifying resources in a policy .....	71
Specifying principals in a policy .....	71
Specifying permissions in a policy .....	72
Using the PutOverwriteObject permission .....	74
Specifying conditions in a policy .....	75
Specifying variables in a policy .....	77
Creating policies requiring special handling .....	78
Write-once-read-many (WORM) protection .....	79
S3 policy examples .....	80
S3 bucket policy examples .....	80
S3 group policy examples .....	83
<b>Configuring security for the REST API .....</b>	<b>86</b>
How StorageGRID provides security for the REST API .....	86
Supported hashing and encryption algorithms for TLS libraries .....	87
<b>Monitoring and auditing operations .....</b>	<b>89</b>
Monitoring object ingest and retrieval rates .....	89
Accessing and reviewing audit logs .....	91
S3 operations tracked in the audit logs .....	91
<b>Benefits of active, idle, and concurrent HTTP connections .....</b>	<b>93</b>
Benefits of keeping idle HTTP connections open .....	93
Benefits of active HTTP connections .....	93
Benefits of concurrent HTTP connections .....	94
Separation of HTTP connection pools for read and write operations .....	95
<b>Copyright .....</b>	<b>96</b>
<b>Trademark information .....</b>	<b>97</b>
<b>How to send comments about documentation and receive update notifications .....</b>	<b>98</b>

## Support for the S3 REST API

---

StorageGRID supports the Simple Storage Service (S3) API, which is implemented as a set of Representational State Transfer (REST) web services. Support for the S3 REST API enables you to connect service-oriented applications developed for S3 web services with on-premises object storage that uses the StorageGRID system. This requires minimal changes to a client application's current use of S3 REST API calls.

### Changes to S3 REST API support

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

Release	Comments
11.3	<ul style="list-style-type: none"> <li>Added support for server-side encryption of object data with customer-provided keys (SSE-C).</li> <li>Added support for DELETE, GET, and PUT Bucket lifecycle operations (Expiration action only) and for the <code>x-amz-expiration</code> response header.</li> <li>Updated PUT Object, PUT Object - Copy, and Multipart Upload to describe the impact of ILM rules that use synchronous placement at ingest.</li> <li>Updated list of supported TLS cipher suites.</li> </ul>
11.2	<p>Added support for POST Object restore for use with Cloud Storage Pools. Added support for using the AWS syntax for ARN, policy condition keys, and policy variables in group and bucket policies. Existing group and bucket policies that use the StorageGRID syntax will continue to be supported.</p> <p><b>Note:</b> Uses of ARN/URN in other configuration JSON/XML, including those used in custom StorageGRID features, have not changed.</p>
11.1	Added support for Cross-Origin Resource Sharing (CORS), HTTP for S3 client connections to grid nodes, and compliance settings on buckets.
11.0	Added support for configuring platform services (CloudMirror replication, notifications, and Elasticsearch search integration) for buckets. Also added support for object tagging location constraints for buckets, and the Available consistency control setting.
10.4	Added support for ILM scanning changes to versioning, Endpoint Domain Names page updates, conditions and variables in policies, policy examples, and the PutOverwriteObject permission.
10.3	Added support for versioning.
10.2	Added support for group and bucket access policies, and for multipart copy (Upload Part - Copy).

Release	Comments
10.1	Added support for multipart upload, virtual hosted-style requests, and v4 authentication.
10.0	Initial support of the S3 REST API by the StorageGRID system. The currently supported version of the <i>Simple Storage Service API Reference</i> is 2006-03-01.

## Supported versions

StorageGRID supports the following specific versions of S3 and HTTP.

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

### Related information

[IETF RFC 2616: Hypertext Transfer Protocol \(HTTP/1.1\)](#)

[Amazon Web Services \(AWS\) Documentation: Amazon Simple Storage Service API Reference](#)

## Support for StorageGRID platform services

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

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

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

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

## Recommendations for using platform services

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

- NetApp recommends that you allow no more than 100 active tenants with S3 requests requiring CloudMirror replication, notifications, and search integration. Having more than 100 active tenants can result in slower S3 client performance.
- If an S3 bucket in the StorageGRID system has both versioning and CloudMirror replication enabled, NetApp recommends that the destination endpoint also have S3 bucket versioning enabled. This allows CloudMirror replication to generate similar object versions on the endpoint.
- If the destination S3 bucket for CloudMirror replication has compliance enabled, the replication operation will fail with an AccessDenied error.

### Related concepts

[Operations on buckets](#) on page 21

### Related references

[PUT Bucket metadata notification configuration request](#) on page 63

### Related information

[Using tenant accounts](#)

[Administering StorageGRID](#)

## Configuring tenant accounts and connections

---

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

### Steps

1. [Creating and configuring S3 tenant accounts](#) on page 8
2. [Configuring client connections](#) on page 9
3. [Endpoint domain names for S3 requests](#) on page 11
4. [Testing your S3 REST API configuration](#) on page 12

## Creating and configuring S3 tenant accounts

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

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

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

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

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

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



**Related information**[Administering StorageGRID](#)[Using tenant accounts](#)

## Configuring client connections

A grid administrator makes configuration choices that affect how S3 clients connect to StorageGRID to store and retrieve data. The specific information you need to make a connection depends upon the configuration that was chosen.

Client applications can store or retrieve objects by connecting to any of the following:

- The Load Balancer service on Admin Nodes or Gateway Nodes, or optionally, the virtual IP address of a high availability (HA) group of Admin Nodes or Gateway Nodes
- The legacy CLB service on Gateway Nodes, or optionally, the virtual IP address of a high availability group of Gateway Nodes
- Storage Nodes, with or without an external load balancer

The Load Balancer service is new in StorageGRID 11.3. Existing clients can continue to use the legacy CLB service on Gateway Nodes to connect to StorageGRID. New client applications that depend on StorageGRID to provide load balancing should connect using the Load Balancer service.

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

1. **Configure endpoints for the Load Balancer service.**  
You must configure endpoints to use the Load Balancer service. The Load Balancer service on Admin Nodes or Gateway Nodes distributes incoming network connections from client applications to Storage Nodes. When creating a load balancer endpoint, the grid administrator specifies a port number, whether the endpoint accepts HTTP or HTTPS connections, the type of client (S3 or Swift) that will use the endpoint, and the certificate to be used for HTTPS connections (if applicable).
2. **Configure Untrusted Client Networks.**  
If a grid administrator configures a node's Client Network to be untrusted, the node only accepts inbound connections on the Client Network on ports that are explicitly configured as load balancer endpoints.
3. **Configure high availability groups.**  
If an administrator creates an HA group, the network interfaces of multiple Admin Nodes or Gateway Nodes are placed into an active-backup configuration. Client connections are made using the virtual IP address of the HA group.

For more information about each option, see the instructions for administering StorageGRID.

**Information required to make client connections**

The table summarizes the information that you require to make a client connection for each configuration.

Where connection is made	Service that client connects to	IP address	Port
HA group	Load Balancer	HA group virtual IP address	<ul style="list-style-type: none"> <li>• Load balancer endpoint port</li> </ul>

Where connection is made	Service that client connects to	IP address	Port
Admin Node	Load Balancer	Admin Node IP address	<ul style="list-style-type: none"> <li>Load balancer endpoint port</li> </ul>
Gateway Node	Load Balancer	Gateway Node IP address	<ul style="list-style-type: none"> <li>Load balancer endpoint port</li> </ul>
Gateway Node	Legacy CLB	Gateway Node IP address	<ul style="list-style-type: none"> <li>HTTPS: 8082</li> <li>HTTP: 8084</li> </ul>
Storage Node	LDR	Storage Node IP address	<ul style="list-style-type: none"> <li>HTTPS: 18082</li> <li>HTTP: 18084</li> </ul>

#### Related information

[Administering StorageGRID](#)

## Identifying IP addresses for client connections

Client applications can use the IP address of a Gateway Node, Admin Node, or Storage Node to connect to StorageGRID. If high availability groups are configured, client applications can connect using the virtual IP address of the HA group.

#### About this task

You can use the Grid Manager to look up the IP address or virtual IP address that you require to make a client connection to StorageGRID.

#### Steps

1. Sign in to the Grid Manager using a supported browser.
2. To find the IP address of a grid node:
  - a. Select **Nodes**.
  - b. Select the Admin Node, Gateway Node, or Storage Node to which you want to connect.
  - c. Select the **Overview** tab.
  - d. In the Node Information section, note the IP addresses for the node.
  - e. Click **Show more** to view IPv6 addresses and interface mappings.

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

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

**Note:** If you are viewing an Admin Node or a Gateway Node and it is the active node in a high availability group, the virtual IP address of the HA group is shown on eth2.

3. To find the virtual IP address of a high availability group:
  - a. Select **Configuration > High Availability Groups**.
  - b. In the table, note the virtual IP address of the HA group.

#### Related information

[Administering StorageGRID](#)

## Deciding to use HTTPS or HTTP connections

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

By default, when connecting to Storage Nodes or the legacy CLB service on Gateway Nodes, client applications must use encrypted HTTPS for all connections. Optionally, you can enable less-secure HTTP connections by enabling the **HTTP** option from the Grid Manager. For example, a client application might use HTTP when testing the connection to a Storage Node in a non-production environment.

**Attention:** Be careful when enabling HTTP for a production grid since requests will be sent unencrypted.

To learn how to enable this option, see information about administering StorageGRID.

If the **HTTP** option has been enabled for the grid, for connections to Storage Nodes or the legacy CLB service on Gateway Nodes you use different ports for HTTP than you use for HTTPS.

#### Related concepts

[Benefits of active, idle, and concurrent HTTP connections](#) on page 93

#### Related information

[Administering StorageGRID](#)

## Endpoint domain names for S3 requests

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

#### About this task

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

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

For example, if the endpoint is `s3.company.com`, the grid administrator must ensure that the certificate used for HTTPS connections includes the `s3.company.com` endpoint and the endpoint's wildcard Subject Alternative Name (SAN): `*.s3.company.com`.

- Configure the DNS server used by the client to include DNS records that match the endpoint domain names, including any required wildcard records.

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

**Note:** Each load balancer endpoint has its own certificate, and each endpoint can be configured to recognize different endpoint domain names.

If the client connects Storage Nodes or to the legacy CLB service on Gateway Nodes, the certificate that the grid administrator configures is the single custom server certificate used for the grid.

See the instructions for administering StorageGRID for more information.

After these steps have been completed, you can use virtual hosted-style requests (for example, `bucket.s3.company.com`).

#### Related concepts

[Configuring security for the REST API](#) on page 86

#### Related information

[Administering StorageGRID](#)

## Testing your S3 REST API configuration

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

#### Before you begin

- You must have downloaded and installed the AWS CLI from [aws.amazon.com/cli](https://aws.amazon.com/cli).
- You must have created an S3 tenant account in the StorageGRID system.

#### Steps

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

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

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

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

**3.** Upload an object.

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

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

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

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

**5.** Delete the object.

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

**6.** Delete the bucket.

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

## How StorageGRID implements the S3 REST API

A client application can use S3 REST API calls to connect to StorageGRID to create, delete, and modify buckets, as well as storing and retrieving objects.

### Conflicting client requests

Conflicting client requests, such as two clients writing to the same key, are resolved on a “latest-wins” basis.

The timing for the “latest-wins” evaluation is based on when the StorageGRID system completes a given request, and not on when S3 clients begin an operation.

### Consistency controls

Consistency controls let you make a trade-off between the availability of the objects and the consistency of those objects across different Storage Nodes and sites, as required by your application.

