Cloudian HyperStore AWS APIs Support Reference

Support for the S3 API

and the IAM, STS, and SQS APIs

Version 7.5.2

Contents

Chapter 1. S3 API

1.1. Introduction

1.1.1. HyperStore Support for the AWS S3 API

The Cloudian HyperStore system supports the great majority of the Amazon Web Services S3 REST API, including advanced features.

This documentation provides the details of the HyperStore system’s compliance with the S3 REST API. The organization of this documentation parallels that of the AWS S3 API Reference. Links are provided to specific parts of the AWS S3 API Reference so you can easily view additional information about individual API operations.

This documentation takes the approach of specifying in detail the things that the HyperStore system **does support** from the AWS S3 REST API — from operations down to the level of particular request parameters, request headers, request elements, response headers, and response elements. **If it’s not listed in this HyperStore S3 API Support documentation, the HyperStore system does not currently support it**.

This documentation also describes ways in which the HyperStore system extends the AWS S3 API, to support additional functionality. Most of these extensions are in the form of additional request headers that add enhanced functionality to standard AWS S3 operations on buckets. These extensions are described within the sections that document HyperStore compliance with standard AWS S3 operations. The extensions are always identified by a sub-heading that says **HyperStore Extension to the S3 API**. (For a summary of the extensions see **"HyperStore Extensions to the S3 API"** (page 19).)

To access the S3 Service, HyperStore users **need S3 access credentials**. When users are created in HyperStore, S3 access credentials are automatically created for the users.

If users are using a third party S3 client to access the HyperStore IAM Service, the users can obtain their S3 access credentials by logging in to the CMC and going to the **Security Credentials** page (via the drop-down menu under the user login name). They can then supply those credentials to the third party S3 client application.

If users are using the CMC's built-in S3 client, the CMC automatically uses the user's S3 credentials to access the S3 service. Through the CMC's **Bucket & Objects** section, HyperStore users can create buckets, upload and download objects, and so on.

1.1.1.1. S3 API Notes for HyperStore Administrators

S3 Service Endpoints

To find the S3 service endpoint(s) for your system go to the CMC's **Cluster Information** page (**Cluster -> Cluster Config -> Cluster Information**) -- or the **System Information** page if you have a HyperStore Single-Node system.

Using TLS/SSL

For information about setting up HTTPS for the S3 Service see the "HTTPS Feature Overview" section of the ***Cloudian HyperStore Administrator's Guide***. If HTTPS is enabled for your S3 Service it will listen for HTTPS connections on port 443, as well as listening for regular HTTP connections on port 80.

DNS

Be sure to configure the S3 endpoint domains in your DNS environment. For more information see the "DNS Set-Up" section of the ***Cloudian HyperStore Administrator's Guide***.

S3 Request Logging

Information about requests processed by the S3 Service are logged to *cloudian-request-info.log*, which exists on each node.For more information see the "Logging" section of the ***Cloudian HyperStore Administrator's Guide***.

Caution About Mass Deletes

S3 users should not attempt to delete more than 100,000 objects from a single bucket in less than an hour using the S3 API, if the bucket was created in a HyperStore version older than 7.4. Doing so will result in TombstoneOverwhelmingException errors in the Cassandra logs and an inability to successfully execute an **S3 ListObjectsVersion 1** or **ListObjectsVersion 2** operation on the bucket. If the system is in this error condition, you can trigger a tombstone purge as described in "Dealing with Excessive Tombstone Build-Up" in the "System cron jobs" section of the ***Cloudian HyperStore Administrator's Guide***.

Buckets created in HyperStore 7.4 and later use a different metadata structure than older buckets. Starting in HyperStore 7.4.2, all buckets created in HyperStore 7.4 and later use an improved tombstone monitoring and cleanup mechanism that leverages the new metadata structure. Such buckets are able to support mass deletions without negative service impact (they are not subject to the 100,000 deletes per hour limit that older buckets are). If you have buckets that were created prior to HyperStore 7.4 and that need to be able to support mass deletes, contact Cloudian Support for assistance in upgrading those buckets to the newer metadata structure (known as "rules based partitioning").

1.1.2. S3 Client Application Options

Broadly you have three options for using HyperStore's implementation of the AWS S3 API to create storage buckets in HyperStore, upload objects, retrieve objects, and so on:

- **"Using the CMC as Your S3 Client"** (page 14)
- **"Using Third Party S3 Applications"** (page 14)
- **"Developing Custom S3 Applications for HyperStore"** (page 15)

1.1.2.1. Using the CMC as Your S3 Client

The Cloudian Management Console's **Buckets & Objects** section serves as a graphical S3 client for interacting with the HyperStore object store. With the CMC, users can create and configure buckets, and upload and download objects.

1.1.2.2. Using Third Party S3 Applications

Because of HyperStore’s comprehensive compliance with the AWS S3 API, you can use most off-the-shelf third party S3 client applications with HyperStore. For feedback on particular S3 applications that you are considering using with HyperStore, consult with Cloudian Sales Engineering or Cloudian Support.