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

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

#### Consistency controls

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

Consistency control	Description
all	All nodes receive the data immediately, or the request will fail.
strong-global	Guarantees read-after-write consistency for all client requests across all sites.
strong-site	Guarantees read-after-write consistency for all client requests within a site.
read-after-new-write	(Default) Provides read-after-write consistency for new objects and eventual consistency for object updates. Offers high availability and data protection guarantees. Matches AWS S3 consistency guarantees.  <b>Note:</b> If your application uses HEAD requests on objects that do not exist, you might receive a high number of 500 Internal Server errors if one or more Storage Nodes are unavailable. To prevent these errors, set the consistency control to “available” unless you require consistency guarantees similar to AWS S3.
available (eventual consistency for HEAD operations)	Behaves the same as the “read-after-new-write” consistency level, but only provides eventual consistency for HEAD operations. Offers higher availability for HEAD operations than “read-after-new-write” if Storage Nodes are unavailable. Differs from AWS S3 consistency guarantees for HEAD operations only.

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

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

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

- It first looks up the object using a low consistency.
- If that lookup fails, it repeats the lookup at the next consistency level until it reaches the highest consistency level, “all,” which requires all copies of the object metadata to be available.

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

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

### Specifying the consistency control for an API operation

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

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

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

### Specifying the consistency control for a bucket

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

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

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

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

#### Related references

[GET Bucket consistency request](#) on page 57

[PUT Bucket consistency request](#) on page 58

## How StorageGRID ILM rules manage objects

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

ILM settings determine the following aspects of an object:

#### Geography

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

#### Storage grade

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

#### Loss protection

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

#### Retention

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

#### Protection during ingest

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

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

- Tenant Account
- Bucket Name
- Ingest Time
- Key
- Last Access Time

**Note:** By default, updates to last access time are disabled for all S3 buckets. If your StorageGRID system includes an ILM rule that uses the Last Access Time option, you must enable updates to last access time for the S3 buckets specified in that rule. You can enable last access time updates using the PUT Bucket last access time request, the **S3 > Buckets > Configure Last Access Time** check box in the Tenant Manager, or using the Tenant



Management API. When enabling last access time updates, be aware that StorageGRID performance might be reduced, especially in systems with small objects.

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

For more information about ILM, see “Managing objects through information lifecycle management” in the instructions for administering StorageGRID.

#### Related references

[PUT Bucket last access time request](#) on page 59

#### Related information

[Using tenant accounts](#)

[Administering StorageGRID](#)

## Object versioning

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

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

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

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

**Note:** Versioning can be enabled only on buckets created with StorageGRID version 10.3 or later.

#### ILM and versioning

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

For S3 objects in versioning-enabled buckets, versioning support allows you to create ILM rules that use Noncurrent Time as the Reference Time. When an object is updated, its previous versions become noncurrent. Using a noncurrent time filter allows you to create policies that reduce the storage impact of previous versions of objects.

See the instructions for administering StorageGRID for an example ILM policy for managing S3 versioned objects.

#### Related information

[Administering StorageGRID](#)

## Recommendations for implementing the S3 REST API

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

### Recommendations for HEADs to non-existent objects

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

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

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

### Recommendations for object keys

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

If you do follow the AWS recommendation to use random and unique characters in key prefixes, you should prefix the object keys with a directory name. That is, use this format:

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

Instead of this format:

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

### Recommendations for “range reads”

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

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

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

### Related concepts

[Consistency controls](#) on page 14

### Related references

[PUT Bucket consistency request](#) on page 58

### Related information

[Administering StorageGRID](#)

## S3 REST API supported operations and limitations

---

The StorageGRID system implements the *Simple Storage Service API Reference* (API Version 2006-03-01) with support for most operations, and with some limitations. You need to understand the implementation details when you are integrating S3 REST API client applications.

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

### Related information

[Amazon Web Services \(AWS\) Documentation: Amazon Simple Storage Service API Reference](#)

## Date handling

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

The StorageGRID system only supports valid HTTP date formats for any headers that accept date values. The time portion of the date can be specified in Greenwich Mean Time (GMT) format, or in Universal Coordinated Time (UTC) format with no time zone offset (+0000 must be specified). If you include the `x-amz-date` header in your request, it overrides any value specified in the Date request header. When using AWS Signature Version 4, the `x-amz-date` header must be present in the signed request because the date header is not supported.

## Common request headers

The StorageGRID system supports common request headers defined by the *Simple Storage Service API Reference*, with one exception.

Request header	Implementation
Authorization	Full support for AWS Signature Version 2 Support for AWS Signature Version 4, with the following exceptions: <ul style="list-style-type: none"> <li>The SHA256 value is not calculated for the body of the request. The user submitted value is accepted without validation.</li> </ul>
x-amz-security-token	Not implemented. Returns XNotImplemented.

### Related information

[Amazon Web Services \(AWS\) Documentation: Amazon Simple Storage Service API Reference](#)

## Common response headers

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

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

**Related information**

[Amazon Web Services \(AWS\) Documentation: Amazon Simple Storage Service API Reference](#)

**Authenticating requests**

The StorageGRID system supports both authenticated and anonymous access to objects using the S3 API.

The S3 API supports Signature version 2 and Signature version 4 for authenticating S3 API requests.

Authenticated requests must be signed using your access key ID and secret access key.

The StorageGRID system supports two authentication methods: the HTTP `Authorization` header and using query parameters.

**Using the HTTP Authorization header**

The HTTP `Authorization` header is used by all S3 API operations except Anonymous requests where permitted by the bucket policy. The `Authorization` header contains all of the required signing information to authenticate a request.

**Using query parameters**

You can use query parameters to add authentication information to a URL. This is known as presigning the URL, which can be used to grant temporary access to specific resources. Users with the presigned URL do not need to know the secret access key in order to access the resource, which enables you to provide third-party restricted access to a resource.

**Operations on the service**

The StorageGRID system supports the following operations on the service.

Operation	Implementation
GET Service	Implemented with all Amazon S3 REST API behavior.
GET Storage Usage	The GET Storage Usage request tells you the total amount of storage in use by an account, and for each bucket associated with the account. This is an operation on the service with a path of / and a custom query parameter ( <code>?x-ntap-sg-usage</code> ) added.
OPTIONS /	Client applications can issue <code>OPTIONS /</code> requests to the S3 port on a Storage Node, without providing S3 authentication credentials, to determine whether the Storage Node is available. You can use this request for monitoring, or to allow external load balancers to identify when a Storage Node is down.

**Related references**

[GET Storage Usage request](#) on page 67

## Operations on buckets

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

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

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

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

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

Operation	Implementation
DELETE Bucket	Implemented with all Amazon S3 REST API behavior.
DELETE Bucket cors	This operation deletes the CORS configuration for the bucket.
DELETE Bucket lifecycle	This operation deletes the lifecycle configuration from the bucket.
DELETE Bucket policy	This operation deletes the policy attached to the bucket.
DELETE Bucket replication	This operation deletes the replication configuration attached to the bucket.
GET Bucket (List Objects), version 1 and version 2	<p>This operation returns some or all (up to 1,000) of the objects in a bucket.</p> <p>The Storage Class for objects can have either of two values, even if the object was ingested with the REDUCED_REDUNDANCY storage class option:</p> <ul style="list-style-type: none"> <li>STANDARD, which indicates the object is stored in a storage pool consisting of Storage Nodes.</li> <li>GLACIER, which indicates that the object has been moved to the external bucket specified by the Cloud Storage Pool.</li> </ul> <p>If the bucket contains large numbers of deleted keys that have the same prefix, the response might include some CommonPrefixes that do not contain keys.</p>
GET Bucket acl	This operation returns a positive response and the ID, DisplayName, and Permission of the bucket owner, indicating that the owner has full access to the bucket.
GET Bucket cors	This operation returns the cors configuration for the bucket.
GET Bucket lifecycle	This operation returns the lifecycle configuration for the bucket.
GET Bucket location	This operation returns the region that was set using the LocationConstraint element in the PUT Bucket request. If the bucket's region is us-east-1, an empty string is returned for the region.
GET Bucket notification	This operation returns the notification configuration attached to the bucket.

Operation	Implementation
GET Bucket Object versions	With READ access on a bucket, this operation with the <code>versions</code> subresource lists metadata of all of the versions of objects in the bucket.
GET Bucket policy	This operation returns the policy attached to the bucket.
GET Bucket replication	This operation returns the replication configuration attached to the bucket.
GET Bucket versioning	This implementation uses the <code>versioning</code> subresource to return the versioning state of a bucket. The versioning state returned indicates if the bucket is “Unversioned” or if the bucket is version “Enabled” or “Suspended.”
HEAD Bucket	This operation determines if a bucket exists and you have permission to access it.
PUT Bucket	<p>This operation creates a new bucket. By creating the bucket, you become the bucket owner.</p> <ul style="list-style-type: none"> <li>• Bucket names must comply with the following rules: <ul style="list-style-type: none"> <li>◦ Must be unique across each StorageGRID system (not just unique within the tenant account).</li> <li>◦ Must be DNS compliant.</li> <li>◦ Must contain between 3 and 63 characters.</li> <li>◦ Can be a series of one or more labels, with adjacent labels separated by a period. Each label must start and end with a lowercase letter or a number and can only use lowercase letters, numbers, and hyphens.</li> <li>◦ Must not look like a text-formatted IP address.</li> <li>◦ Should not use periods in virtual hosted-style requests because periods will cause problems with server wildcard certificate verification.</li> </ul> </li> <li>• By default, buckets are created in the <code>us-east-1</code> region; however, you can use the <code>LocationConstraint</code> request element in the request body to specify a different region. When using the <code>LocationConstraint</code> element, you must specify the exact name of a region that has been defined using the Grid Manager or the Grid Management API. Contact your system administrator if you do not know the region name you should use.</li> </ul> <p><b>Note:</b> An error will occur if your PUT Bucket request uses a region that has not been defined in StorageGRID.</p> <ul style="list-style-type: none"> <li>• When using a PUT Bucket request, you can include the <code>SGCompliance</code> XML element in the XML request body. <code>SGCompliance</code> is a custom StorageGRID element that allows you to create buckets that are compliant.</li> </ul>

Operation	Implementation
PUT Bucket cors	<p>This operation sets the CORS configuration for a bucket so that the bucket can service cross-origin requests.</p> <p>Cross-origin resource sharing (CORS) is a security mechanism that allows client web applications in one domain to access resources in a different domain. For example, suppose you use an S3 bucket named <code>images</code> to store graphics. By setting the CORS configuration for the <code>images</code> bucket, you can allow the images in that bucket to be displayed on the website <code>http://www.example.com</code>.</p>
PUT Bucket lifecycle	<p>This operation creates a new lifecycle configuration for the bucket or replaces an existing lifecycle configuration.</p> <p>StorageGRID supports up to 1,000 lifecycle rules in a lifecycle configuration. Each rule can include the following XML elements:</p> <ul style="list-style-type: none"> <li>• Expiration (Days, Date)</li> <li>• NoncurrentVersionExpiration (NoncurrentDays)</li> <li>• Filter (Prefix, Tag)</li> <li>• Status</li> <li>• ID</li> </ul> <p>StorageGRID does not support these actions:</p> <ul style="list-style-type: none"> <li>• AbortIncompleteMultipartUpload</li> <li>• ExpiredObjectDeleteMarker</li> <li>• Transition</li> </ul> <p>To understand how the Expiration action in a bucket lifecycle interacts with ILM placement instructions, see “How ILM operates throughout an object’s life” in the instructions for administering StorageGRID.</p> <p><b>Note:</b> Lifecycle configuration is not supported for compliant buckets.</p>

Operation	Implementation										
PUT Bucket notification	<p>This operation configures notifications for the bucket using the notification configuration XML included in the request body. You should be aware of the following implementation details:</p> <ul style="list-style-type: none"> <li>• StorageGRID supports Simple Notification Service (SNS) topics as destinations. Simple Queue Service (SQS) or Amazon Lambda endpoints are not supported.</li> <li>• The destination for notifications must be specified as the URN of an StorageGRID endpoint. Endpoints can be created using the Tenant Manager or the Tenant Management API. The endpoint must exist for notification configuration to succeed. If the endpoint does not exist, a 400 <code>Bad Request</code> error is returned with the code <code>InvalidArgument</code>.</li> <li>• You cannot configure a notification for the following event types. These event types are <b>not</b> supported. <ul style="list-style-type: none"> <li>◦ <code>s3:ReducedRedundancyLostObject</code></li> <li>◦ <code>s3:ObjectRestore:Completed</code></li> <li>◦ <code>s3:ObjectRestore:Post</code></li> </ul> </li> <li>• Event notifications sent from StorageGRID use the standard JSON format except that they do not include some keys and use specific values for others, as shown in the table:</li> </ul> <table border="1" data-bbox="737 1129 1377 1381"> <thead> <tr> <th>Key name</th> <th>StorageGRID value</th> </tr> </thead> <tbody> <tr> <td><code>eventSource</code></td> <td><code>sgws:s3</code></td> </tr> <tr> <td><code>awsRegion</code></td> <td>not included</td> </tr> <tr> <td><code>x-amz-id-2</code></td> <td>not included</td> </tr> <tr> <td><code>arn</code></td> <td><code>urn:sgws:s3:::bucket_name</code></td> </tr> </tbody> </table>	Key name	StorageGRID value	<code>eventSource</code>	<code>sgws:s3</code>	<code>awsRegion</code>	not included	<code>x-amz-id-2</code>	not included	<code>arn</code>	<code>urn:sgws:s3:::bucket_name</code>
Key name	StorageGRID value										
<code>eventSource</code>	<code>sgws:s3</code>										
<code>awsRegion</code>	not included										
<code>x-amz-id-2</code>	not included										
<code>arn</code>	<code>urn:sgws:s3:::bucket_name</code>										
PUT Bucket policy	This operation sets the policy attached to the bucket.										



Operation	Implementation
PUT Bucket replication	<p>This operation configures StorageGRID CloudMirror replication for the bucket using the replication configuration XML provided in the request body.</p> <p>For CloudMirror replication, you should be aware of the following implementation details:</p> <ul style="list-style-type: none"> <li>• StorageGRID only supports V1 of the replication configuration. This means that StorageGRID does not support the use of the <code>Filter</code> element for rules, and follows V1 conventions for deletion of object versions. See the Amazon documentation on replication configuration for details.</li> <li>• Bucket replication can be configured on versioned or unversioned buckets.</li> <li>• 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.</li> <li>• Destination buckets must be specified as the URN of StorageGRID endpoints as specified in the Tenant Manager or the Tenant Management API. The endpoint must exist for replication configuration to succeed. If the endpoint does not exist, the request fails as a 400 Bad Request. The error message states: <code>Unable to save the replication policy. The specified endpoint URN does not exist: URN.</code></li> <li>• You do not need to specify a <code>Role</code> in the configuration XML. This value is not used by StorageGRID and will be ignored if submitted.</li> <li>• If you omit the storage class from the configuration XML, StorageGRID uses the <code>STANDARD</code> storage class by default.</li> <li>• If you delete an object from the source bucket or you delete the source bucket itself, the cross-region replication behavior is as follows: <ul style="list-style-type: none"> <li>◦ If you delete the object or bucket before it has been replicated, the object/bucket is not replicated and you are not notified.</li> <li>◦ If you delete the object or bucket after it has been replicated, StorageGRID follows standard Amazon S3 delete behavior for V1 of cross-region replication.</li> </ul> </li> </ul>

Operation	Implementation
PUT Bucket versioning	<p>This implementation uses the <code>versioning</code> subresource to set the versioning state of an existing bucket. You can set the versioning state with one of the following values:</p> <ul style="list-style-type: none"> <li>• <b>Enabled:</b> Enables versioning for the objects in the bucket. All objects added to the bucket receive a unique version ID.</li> <li>• <b>Suspended:</b> Disables versioning for the objects in the bucket. All objects added to the bucket receive the version ID <code>null</code>.</li> </ul>

**Related concepts**

[Consistency controls](#) on page 14

[Bucket and group access policies](#) on page 68

[S3 operations tracked in the audit logs](#) on page 91

**Related references**

[GET Bucket last access time request](#) on page 59

[PUT Bucket request modifications for compliance](#) on page 52

**Related information**

[Amazon Web Services \(AWS\) Documentation: Cross-Region Replication](#)

[Administering StorageGRID](#)

[Using tenant accounts](#)

## Custom operations on buckets

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

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

Operation	Description	For more information
PUT Bucket modifications for compliance	Creates a compliant bucket and sets the compliance settings for that bucket.	<a href="#">PUT Bucket request modifications for compliance</a> on page 52
GET Bucket compliance	Returns the compliance settings currently in effect for a bucket that was created to be compliant.	<a href="#">GET Bucket compliance request</a> on page 53
PUT Bucket compliance	Allows you to modify the compliance settings for an existing bucket that was created to be compliant.	<a href="#">PUT Bucket compliance request</a> on page 55
GET Bucket consistency	Returns the consistency level being applied to a particular bucket.	<a href="#">GET Bucket consistency request</a> on page 57

Operation	Description	For more information
PUT Bucket consistency	Sets the consistency level applied to a particular bucket.	<a href="#">PUT Bucket consistency request</a> on page 58
GET Bucket last access time	Returns whether last access time updates are enabled or disabled for a particular bucket.	<a href="#">GET Bucket last access time request</a> on page 59
PUT Bucket last access time	Allows you to enable or disable last access time updates for a particular bucket.	<a href="#">PUT Bucket last access time request</a> on page 59
DELETE Bucket metadata notification configuration	Deletes the metadata notification configuration XML associated with a particular bucket.	<a href="#">DELETE Bucket metadata notification configuration request</a> on page 60
GET Bucket metadata notification configuration	Returns the metadata notification configuration XML associated with a particular bucket.	<a href="#">GET Bucket metadata notification configuration request</a> on page 61
PUT Bucket metadata notification configuration	Configures the metadata notification service for a bucket.	<a href="#">PUT Bucket metadata notification configuration request</a> on page 63

### Related concepts

[S3 operations tracked in the audit logs](#) on page 91

## Operations on objects

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

The following conditions apply to all object operations:

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

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

Operation	Implementation
DELETE Object	<p>Multi-Factor Authentication (MFA) and the response header <code>x-amz-mfa</code> are not supported.</p> <p>When processing a DELETE Object request, StorageGRID attempts to immediately remove all copies of the object from all stored locations. If successful, StorageGRID returns a response to the client immediately. If all copies cannot be removed within 30 seconds (for example, because a location is temporarily unavailable), StorageGRID queues the copies for removal and then indicates success to the client.</p> <p>In previous releases, StorageGRID always queued objects for removal. With the introduction of immediate deletion, clients might sometimes receive a slower response, even though object copies are generally being removed more quickly.</p> <p><b>Versioning</b></p> <p>To remove a specific version, the requestor must be the bucket owner and use the <code>versionId</code> subresource. Using this subresource permanently deletes the version. If the <code>versionId</code> corresponds to a delete marker, the response header <code>x-amz-delete-marker</code> is returned set to <code>true</code>.</p> <ul style="list-style-type: none"> <li>• If an object is deleted without the <code>versionId</code> subresource on a version enabled bucket, it results in the generation of a delete marker. The <code>versionId</code> for the delete marker is returned using the <code>x-amz-version-id</code> response header, and the <code>x-amz-delete-marker</code> response header is returned set to <code>true</code>.</li> <li>• If an object is deleted without the <code>versionId</code> 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 <code>x-amz-delete-marker</code> response header is returned set to <code>true</code>.</li> </ul> <p><b>Note:</b> In certain cases, multiple delete markers might exist for an object.</p>
DELETE Multiple Objects	<p>Multi-Factor Authentication (MFA) and the response header <code>x-amz-mfa</code> are not supported.</p> <p>Multiple objects can be deleted in the same request message.</p> <p><b>Note:</b> The DELETE Multiple Objects request is not supported on versioned buckets.</p>
DELETE Object tagging	<p>Uses the <code>tagging</code> subresource to remove all tags from an object. Implemented with all Amazon S3 REST API behavior.</p> <p><b>Versioning</b></p> <p>If the <code>versionId</code> 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 <code>x-amz-delete-marker</code> response header set to <code>true</code>.</p>
GET Object	<a href="#">GET Object</a> on page 31

Operation	Implementation
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	<p>Uses the <code>tagging</code> subresource to return all tags for an object. Implemented with all Amazon S3 REST API behavior</p> <p><b>Versioning</b></p> <p>If the <code>versionId</code> 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 <code>x-amz-delete-marker</code> response header set to <code>true</code>.</p>
HEAD Object	<a href="#">HEAD Object</a> on page 33
POST Object restore	<a href="#">POST Object restore</a> on page 35
PUT Object	<a href="#">PUT Object</a> on page 36
PUT Object - Copy	<a href="#">PUT Object - Copy</a> on page 40
PUT Object tagging	<p>Uses the <code>tagging</code> subresource to add a set of tags to an existing object. Implemented with all Amazon S3 REST API behavior</p> <p><b>Tag updates and ingest behavior</b></p> <p>When you use PUT Object tagging to update an object's tags, StorageGRID does not re-ingest the object. This means that the option for Ingest Behavior specified in the matching ILM rule is not used. Any changes to object placement that are triggered by the update are made when ILM is re-evaluated by normal background ILM processes. This means that if the ILM rule uses the Strict option for ingest behavior, no action is taken if the required object placements cannot be made (for example, because a newly required location is unavailable). The updated object retains its current placement until the required placement is possible.</p> <p><b>Resolving conflicts</b></p> <p>Conflicting client requests, such as two clients writing to the same key, are resolved on a “latest-wins” basis. The timing for the “latest-wins” evaluation is based on when the StorageGRID system completes a given request, and not on when S3 clients begin an operation.</p> <p><b>Versioning</b></p> <p>If the <code>versionId</code> 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 <code>x-amz-delete-marker</code> response header set to <code>true</code>.</p>

**Related concepts**

[Consistency controls](#) on page 14

[S3 operations tracked in the audit logs](#) on page 91

## Using server-side encryption

Server-side encryption allows you to protect your object data at rest. StorageGRID encrypts the data as it writes the object and decrypts the data when you access the object.

If you want to use server-side encryption, you can choose either of two mutually exclusive options, based on how the encryption keys are managed:

- **SSE (server-side encryption with StorageGRID-managed keys):** When you issue an S3 request to store an object, StorageGRID encrypts the object with a unique key. When you issue an S3 request to retrieve the object, StorageGRID uses the stored key to decrypt the object.
- **SSE-C (server-side encryption with customer-provided keys):** When you issue an S3 request to store an object, you provide your own encryption key. When you retrieve an object, you provide the same encryption key as part of your request. If the two encryption keys match, the object is decrypted and your object data is returned.

While StorageGRID manages all object encryption and decryption operations, you must manage the encryption keys you provide.

**Attention:** The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object.

### Using SSE

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

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

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

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

### Using SSE-C

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

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

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

- GET Object
- HEAD Object

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

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

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

- You must use `https`.
  - Attention:** StorageGRID rejects any requests made over `http` when using SSE-C. For security considerations, you should consider any key you send accidentally using `http` to be compromised. Discard the key, and rotate as appropriate.
- The ETag in the response is not the MD5 of the object data.
- You must manage the mapping of encryption keys to objects. StorageGRID does not store encryption keys. You are responsible for tracking the encryption key you provide for each object.
- If your bucket is versioning-enabled, each object version should have its own encryption key. You are responsible for tracking the encryption key used for each object version.
- Because you manage encryption keys on the client side, you must also manage any additional safeguards, such as key rotation, on the client side.
  - Attention:** The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object.
- If CloudMirror replication is configured for the bucket, you cannot ingest SSE-C objects. The ingest operation will fail.

#### Related concepts

- [GET Object](#) on page 31
- [HEAD Object](#) on page 33
- [PUT Object](#) on page 36
- [PUT Object - Copy](#) on page 40
- [Initiate Multipart Upload](#) on page 44
- [Upload Part](#) on page 46
- [Upload Part - Copy](#) on page 47

#### Related information

[Amazon S3 Developer Guide: Protecting Data Using Server-Side Encryption with Customer-Provided Encryption Keys \(SSE-C\)](#)

## GET Object

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

### Request headers for server-side encryption with customer-provided encryption keys (SSE-C)

Use all three of these headers if the object is encrypted with a unique key that you provided.

- `x-amz-server-side-encryption-customer-algorithm`: Specify AES256.
- `x-amz-server-side-encryption-customer-key`: Specify your encryption key for the object.
- `x-amz-server-side-encryption-customer-key-MD5`: Specify the MD5 digest of the object's encryption key.

**Attention:** The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object. Before using customer-provided keys to secure object data, review the considerations in “Using server-side encryption.”

### UTF-8 characters in user metadata

StorageGRID does not parse or interpret escaped UTF-8 characters in user-defined metadata. GET requests for an object with escaped UTF-8 characters in user-defined metadata do not return the `x-amz-missing-meta` header if the key name or value includes unprintable characters.

### Unsupported request header

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

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

### Versioning

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

### Behavior of GET Object for Cloud Storage Pool objects

If an object has been stored in a Cloud Storage Pool (see the instructions for administering StorageGRID), the behavior of a GET Object request depends on the state of the object. See “HEAD Object” for more details.

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

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



State of object	Behavior of GET Object
Object fully restored to the Cloud Storage Pool	200 OK A copy of the object is retrieved.

### Multipart or segmented objects in a Cloud Storage Pool

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

In these cases:

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

#### Related concepts

[Using server-side encryption](#) on page 30

[HEAD Object](#) on page 33

[POST Object restore](#) on page 35

[S3 operations tracked in the audit logs](#) on page 91

#### Related information

[Administering StorageGRID](#)

## HEAD Object

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

### Request headers for server-side encryption with customer-provided encryption keys (SSE-C)

Use all three of these headers if the object is encrypted with a unique key that you provided.

- `x-amz-server-side-encryption-customer-algorithm`: Specify AES256.
- `x-amz-server-side-encryption-customer-key`: Specify your encryption key for the object.
- `x-amz-server-side-encryption-customer-key-MD5`: Specify the MD5 digest of the object's encryption key.

**Attention:** The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object. Before using customer-provided keys to secure object data, review the considerations in “Using server-side encryption.”

### UTF-8 characters in user metadata

StorageGRID does not parse or interpret escaped UTF-8 characters in user-defined metadata. HEAD requests for an object with escaped UTF-8 characters in user-defined metadata do not return the `x-amz-missing-meta` header if the key name or value includes unprintable characters.

### Unsupported request header

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

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

### Response headers for Cloud Storage Pool objects

If the object is stored in a Cloud Storage Pool (see the instructions for administering StorageGRID), the following response headers are returned:

- `x-amz-storage-class: GLACIER`
- `x-amz-restore`

The response headers provide information about the state of an object as it is moved to a Cloud Storage Pool, optionally transitioned to a non-retrievable state, and restored.

State of object	Response to HEAD object
Object ingested into StorageGRID but not yet evaluated by ILM, or object stored in a traditional storage pool or using erasure coding	200 OK (No special response header is returned.)
Object in Cloud Storage Pool but not yet transitioned to a non-retrievable state	200 OK <code>x-amz-storage-class: GLACIER</code> <code>x-amz-restore: ongoing-request="false", expiry-date="Sat, 23 July 20 2030 00:00:00 GMT"</code> Until the object is transitioned to a non-retrievable state, the value for <code>expiry-date</code> is set to some distant time in the future. The exact time of transition is not controlled by the StorageGRID system.
Object has transitioned to non-retrievable state, but at least one copy also exists on the grid	200 OK <code>x-amz-storage-class: GLACIER</code> <code>x-amz-restore: ongoing-request="false", expiry-date="Sat, 23 July 20 2030 00:00:00 GMT"</code> The value for <code>expiry-date</code> is set to some distant time in the future.  <b>Note:</b> If the copy on the grid is not available (for example, a Storage Node is down), you must issue a POST Object restore request to restore the copy from the Cloud Storage Pool before you can successfully retrieve the object.
Object transitioned to a non-retrievable state, and no copy exists on the grid	200 OK <code>x-amz-storage-class: GLACIER</code>
Object in process of being restored from a non-retrievable state	200 OK <code>x-amz-storage-class: GLACIER</code> <code>x-amz-restore: ongoing-request="true"</code>

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

### Multipart or segmented objects in a Cloud Storage Pool

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

### Versioning

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

### Related concepts

[Using server-side encryption](#) on page 30

[POST Object restore](#) on page 35

[S3 operations tracked in the audit logs](#) on page 91

### Related information

[Administering StorageGRID](#)

## POST Object restore

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

### Supported request type

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

### Versioning

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

### Behavior of POST Object restore on Cloud Storage Pool objects

If an object has been stored in a Cloud Storage Pool (see the instructions for administering StorageGRID), a POST Object restore request has the following behavior, based on the state of the object. See “HEAD Object” for more details.

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

State of object	Behavior of POST Object restore
Object ingested into StorageGRID but not yet evaluated by ILM, or object is not in a Cloud Storage Pool	403 Forbidden, InvalidObjectState
Object in Cloud Storage Pool but not yet transitioned to a non-retrievable state	200 OK No changes are made. <b>Note:</b> Before an object has been transitioned to a non-retrievable state, you cannot change its <code>expiry-date</code> .
Object transitioned to a non-retrievable state	202 Accepted Restores a retrievable copy of the object to the Cloud Storage Pool for the number of days specified in the request body. At the end of this period, the object is returned to a non-retrievable state. Optionally, use the <code>Tier</code> request element to determine how long the restore job will take to finish ( <code>Expedited</code> , <code>Standard</code> , or <code>Bulk</code> ). If you do not specify <code>Tier</code> , the <code>Standard</code> tier is used. <b>Attention:</b> If an object has been transitioned to S3 Glacier Deep Archive or the Cloud Storage Pool uses Azure Blob Storage, you cannot restore it using the <code>Expedited</code> tier. The following error is returned 403 Forbidden, InvalidTier: Retrieval option is not supported by this storage class.
Object in process of being restored from a non-retrievable state	409 Conflict, RestoreAlreadyInProgress
Object fully restored to the Cloud Storage Pool	200 OK <b>Note:</b> If an object has been restored to a retrievable state, you can change its <code>expiry-date</code> by reissuing the POST Object restore request with a new value for <code>Days</code> . The restoration date is updated relative to the time of the request.

**Related concepts**

[HEAD Object](#) on page 33

[S3 operations tracked in the audit logs](#) on page 91

**Related information**

[Administering StorageGRID](#)

**PUT Object**

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

**Resolving conflicts**

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

## Object size

StorageGRID supports objects up to 5 TB in size.

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

## User metadata size

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

## UTF-8 characters in user metadata

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

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

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

## Object tag limits

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

## Object ownership

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

## Supported request headers

The following request headers are supported:

- `Cache-Control`
- `Content-Disposition`
- `Content-Encoding`  
When you specify `aws-chunked` for `Content-Encoding`, StorageGRID does not verify the value that you provide for `x-amz-decoded-content-length` against the object.
- `Content-Language`
- `Content-Length`
- `Content-Type`
- `Expires`
- `x-amz-meta-`, followed by a name-value pair containing user-defined metadata

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

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

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

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

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

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

**Note:** An ILM rule cannot use both a **User Defined Creation Time** for the Reference Time and the Balanced or Strict options for Ingest Behavior. An error is returned when the ILM rule is created.

- `x-amz-tagging`

### Unsupported request headers

The following request headers are not supported:

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

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

- `Transfer-Encoding`
- `x-amz-website-redirect-location`

### Storage class options

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

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

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

- `STANDARD` (Default)
  - **Dual commit:** If the ILM rule specifies the Dual commit option for Ingest Behavior, as soon as an object is ingested a second copy of that object is created and distributed to a different Storage Node (dual commit). When the ILM is evaluated, StorageGRID determines if these initial interim copies satisfy the placement instructions in the rule. If they do not, new object copies might need to be made in different locations and the initial interim copies might need to be deleted.
  - **Balanced:** If the ILM rule specifies the Balanced option and StorageGRID cannot immediately make all copies specified in the rule, StorageGRID makes two interim copies on different Storage Nodes.

If StorageGRID can immediately create all object copies specified in the ILM rule (synchronous placement), the `x-amz-storage-class` header has no effect.

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

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

Using the `REDUCED_REDUNDANCY` option is not recommended in other circumstances.

`REDUCED_REDUNDANCY` increases the risk the loss of object data during ingest. For example, you might lose data if the single copy is initially stored on a Storage Node that fails before ILM evaluation can occur.

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

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

### Request headers for server-side encryption

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

- **SSE:** Use the following header if you want to encrypt the object with a unique key managed by StorageGRID.
  - `x-amz-server-side-encryption`
- **SSE-C:** Use all three of these headers if you want to encrypt the object with a unique key that you provide and manage.
  - `x-amz-server-side-encryption-customer-algorithm`: Specify AES256.
  - `x-amz-server-side-encryption-customer-key`: Specify your encryption key for the new object.
  - `x-amz-server-side-encryption-customer-key-MD5`: Specify the MD5 digest of the new object's encryption key.

**Attention:** The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object. Before using customer-provided keys to secure object data, review the considerations in “Using server-side encryption.”

### Versioning

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

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

**Related concepts**

- [Operations on buckets](#) on page 21
- [S3 operations tracked in the audit logs](#) on page 91
- [Using server-side encryption](#) on page 30
- [Configuring client connections](#) on page 9

**Related information**

- [Administering StorageGRID](#)

**PUT Object - Copy**

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

**Resolving conflicts**

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

**Object size**

StorageGRID supports objects up to 5 TB in size.

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

**UTF-8 characters in user metadata**

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

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

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

**Supported request headers**

The following request headers are supported:

- `Content-Type`
- `x-amz-copy-source`
- `x-amz-copy-source-if-match`
- `x-amz-copy-source-if-none-match`
- `x-amz-copy-source-if-unmodified-since`
- `x-amz-copy-source-if-modified-since`
- `x-amz-meta-`, followed by a name-value pair containing user-defined metadata



- `x-amz-metadata-directive`: The default value is `COPY`, which enables you to copy the object and associated metadata.  
You can specify `REPLACE` to overwrite the existing metadata when copying the object, or to update the object metadata.
- `x-amz-storage-class`
- `x-amz-tagging-directive`: The default value is `COPY`, which enables you to copy the object and all tags.  
You can specify `REPLACE` to overwrite the existing tags when copying the object, or to update the tags.

### Unsupported request headers

The following request headers are not supported:

- `Cache-Control`
- `Content-Disposition`
- `Content-Encoding`
- `Content-Language`
- `Expires`
- `x-amz-website-redirect-location`

### Storage class options

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

- `STANDARD`  
(Default) Specifies a dual-commit ingest operation when the ILM rule uses the Dual commit option, or when the Balanced option falls back to creating interim copies.
- `REDUCED_REDUNDANCY`  
Specifies a single-commit ingest operation when the ILM rule uses the Dual commit option, or when the Balanced option falls back to creating interim copies.  
  
**Note:** You cannot use the `REDUCED_REDUNDANCY` option if you are ingesting an object into an S3 compliant bucket. This is to ensure that compliance requirements are satisfied (two copies of each object exist) before the object is evaluated by the active ILM policy. See the instructions for administering StorageGRID.

### Request headers for server-side encryption

If you use server-side encryption, the request headers you provide depend on whether the source object is encrypted and on whether you plan to encrypt the target object.

- If the source object is encrypted using a customer-provided key (SSE-C), you must include the following three headers in the PUT Object - Copy request, so the object can be decrypted and then copied:
  - `x-amz-copy-source-server-side-encryption-customer-algorithm`: Specify `AES256`.
  - `x-amz-copy-source-server-side-encryption-customer-key`: Specify the encryption key you provided when you created the source object.

- `x-amz-copy-source-server-side-encryption-customer-key-MD5`: Specify the MD5 digest you provided when you created the source object.
- If you want to encrypt the target object (the copy) with a unique key that you provide and manage, include the following three headers:
  - `x-amz-server-side-encryption-customer-algorithm`: Specify AES256.
  - `x-amz-server-side-encryption-customer-key`: Specify a new encryption key for the target object.
  - `x-amz-server-side-encryption-customer-key-MD5`: Specify the MD5 digest of the new encryption key.

**Attention:** The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object. Before using customer-provided keys to secure object data, review the considerations in “Using server-side encryption.”

- If you want to encrypt the target object (the copy) with a unique key managed by StorageGRID (SSE), include this header in the PUT Object - Copy request:
  - `x-amz-server-side-encryption`

**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`.

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

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

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

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

This means that if the ILM rule uses the Strict option for ingest behavior, no action is taken if the required object placements cannot be made (for example, because a newly required location is unavailable). The updated object retains its current placement until the required placement is possible.

### Versioning

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

### Related concepts

[Using server-side encryption](#) on page 30

[S3 operations tracked in the audit logs](#) on page 91

[PUT Object](#) on page 36

#### Related information

[Administering StorageGRID](#)

## Operations for multipart uploads

This section describes how StorageGRID supports operations for multipart uploads.

The following conditions and notes apply to all multipart upload operations:

- You should not exceed 1,000 concurrent multipart uploads to a single bucket because the results of List Multipart Uploads queries for that bucket might return incomplete results.
- StorageGRID does not enforce multipart size limits. You must follow these guidelines:
  - As a general rule, each multipart part except the last must be between 5 MB and 5 GB. However, it is best to use part sizes that are as large as possible. For example, use part sizes of 5 GB for a 100 GB object. Since each part is considered a unique object, using large part sizes reduces StorageGRID system metadata overhead.
  - The last part can be less than 5 MB.
  - For objects of less than 5 GB, consider using non-multipart upload instead.
- ILM is evaluated for each part of a multipart object as it ingested and for the object as a whole when the multipart upload completes, if the ILM rule uses the Strict or Balanced ingest behavior. You should be aware of how this affects object and part placement:
  - If ILM changes while an S3 multipart upload is in progress, when the multipart upload completes some parts of the object might not meet current ILM requirements. Any part that is not placed correctly is queued for ILM re-evaluation, and is moved to the correct location later.
  - When evaluating ILM for a part, StorageGRID filters on the size of the part, not the size of the object. This means that parts of an object can be stored in locations that do not meet ILM requirements for the object as a whole. For example, if a rule specifies that all objects 10 GB or larger are stored at DC1 while all smaller objects are stored at DC2, at ingest each 1 GB part of a 10-part multipart upload is stored at DC2. When ILM is evaluated for the object as a whole, all parts of the object are moved to DC1.
- All of the multipart upload operations support StorageGRID consistency controls.
- As required, you can use server-side encryption with multipart uploads. To use SSE (server-side encryption with StorageGRID-managed keys), you include the `x-amz-server-side-encryption` request header in the Initiate Multipart Upload request only. To use SSE-C (server-side encryption with customer-provided keys), you specify the same three encryption key request headers in the Initiate Multipart Upload request and in each subsequent Upload Part request.

Operation	Implementation
List Multipart Uploads	See <a href="#">List Multipart Uploads</a> on page 44
Initiate Multipart Upload	See <a href="#">Initiate Multipart Upload</a> on page 44
Upload Part	See <a href="#">Upload Part</a> on page 46
Upload Part - Copy	See <a href="#">Upload Part - Copy</a> on page 47

Operation	Implementation
Complete Multipart Upload	See <a href="#">Complete Multipart Upload</a> on page 48
Abort Multipart Upload	Implemented with all Amazon S3 REST API behavior
List Parts	Implemented with all Amazon S3 REST API behavior

#### Related concepts

[Consistency controls](#) on page 14

[Using server-side encryption](#) on page 30

## List Multipart Uploads

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

The following request parameters are supported:

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

The `delimiter` request parameter is not supported.

#### Versioning

Multipart upload consists of separate operations for initiating the upload, listing uploads, uploading parts, assembling the uploaded parts, and completing the upload. When the Complete Multipart Upload operation is performed, that is the point when objects are created (and versioned if applicable).

## Initiate Multipart Upload

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

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

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

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

- STANDARD (Default)
  - **Dual commit:** If the ILM rule specifies the Dual commit option for Ingest Behavior, as soon as an object is ingested a second copy of that object is created and distributed to a different Storage Node (dual commit). When the ILM is evaluated, StorageGRID determines if these initial interim copies satisfy the placement instructions in the rule. If they do not, new object copies might need to be made in different locations and the initial interim copies might need to be deleted.

- **Balanced:** If the ILM rule specifies the Balanced option and StorageGRID cannot immediately make all copies specified in the rule, StorageGRID makes two interim copies on different Storage Nodes.  
If StorageGRID can immediately create all object copies specified in the ILM rule (synchronous placement), the `x-amz-storage-class` header has no effect.
- REDUCED\_REDUNDANCY
  - **Dual commit:** If the ILM rule specifies the Dual commit option for Ingest Behavior, StorageGRID creates a single interim copy as the object is ingested (single commit).
  - **Balanced:** If the ILM rule specifies the Balanced option, StorageGRID makes a single interim copy only if the system cannot immediately make all copies specified in the rule. If StorageGRID can perform synchronous placement, this header has no effect.

The REDUCED\_REDUNDANCY option is best used when the ILM rule that matches the object creates a single replicated copy. In this case using REDUCED\_REDUNDANCY eliminates the unnecessary creation and deletion of an extra object copy for every ingest operation. Using the REDUCED\_REDUNDANCY option is not recommended in other circumstances. REDUCED\_REDUNDANCY increases the risk the loss of object data during ingest. For example, you might lose data if the single copy is initially stored on a Storage Node that fails before ILM evaluation can occur.

Specifying REDUCED\_REDUNDANCY only affects how many copies are created when an object is first ingested. It does not affect how many copies of the object are made when the object is evaluated by the active ILM policy, and does not result in data being stored at lower levels of redundancy in the StorageGRID system.

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

The following request headers are supported:

- Content-Type
- x-amz-meta-, followed by a name-value pair containing user-defined metadata  
When specifying the name-value pair for user-defined metadata, use this general format:

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

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

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

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

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

**Note:** For information on how StorageGRID handles UTF-8 characters, see the documentation for PUT Object.

## Request headers for server-side encryption

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

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

**Attention:** The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object. Before using customer-provided keys to secure object data, review the considerations in “Using server-side encryption.”

### Unsupported request headers

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

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

### Versioning

Multipart upload consists of separate operations for initiating the upload, listing uploads, uploading parts, assembling the uploaded parts, and completing the upload. Objects are created (and versioned if applicable) when the Complete Multipart Upload operation is performed.

### Related concepts

[Using server-side encryption](#) on page 30

[PUT Object](#) on page 36

### Related information

[Administering StorageGRID](#)

## Upload Part

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

### Supported request header

The following request header is supported:

- `Content-Length`

### Request headers for server-side encryption

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

- `x-amz-server-side-encryption-customer-algorithm`: Specify AES256.

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

**Attention:** The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object. Before using customer-provided keys to secure object data, review the considerations in “Using server-side encryption.”

## Versioning

Multipart upload consists of separate operations for initiating the upload, listing uploads, uploading parts, assembling the uploaded parts, and completing the upload. Objects are created (and versioned if applicable) when the Complete Multipart Upload operation is performed.

### Related concepts

[Using server-side encryption](#) on page 30

## Upload Part - Copy

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

The Upload Part - Copy operation is implemented with all Amazon S3 REST API behavior.

This request reads and writes the object data specified in `x-amz-copy-source-range` within the StorageGRID system.

The following request headers are supported:

- `x-amz-copy-source-if-match`
- `x-amz-copy-source-if-none-match`
- `x-amz-copy-source-if-unmodified-since`
- `x-amz-copy-source-if-modified-since`

### Request headers for server-side encryption

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

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

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

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

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

**Attention:** The encryption keys you provide are never stored. If you lose an encryption key, you lose the corresponding object. Before using customer-provided keys to secure object data, review the considerations in “Using server-side encryption.”

## Versioning

Multipart upload consists of separate operations for initiating the upload, listing uploads, uploading parts, assembling the uploaded parts, and completing the upload. Objects are created (and versioned if applicable) when the Complete Multipart Upload operation is performed.

## Complete Multipart Upload

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

### Resolving conflicts

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

### Object size

StorageGRID supports objects up to 5 TB in size.

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

### Request headers

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

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

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

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

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



## Versioning

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

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

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

**Note:** When versioning is enabled for a bucket, completing a multipart upload always creates a new version, even if there are concurrent multipart uploads completed on the same object key. When versioning is not enabled for a bucket, it is possible to initiate a multipart upload and then have another multipart upload initiate and complete first on the same object key. On non-versioned buckets, the multipart upload that completes last takes precedence.

## Failed replication, notification, or metadata notification

If the bucket where the multipart upload occurs is configured for a platform service, multipart upload succeeds even if the associated replication or notification action fails.

If this occurs, an alarm is raised in the Grid Manager on Total Events (SMTT). The Last Event message displays “Failed to publish notifications for *bucket-name object key*” for the last object whose notification failed. (To see this message, select **Nodes > Storage Node > Events**. View Last Event at the top of the table.) Event messages are also listed in `/var/local/log/bycasterr.log`.

A tenant can trigger the failed replication or notification by updating the object's metadata or tags. A tenant can resubmit the existing values to avoid making unwanted changes.

## Error responses

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

### Supported S3 API error codes

Name	HTTP status
AccessDenied	403 Forbidden
BadDigest	400 Bad Request
BucketAlreadyExists	409 Conflict
BucketNotEmpty	409 Conflict
IncompleteBody	400 Bad Request
InternalServerError	500 Internal Server Error
InvalidAccessKeyId	403 Forbidden
InvalidArgument	400 Bad Request
InvalidBucketName	400 Bad Request
InvalidDigest	400 Bad Request
InvalidEncryptionAlgorithmError	400 Bad Request
InvalidPart	400 Bad Request

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

#### **StorageGRID custom error codes**

<b>Name</b>	<b>Description</b>	<b>HTTP status</b>
XBucketLifecycleNotAllowed	Bucket lifecycle configuration is not allowed in a compliant bucket	400 Bad Request
XBucketPolicyParseException	Failed to parse received bucket policy JSON.	400 Bad Request
XComplianceConflict	Operation denied because of compliance settings.	403 Forbidden
XComplianceReducedRedundancyForbidden	Reduced redundancy is not allowed in compliant bucket	400 Bad Request
XMaxBucketPolicyLengthExceeded	Your policy exceeds the maximum allowed bucket policy length.	400 Bad Request

<b>Name</b>	<b>Description</b>	<b>HTTP status</b>
XMissingInternalRequestHeader	Missing a header of an internal request.	400 Bad Request
XNoSuchBucketCompliance	The specified bucket does not have compliance enabled.	404 Not Found
XNotAcceptable	The request contains one or more accept headers that could not be satisfied.	406 Not Acceptable
XNotImplemented	The request you provided implies functionality that is not implemented.	501 Not Implemented

## StorageGRID S3 REST API operations

---

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

### PUT Bucket request modifications for compliance

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

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

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

When compliance is enabled for a bucket, you cannot:

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

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

- Delete the object

When using the LocationConstraint element, you must specify the exact name of a region that has been defined using the Grid Manager or the Grid Management API. Contact your system administrator if you do not know the region name you should use.

**Note:** An error will occur if your PUT Bucket request uses a region that has not been defined in StorageGRID.

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

#### Request example

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

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

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

Name	Description
RetentionPeriodMinutes	<p>The length of the retention period for objects added to this bucket, in minutes. The retention period starts when the object is ingested into the grid.</p> <p>You can specify any non-zero positive integer as the value.</p> <p><b>Attention:</b> After the bucket's retention period is set, you cannot decrease that value; you can only increase it.</p> <p><b>Note:</b> If you want to test compliance settings before implementing this feature in a production-level grid, you can set RetentionPeriodMinutes to a small value. However, be aware that it will take longer than an hour to automatically delete objects in compliant buckets, even if the retention period is very short. This is because of the delays due to ILM scanning.</p>
LegalHold	<ul style="list-style-type: none"> <li>• True: This bucket is currently under a legal hold. Objects in this bucket cannot be deleted until the legal hold is lifted, even if their retention period has expired.</li> <li>• False: This bucket is not currently under a legal hold. Objects in this bucket can be deleted when their retention period expires.</li> </ul>
AutoDelete	<ul style="list-style-type: none"> <li>• True: The objects in this bucket will be deleted automatically when their retention period expires, unless the bucket is under a legal hold.</li> <li>• False: The objects in this bucket will not be deleted automatically when the retention period expires. You must delete these objects manually if you need to delete them.</li> </ul>

### Error responses

If a PUT Bucket request attempts to remove the compliance settings of an existing compliant bucket or add compliance settings to an existing non-compliant bucket, the HTTP status code for the response is 409 Conflict.

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

### Related information

[Administering StorageGRID](#)

[Using tenant accounts](#)

## GET Bucket compliance request

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

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

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

### Request example

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

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

### Response example

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

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

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

Name	Description
RetentionPeriodMinutes	The length of the retention period for objects added to this bucket, in minutes. The retention period starts when the object is ingested into the grid.
LegalHold	<ul style="list-style-type: none"> <li>• True: This bucket is currently under a legal hold. Objects in this bucket cannot be deleted until the legal hold is lifted, even if their retention period has expired.</li> <li>• False: This bucket is not currently under a legal hold. Objects in this bucket can be deleted when their retention period expires.</li> </ul>
AutoDelete	<ul style="list-style-type: none"> <li>• True: The objects in this bucket will be deleted automatically when their retention period expires, unless the bucket is under a legal hold.</li> <li>• False: The objects in this bucket will not be deleted automatically when the retention period expires. You must delete these objects manually if you need to delete them.</li> </ul>

### Error responses

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

**Related information**[Administering StorageGRID](#)[Using tenant accounts](#)

## PUT Bucket compliance request

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

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

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

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

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

**Request example**

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

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

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

Name	Description
RetentionPeriodMinutes	<p>The length of the retention period for objects added to this bucket, in minutes. The retention period starts when the object is ingested into the grid.</p> <p><b>Attention:</b> When specifying a new value for <code>RetentionPeriodMinutes</code>, you must specify a value that is equal to or greater than the bucket's current retention period. After the bucket's retention period is set, you cannot decrease that value; you can only increase it.</p>
LegalHold	<ul style="list-style-type: none"> <li>True: This bucket is currently under a legal hold. Objects in this bucket cannot be deleted until the legal hold is lifted, even if their retention period has expired.</li> <li>False: This bucket is not currently under a legal hold. Objects in this bucket can be deleted when their retention period expires.</li> </ul>

Name	Description
AutoDelete	<ul style="list-style-type: none"> <li>• True: The objects in this bucket will be deleted automatically when their retention period expires, unless the bucket is under a legal hold.</li> <li>• False: The objects in this bucket will not be deleted automatically when the retention period expires. You must delete these objects manually if you need to delete them.</li> </ul>

### Consistency level for compliance settings

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

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

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

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

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

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

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

### Error responses

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

### Related references

[PUT Bucket request modifications for compliance](#) on page 52



**Related information**[Using tenant accounts](#)[Administering StorageGRID](#)

## GET Bucket consistency request

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

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

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

**Request example**

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

**Response**

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

Consistency control	Description
all	All nodes receive the data immediately, or the request will fail.
strong-global	Guarantees read-after-write consistency for all client requests across all sites.
strong-site	Guarantees read-after-write consistency for all client requests within a site.
read-after-new-write	(Default) Provides read-after-write consistency for new objects and eventual consistency for object updates. Offers high availability and data protection guarantees. Matches AWS S3 consistency guarantees.  <b>Note:</b> If your application uses HEAD requests on objects that do not exist, you might receive a high number of 500 Internal Server errors if one or more Storage Nodes are unavailable. To prevent these errors, set the consistency control to “available” unless you require consistency guarantees similar to AWS S3.
available (eventual consistency for HEAD operations)	Behaves the same as the “read-after-new-write” consistency level, but only provides eventual consistency for HEAD operations. Offers higher availability for HEAD operations than “read-after-new-write” if Storage Nodes are unavailable. Differs from AWS S3 consistency guarantees for HEAD operations only.
weak	Provides eventual consistency and high availability, with minimal data protection guarantees, especially if a Storage Node fails or is unavailable. Suitable only for write-heavy workloads that require high availability, do not require read-after-write consistency, and can tolerate the potential loss of data if a node fails.

## Response example

```

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

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

```

## Related concepts

[Consistency controls](#) on page 14

# PUT Bucket consistency request

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

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

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

## Request

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

Consistency control	Description
all	All nodes receive the data immediately, or the request will fail.
strong-global	Guarantees read-after-write consistency for all client requests across all sites.
strong-site	Guarantees read-after-write consistency for all client requests within a site.
read-after-new-write	<p>(Default) Provides read-after-write consistency for new objects and eventual consistency for object updates. Offers high availability and data protection guarantees. Matches AWS S3 consistency guarantees.</p> <p><b>Note:</b> If your application uses HEAD requests on objects that do not exist, you might receive a high number of 500 Internal Server errors if one or more Storage Nodes are unavailable. To prevent these errors, set the consistency control to “available” unless you require consistency guarantees similar to AWS S3.</p>
available (eventual consistency for HEAD operations)	Behaves the same as the “read-after-new-write” consistency level, but only provides eventual consistency for HEAD operations. Offers higher availability for HEAD operations than “read-after-new-write” if Storage Nodes are unavailable. Differs from AWS S3 consistency guarantees for HEAD operations only.
weak	Provides eventual consistency and high availability, with minimal data protection guarantees, especially if a Storage Node fails or is unavailable. Suitable only for write-heavy workloads that require high availability, do not require read-after-write consistency, and can tolerate the potential loss of data if a node fails.

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

### Request example

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

### Related concepts

[Consistency controls](#) on page 14

## GET Bucket last access time request

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

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

### Request example

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

### Response example

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

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

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

## PUT Bucket last access time request

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

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

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

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

- GET Object, GET Object ACL, GET Object Tagging, and HEAD Object requests do not update last access time. The object is not added to queues for information lifecycle management (ILM) evaluation.
- PUT Object - Copy and PUT Object Tagging requests that update only the metadata also update last access time. The object is added to queues for ILM evaluation.
- If updates to last access time are disabled for the source bucket, PUT Object - Copy requests do not update last access time for the source bucket. The object that was copied is not added to queues for ILM evaluation for the source bucket. However, for the destination, PUT Object - Copy requests always update last access time. The copy of the object is added to queues for ILM evaluation.
- Complete Multipart Upload requests update last access time. The completed object is added to queues for ILM evaluation.

### Request examples

This example enables last access time for a bucket.

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

This example disables last access time for a bucket.

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

### Related information

[Using tenant accounts](#)

## DELETE Bucket metadata notification configuration request

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

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

### Request example

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

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

## GET Bucket metadata notification configuration request

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

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

### Request example

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

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

### Response

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

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

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

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

Name	Description	Required
Rule	<p>Container tag for a rule that identifies the objects whose metadata should be added to a specified index.</p> <p>Rules with overlapping prefixes are rejected.</p> <p>Included in the <code>MetadataNotificationConfiguration</code> element.</p>	Yes
ID	<p>Unique identifier for the rule.</p> <p>Included in the Rule element.</p>	No
Status	<p>Status can be 'Enabled' or 'Disabled'. No action is taken for rules that are disabled.</p> <p>Included in the Rule element.</p>	Yes
Prefix	<p>Objects that match the prefix are affected by the rule, and their metadata is sent to the specified destination.</p> <p>To match all objects, specify an empty prefix.</p> <p>Included in the Rule element.</p>	Yes
Destination	<p>Container tag for the destination of a rule.</p> <p>Included in the Rule element.</p>	Yes
Urn	<p>URN of the destination where object metadata is sent. Must be the URN of a StorageGRID endpoint with the following properties:</p> <ul style="list-style-type: none"> <li>• <code>es</code> must be the third element.</li> <li>• The URN must end with the index and type where the metadata is stored, in the form <code>domain-name/myindex/mytype</code>.</li> </ul> <p>Endpoints are configured using the Tenant Manager or Tenant Management API. They take the following form:</p> <ul style="list-style-type: none"> <li>• <code>arn:aws:es:region:account-ID:domain/mydomain/myindex/mytype</code></li> <li>• <code>urn:mysite:es:::mydomain/myindex/mytype</code></li> </ul> <p>The endpoint must be configured before the configuration XML is submitted, or configuration will fail with a 404 error.</p> <p>Urn is included in the Destination element.</p>	Yes

### Response example

The XML included between the `<MetadataNotificationConfiguration>` `</MetadataNotificationConfiguration>` tags shows how integration with a search integration

endpoint is configured for the bucket. In this example, object metadata is being sent to an Elasticsearch index named `current` and type named `2017` that is hosted in an AWS domain named `records`.

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

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

### Related information

[Using tenant accounts](#)

## PUT Bucket metadata notification configuration request

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

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

### Request

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

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

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

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

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

```

</Rule>
...
</MetadataNotificationConfiguration>

```

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

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



Name	Description	Required
Urn	<p>URN of the destination where object metadata is sent. Must be the URN of a StorageGRID endpoint with the following properties:</p> <ul style="list-style-type: none"> <li>• <code>es</code> must be the third element.</li> <li>• The URN must end with the index and type where the metadata is stored, in the form <code>domain-name/myindex/mytype</code>.</li> </ul> <p>Endpoints are configured using the Tenant Manager or Tenant Management API. They take the following form:</p> <ul style="list-style-type: none"> <li>• <code>arn:aws:es:region:account-ID:domain/mydomain/myindex/mytype</code></li> <li>• <code>urn:mysite:es:::mydomain/myindex/mytype</code></li> </ul> <p>The endpoint must be configured before the configuration XML is submitted, or configuration will fail with a 404 error.</p> <p>Urn is included in the Destination element.</p>	Yes

### Request examples

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

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

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

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

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

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

```

<Status>Enabled</Status>
<Prefix>/videos</Prefix>
<Destination>
  <Urn>arn:aws:es:us-west-1:22222222:domain/es-domain/graphics/videotype</Urn>
</Destination>
</Rule>
</MetadataNotificationConfiguration>

```

### Related information

[Using tenant accounts](#)

## JSON generated by the search integration service

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

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

```

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

```

## Object metadata included in metadata notifications

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

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

Type	Item name	Description
Bucket and object information	bucket	Name of the bucket
	key	Object key name
	versionID	Object version, for objects in versioned buckets
	region	Bucket region, for example <code>us-east-1</code>
System metadata	size	Object size (in bytes) as visible to an HTTP client
	md5	Object hash
User metadata	metadata <i>key:value</i>	All user metadata for the object, as key-value pairs
Tags	tags <i>key:value</i>	All object tags defined for the object, as key-value pairs

## GET Storage Usage request

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

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

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

### Request example

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

### Response example

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

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

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

### Versioning

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

## Bucket and group access policies

---

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

### Related concepts

[S3 bucket policy examples](#) on page 80

[S3 group policy examples](#) on page 83

## Access policy overview

There are two kinds of access policies supported by StorageGRID.

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

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

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

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

Each policy element is used for a specific function:

Element	Description
Sid	The Sid element is optional. The Sid is only intended as a description for the user. It is stored but not interpreted by the StorageGRID system.
Effect	Use the Effect element to establish whether the specified operations are allowed or denied. You must identify operations you allow (or deny) on buckets or objects using the supported Action element keywords.

Element	Description
Principal/ NotPrincipal	You can allow users, groups, and accounts to access specific resources and perform specific actions. If no S3 signature is included in the request, anonymous access is allowed by specifying the wildcard character (*) as the principal. By default, only the account root has access to resources owned by the account.  You only need to specify the Principal element in a bucket policy. For group policies, the group to which the policy is attached is the implicit Principal element.
Resource/ NotResource	The Resource element identifies buckets and objects. You can allow or deny permissions to buckets and objects using the Amazon Resource Name (ARN) to identify the resource.
Action/NotAction	The Action and Effect elements are the two components of permissions. When a group requests a resource, they are either granted or denied access to the resource. Access is denied unless you specifically assign permissions, but you can use explicit deny to override a permission granted by another policy.
Condition	The Condition element is optional. Conditions allow you to build expressions to determine when a policy should be applied.

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

```
s3:*Object
```

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

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

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

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

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

```
    ]
  }
```

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

#### Related information

[Using tenant accounts](#)

## Consistency control settings for policies

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

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

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

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

## Using the ARN in policy statements

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

- Use this syntax to specify the S3 resource ARN:

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

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

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

Other considerations:

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

[RFC 2141 URN Syntax](#)

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

## Specifying resources in a policy

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

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

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

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

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

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

### Related concepts

[Specifying variables in a policy](#) on page 77

## Specifying principals in a policy

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

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

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

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

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

- You can specify just the account root:

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

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

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

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

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

- You can specify an anonymous principal:

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

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

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

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

## Specifying permissions in a policy

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

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

### Permissions that apply to buckets

Permissions	S3 REST API operations	Custom for StorageGRID
s3:CreateBucket	PUT Bucket	
s3>DeleteBucket	DELETE Bucket	
s3>DeleteBucketMetadataNotification	DELETE Bucket metadata notification configuration	Yes
s3>DeleteBucketPolicy	DELETE Bucket policy	
s3>DeleteReplicationConfiguration	DELETE Bucket replication	
s3:GetBucketAcl	GET Bucket ACL	
s3:GetBucketCompliance	GET Bucket compliance	Yes
s3:GetBucketConsistency	GET Bucket consistency	Yes
s3:GetBucketCORS	GET Bucket cors	
s3:GetBucketLastAccessTime	GET Bucket last access time	Yes
s3:GetBucketLocation	GET Bucket location	
s3:GetBucketMetadataNotification	GET Bucket metadata notification configuration	Yes
s3:GetBucketNotification	GET Bucket notification	



Permissions	S3 REST API operations	Custom for StorageGRID
s3:GetBucketPolicy	GET Bucket policy	
s3:GetBucketVersioning	GET Bucket versioning	
s3:GetLifecycleConfiguration	GET Bucket lifecycle	
s3:GetReplicationConfiguration	GET Bucket replication	
s3:ListAllMyBuckets	<ul style="list-style-type: none"> <li>GET Service</li> <li>GET Storage Usage</li> </ul>	Yes, for GET Storage Usage
s3:ListBucket	<ul style="list-style-type: none"> <li>GET Bucket (List Objects)</li> <li>HEAD Bucket</li> <li>POST Object restore</li> </ul>	
s3:ListBucketMultipartUploads	<ul style="list-style-type: none"> <li>List Multipart Uploads</li> <li>POST Object restore</li> </ul>	
s3:ListBucketVersions	GET Bucket versions	
s3:PutBucketCompliance	PUT Bucket compliance	Yes
s3:PutBucketConsistency	PUT Bucket consistency	Yes
s3:PutBucketCORS	<ul style="list-style-type: none"> <li>DELETE Bucket cors</li> <li>PUT Bucket cors</li> </ul>	
s3:PutBucketLastAccessTime	PUT Bucket last access time	Yes
s3:PutBucketMetadataNotification	PUT Bucket metadata notification configuration	Yes
s3:PutBucketNotification	PUT Bucket notification	
s3:PutBucketPolicy	PUT Bucket policy	
s3:PutBucketVersioning	PUT Bucket versioning	
s3:PutLifecycleConfiguration	<ul style="list-style-type: none"> <li>DELETE Bucket lifecycle</li> <li>PUT Bucket lifecycle</li> </ul>	
s3:PutReplicationConfiguration	PUT Bucket replication	

#### Permissions that apply to objects

Permissions	S3 REST API operations	Custom for StorageGRID
s3:AbortMultipartUpload	<ul style="list-style-type: none"> <li>Abort Multipart Upload</li> <li>POST Object restore</li> </ul>	
s3:DeleteObject	<ul style="list-style-type: none"> <li>DELETE Object</li> <li>DELETE Multiple Objects</li> <li>POST Object restore</li> </ul>	
s3:DeleteObjectTagging	DELETE Object Tagging	

Permissions	S3 REST API operations	Custom for StorageGRID
s3:DeleteObjectVersionTagging	DELETE Object Tagging (a specific version of the object)	
s3:DeleteObjectVersion	DELETE Object (a specific version of the object)	
s3:GetObject	<ul style="list-style-type: none"> <li>• GET Object</li> <li>• HEAD Object</li> <li>• POST Object restore</li> </ul>	
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, POST Object restore	
s3:PutObject	<ul style="list-style-type: none"> <li>• PUT Object</li> <li>• PUT Object - Copy</li> <li>• POST Object restore</li> <li>• Initiate Multipart Upload</li> <li>• Complete Multipart Upload</li> <li>• Upload Part</li> <li>• Upload Part - Copy</li> </ul>	
s3:PutObjectTagging	PUT Object Tagging	
s3:PutObjectVersionTagging	PUT Object Tagging (a specific version of the object)	
s3:PutOverwriteObject	<ul style="list-style-type: none"> <li>• PUT Object</li> <li>• PUT Object - Copy</li> <li>• PUT Object tagging</li> <li>• DELETE Object tagging</li> <li>• Complete Multipart Upload</li> </ul>	Yes
s3:RestoreObject	POST Object restore	

## Using the PutOverwriteObject permission

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

Possible settings for this permission include:

- **Allow:** The client can overwrite an object. This is the default setting.
- **Deny:** The client cannot overwrite an object. When set to Deny, the PutOverwriteObject permission works as follows:

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

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

### Related concepts

[S3 group policy examples](#) on page 83

## Specifying conditions in a policy

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

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

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

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

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

### Supported condition operators

Condition operators are categorized as follows:

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

Condition operators	Description
StringEquals	Compares a key to a string value based on exact matching (case sensitive).
StringNotEquals	Compares a key to a string value based on negated matching (case sensitive).
StringEqualsIgnoreCase	Compares a key to a string value based on exact matching (ignores case).
StringNotEqualsIgnoreCase	Compares a key to a string value based on negated matching (ignores case).
StringLike	Compares a key to a string value based on exact matching (case sensitive). Can include * and ? wildcard characters.
StringNotLike	Compares a key to a string value based on negated matching (case sensitive). Can include * and ? wildcard characters.
NumericEquals	Compares a key to a numeric value based on exact matching.
NumericNotEquals	Compares a key to a numeric value based on negated matching.
NumericGreaterThan	Compares a key to a numeric value based on “greater than” matching.
NumericGreaterThanEquals	Compares a key to a numeric value based on “greater than or equals” matching.
NumericLessThan	Compares a key to a numeric value based on “less than” matching.
NumericLessThanEquals	Compares a key to a numeric value based on “less than or equals” matching.
Bool	Compares a key to a Boolean value based on “true or false” matching.
IpAddress	Compares a key to an IP address or range of IP addresses.
NotIpAddress	Compares a key to an IP address or range of IP addresses based on negated matching.  <b>Attention:</b> If you use the NotIpAddress operator to specify an array (not a range) of IP addresses, the condition is always true, because the key does not equal at least one of the IP addresses. For this reason, if the statement has an Effect of Deny, access will be denied to all IP addresses in the array.
Null	Checks if a condition key is present in the current request context.

### Supported condition keys

Category	Applicable condition keys	Description
IP operators	aws:SourceIp	Will compare to the IP address from which the request was sent. Can be used for bucket or object operations.  <b>Note:</b> If the S3 request was sent through the Load Balancer service on Admin Nodes and Gateways Nodes, this will compare to the IP address upstream of the Load Balancer service.  <b>Note:</b> If a third-party, non-transparent load balancer is used, this will compare to the IP address of that load balancer. Any <code>X-Forwarded-For</code> header will be ignored since its validity cannot be ascertained.
Resource/Identity	aws:username	Will compare to the sender's username from which the request was sent. Can be used for bucket or object operations.
S3:ListBucket and S3:ListBucketVersions permissions	s3:delimiter	Will compare to the delimiter parameter specified in a GET Bucket or GET Bucket Object versions request.
	s3:max-keys	Will compare to the max-keys parameter specified in a GET Bucket or GET Bucket Object versions request.
	s3:prefix	Will compare to the prefix parameter specified in a GET Bucket or GET Bucket Object versions request.

## Specifying variables in a policy

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

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

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

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

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

Variable	Description
<code>\${aws:SourceIp}</code>	Uses the <code>SourceIp</code> key as the provided variable.

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

## Creating policies requiring special handling

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

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

Policy description	Policy type	Amazon behavior	StorageGRID behavior
Resource is a non-existent S3 bucket	Group	Valid	Same
Principal is a local group	Bucket	Invalid principal	Valid
Policy grants a non-owner account (including anonymous accounts) permissions to PUT objects	Bucket	Valid. Objects are owned by the creator account, and the bucket policy does not apply. The creator account must grant access permissions for the object using object ACLs.	Valid. Objects are owned by the bucket owner account. Bucket policy applies.

## Write-once-read-many (WORM) protection

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

To ensure that overwrites are always denied, you can:

- From the Grid Manager, set the **Prevent Client Modify** grid option to Enabled.
- Apply the following rules and S3 policies:
  - Add a PutOverwriteObject DENY operation to the S3 policy.
  - Add a DeleteObject DENY operation to the S3 policy.
  - Add a PUT Object ALLOW operation to the S3 policy.

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

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

**Situation A:** Concurrent writes (not guarded against)

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

**Situation B:** Sequential completed overwrites (guarded against)

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

### Related concepts

[Creating policies requiring special handling](#) on page 78

[How StorageGRID ILM rules manage objects](#) on page 16

[S3 group policy examples](#) on page 83

#### Related information

[Administering StorageGRID](#)

## S3 policy examples

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

### S3 bucket policy examples

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

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

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

#### Example: Allow everyone read-only access to a bucket

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

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

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

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

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

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "95390887230002558202"
      },
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::examplebucket",
        "arn:aws:s3:::examplebucket/*"
      ]
    }
  ]
}
```



```

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

```

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

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

```

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

```

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

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

```

{
  "Statement": [
    {
      "Sid": "AllowEveryoneReadWriteAccessIfInSourceIpRange",
      "Effect": "Allow",

```

```

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

```

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

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

```

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

```

### Example: PutOverwriteObject permission

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

```

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

```

```

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

```

### Related concepts

[Operations on buckets](#) on page 21

## S3 group policy examples

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

### Example: Setting the group policy using the Tenant Manager

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

- **No S3 Access:** Default option. Users in this group do not have access to S3 resources, unless access is granted with a bucket policy. If you select this option, only the root user will have access to S3 resources by default.
- **Read Only Access:** Users in this group have read-only access to S3 resources. For example, users in this group can list objects and read object data, metadata, and tags. When you select this option, the JSON string for a read-only group policy appears in the text box. You cannot edit this string.
- **Full Access:** Users in this group have full access to S3 resources, including buckets. When you select this option, the JSON string for a full-access group policy appears in the text box. You cannot edit this string.
- **Custom:** Users in the group are granted the permissions you specify in the text box. In this example, members of the group are only permitted to list and access their specific folder (key prefix) in the specified bucket.



#### Example: Allow group full access to all buckets

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

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

#### Example: Allow group read-only access to all buckets

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

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

```
]
}
```

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

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

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

#### Related concepts

[Using the PutOverwriteObject permission](#) on page 74

[Write-once-read-many \(WORM\) protection](#) on page 79

#### Related information

[Using tenant accounts](#)

## Configuring security for the REST API

---

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

### How StorageGRID provides security for the REST API

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

StorageGRID uses the following security measures.

- Client communications with the Load Balancer service use HTTPS if HTTPS is configured for the load balancer endpoint.  
When you configure a load balancer endpoint, HTTP can optionally be enabled. For example, you might want to use HTTP for testing or other non-production purposes. See the instructions for administering StorageGRID for more information.
- By default, StorageGRID uses HTTPS for client communications with Storage Nodes and the legacy CLB service on Gateway Nodes.  
HTTP can optionally be enabled for these connections. For example, you might want to use HTTP for testing or other non-production purposes. See the instructions for administering StorageGRID for more information.
- Communications between StorageGRID and the client are encrypted using TLS.
- Communications between the Load Balancer service and Storage Nodes within the grid are encrypted whether the load balancer endpoint is configured to accept HTTP or HTTPS connections.
- Clients must supply HTTP authentication headers to StorageGRID to perform REST API operations.

#### Security certificates and client applications

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

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

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

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

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

## Summary

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

Security issue	Implementation for REST API
Connection security	TLS
Server authentication	X.509 server certificate signed by system CA or custom server certificate supplied by administrator
Client authentication	<ul style="list-style-type: none"> <li>S3: S3 account (access key ID and secret access key)</li> <li>Swift: Swift account (user name and password)</li> </ul> <p><b>Note:</b> 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.</p>
Client authorization	<ul style="list-style-type: none"> <li>S3: Bucket ownership and all applicable access control policies</li> <li>Swift: Administrator role access</li> </ul>

## Related information

[Administering StorageGRID](#)

## Supported hashing and encryption algorithms for TLS libraries

The StorageGRID system supports a limited set of cipher suites that client applications can use when establishing a Transport Layer Security (TLS) session.

### Supported versions of TLS

StorageGRID supports TLS 1.2.

**Attention:** SSLv3 and TLS v1.1 (or earlier versions) are no longer supported.

### Supported cipher suites

IANA Name
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

### Deprecated cipher suites

The following cipher suites are deprecated. Support for these ciphers will be removed in a future release.

IANA Name
TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_256_GCM_SHA384



## Monitoring and auditing operations

---

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

### Monitoring object ingest and retrieval rates

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

#### Steps

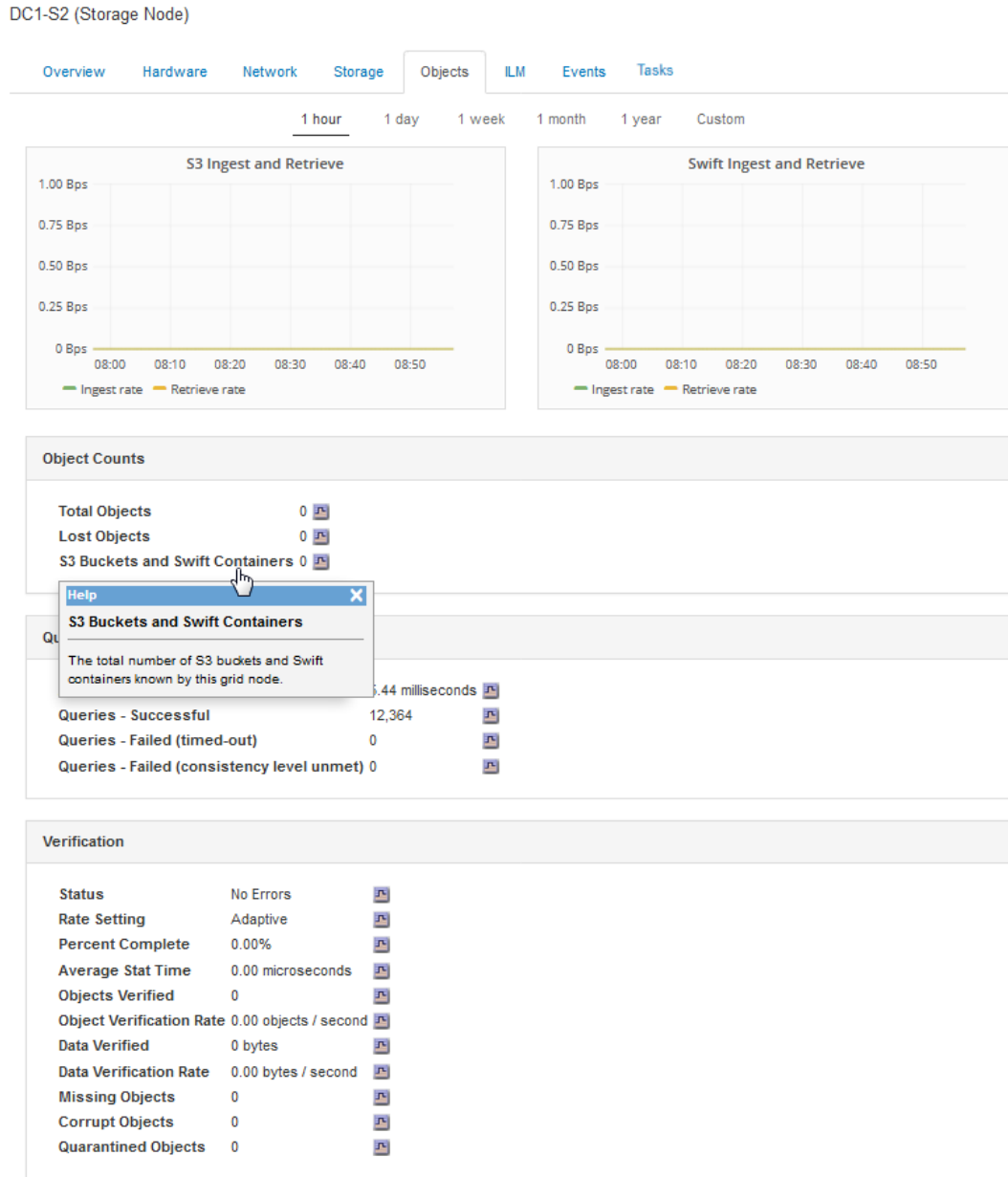
1. Sign in to the Grid Manager using a supported browser.
2. On the Dashboard, locate the Protocol Operations section.  

This section summarizes the number of client operations performed by your StorageGRID system. Protocol rates are averaged over the last two minutes.
3. Select **Nodes**.
4. From the Nodes home page (deployment level), click the **Load Balancer** tab.  

The charts show trends for all client traffic directed to load balancer endpoints within the grid. You can select a time interval in hours, days, weeks, months, or years, or you can apply a custom interval.
5. From the Nodes home page (deployment level), click the **Objects** tab.  

The chart shows ingest and retrieve rates for your entire StorageGRID system in bytes per second and total bytes. You can select a time interval in hours, days, weeks, months, or years, or you can apply a custom interval.
6. To see information for a particular Storage Node, select the node from the list on the left, and click the **Objects** tab.  

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



7. If you want even more detail:

- a. Select **Support > Grid Topology**.
- b. Select *site* > **Overview > Main**.

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

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

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

## Accessing and reviewing audit logs

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

### Before you begin

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

### About this task

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

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

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

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

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

### Steps

1. Log in to an Admin Node:
  - a. Enter the following command: `ssh admin@primary_Admin_Node_IP`
  - b. Enter the password listed in the `Passwords.txt` file.
2. Go to the directory containing the audit log files:
 

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

## S3 operations tracked in the audit logs

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

### Bucket operations tracked in the audit logs

- DELETE Bucket
- DELETE Multiple Objects
- GET Bucket (List Objects)
- GET Bucket Object versions
- HEAD Bucket

- PUT Bucket
- PUT Bucket compliance
- PUT Bucket Versioning

#### **Object operations tracked in the audit logs**

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

#### **Related concepts**

[Operations on buckets](#) on page 21

[Operations on objects](#) on page 27

## Benefits of active, idle, and concurrent HTTP connections

---

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

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

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

### Benefits of keeping idle HTTP connections open

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

Open and idle HTTP connections provide the following benefits:

- Reduced latency from the time that the StorageGRID system determines it has to perform an HTTP transaction to the time that the StorageGRID system can perform the transaction. Reduced latency is the main advantage, especially for the amount of time required to establish TCP/IP and TLS connections.
- Increased data transfer rate by priming the TCP/IP slow-start algorithm with previously performed transfers
- Instantaneous notification of several classes of fault conditions that interrupt connectivity between the client application and the StorageGRID system

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

### Benefits of active HTTP connections

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

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

For client connections to Storage Nodes or to the legacy CLB service, limiting active HTTP connections provides the following benefits:

- Enables optimal load balancing across the StorageGRID system.  
When using the CLB service, you should prevent long-lived TCP/IP connections to optimize load balancing across the StorageGRID system. 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 CLB service balances load across the StorageGRID system at the time that a client application establishes an HTTP connection. Over time, an HTTP connection might no longer be optimal as load balancing requirements change. The system performs its best load balancing when client applications establish a separate HTTP connection for each transaction, but this negates the much more valuable gains associated with persistent connections.

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

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

## Benefits of concurrent HTTP connections

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

Concurrent HTTP connections provide the following benefits:

- **Reduced latency**  
Transactions can start immediately instead of waiting for other transactions to be completed.
- **Increased throughput**  
The StorageGRID system can perform parallel transactions and increase aggregate transaction throughput.

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

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

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

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

## **Separation of HTTP connection pools for read and write operations**

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

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

## Copyright

---

Copyright © 2019 NetApp, Inc. All rights reserved. Printed in the U.S.

No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

Data contained herein pertains to a commercial item (as defined in FAR 2.101) and is proprietary to NetApp, Inc. The U.S. Government has a non-exclusive, non-transferrable, non-sublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b).



## Trademark information

---

Active IQ, AltaVault, Arch Design, ASUP, AutoSupport, Campaign Express, Clustered Data ONTAP, Customer Fitness, Data ONTAP, DataMotion, Element, Fitness, Flash Accel, Flash Cache, Flash Pool, FlexArray, FlexCache, FlexClone, FlexPod, FlexScale, FlexShare, FlexVol, FPolicy, Fueled by SolidFire, GetSuccessful, Helix Design, LockVault, Manage ONTAP, MetroCluster, MultiStore, NetApp, NetApp Insight, OnCommand, ONTAP, ONTAPI, RAID DP, RAID-TEC, SANscreen, SANshare, SANtricity, SecureShare, Simplicity, Simulate ONTAP, Snap Creator, SnapCenter, SnapCopy, SnapDrive, SnapIntegrator, SnapLock, SnapManager, SnapMirror, SnapMover, SnapProtect, SnapRestore, Snapshot, SnapValidator, SnapVault, SolidFire, SolidFire Helix, StorageGRID, SyncMirror, Tech OnTap, Unbound Cloud, and WAFL and other names are trademarks or registered trademarks of NetApp, Inc., in the United States, and/or other countries. All other brands or products are trademarks or registered trademarks of their respective holders and should be treated as such. A current list of NetApp trademarks is available on the web.

<http://www.netapp.com/us/legal/netapptmlist.aspx>

## 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\*](mailto:doccomments@netapp.com)

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

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

You can also contact us in the following ways:

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