To check to see what is your HyperStore S3 Service endpoint -- the URI to which you will submit S3 requests with your third party application -- go to the CMC's **Security Credentials** page.

1.1.2.3. Developing Custom S3 Applications for HyperStore

In nearly every way, developing a client application for the Cloudian HyperStore storage service is the same as developing a client application for AWS S3. Consequently, when building S3 applications for the HyperStore service you can leverage the wealth of resources available to AWS S3 developers.

Good online resources for S3 application developers include:

- **Amazon Simple Storage Service Developer Guide**
- **Amazon S3 resources**

What’s Distinct About Developing for the HyperStore S3 Service

In practice, the main differences between developing for the HyperStore S3 service and developing for Amazon S3 are:

- HyperStore S3 client applications must use the HyperStore S3 service endpoint rather than the Amazon S3 service endpoint. To check to see what is your HyperStore S3 Service endpoint -- the URI to which you will submit S3 requests with your custom application -- go to the CMC's **Security Credentials** page.
- As detailed in the "Supported S3 Operations" section of this documentation, the HyperStore S3 service supports the great majority of but not the entire Amazon S3 API.
- Also as detailed in the "Supported S3 Operations" section of this documentation, the HyperStore S3 service supports a small number of extensions to the Amazon S3 API. (For an overview of the extensions see **"HyperStore Extensions to the S3 API"** (page 19)).

1.1.3. Authenticating Requests (AWS Signature Version 4)

HyperStore supports AWS Signature Version 4 for authenticating inbound API requests. The HyperStore implementation of this feature is compliant with Amazon’s specification of the feature. For example, you can express authentication information in the HTTP Authorization header or in query string parameters; and you can compute a checksum of the entire payload prior to transmission, or for large uploads, you can use chunked upload.

For more information on this Amazon S3 feature, refer to the **"Authenticating Requests (AWS Signature Version 4)" section of the Amazon S3 REST API**.

HyperStore continues to support AWS Signature Version 2 as well.

Note For HyperStore, the region name validation aspect of Signature Version 4 is disabled by default.System administrators can enable it.

1.1.4. Access Control List (ACL) Support

For the AWS S3 "Access Control List (ACL)" functionality, the HyperStore system supports the items listed below. If a grantee group, permission type, or canned ACL type from the AWS S3 documentation is not listed below, the HyperStore system does not support it.

For ACL usage information and for descriptions of ACL items, see **Access Control List (ACL) Overview** in the AWS S3 documentation.

1.1.4.1. AWS S3 Predefined Groups

- Authenticated users group
- All users group
- Log delivery group

1.1.4.2. Permission Types

- READ
- WRITE
- READ_ACP
- WRITE_ACP
- FULL_CONTROL

1.1.4.3. Canned ACL

- private
- public-read
- public-read-write
- authenticated-read
- bucket-owner-read
- bucket-owner-full-control
- log-delivery-write

HyperStore Extension to the S3 API

The HyperStore system supports these additional canned ACLs:

Canned ACL	Applies to	Permissions added to ACL
group-read	Bucket and object	Owner gets FULL_CONTROL. All other members of the owner's HyperStore service user group get READ access.
group-read-write	Bucket and object	Owner gets FULL_CONTROL. All other members of the owner's HyperStore service user group get READ and WRITE access.

Note To grant access to groups other than the requester's own group, you cannot use canned ACLs. Instead, when using standard Amazon S3 methods for assigning privileges to a grantee (via request headers or request body), specify "<groupID>|" as the grantee. The "<groupID>|" format (with vertical bar) indicates that the grantee is a group — for example, "Group5|".

Note When access privileges have through separate requests been granted to a group and to a specific member of the group, the user gets the broader of the privilege grants. For example, if Group5 is granted read-write privileges and a specific user within Group5 is separately granted read privileges, the user gets read-write privileges.

1.1.5. S3 Common Request and Response Headers

1.1.5.1. Common Request Headers

From the **"Common Request Headers" section** of the AWS S3 REST API specification, HyperStore supports the headers listed below. If a header from that specification section is not listed below, HyperStore does not support it.

- Authorization
- Content-Length
- Content-Type
- Content-MD5
- Date
- Expect
- Host

> **Note** For path-style requests, the Host value is your system's S3 endpoint (excluding the protocol and port) -- for example *s3-region1.enterprise.com*. For virtual-hosted-style requests, the Host value is *BucketName.<S3 endpoint>* -- for example *bucket1.s3-region1.enterprise.com*.

- x-amz-content-sha256
- x-amz-date
- x-amz-expected-bucket-owner

> **Note**
> * Unlike the AWS documentation which lists *x-amz-expected-bucket-owner* as a supported request header for nearly every individual S3 API call but omits the header from the Common header list, this HyperStore documentation instead lists the *x-amz-expected-bucket-owner* header here among the Common headers. For the HyperStore S3 Service, the *x-amz-expected-bucket-owner* request header is supported for all S3 API calls except *CreateBucket* and *ListBuckets*.
> * If you use the optional *x-amz-expected-bucket-owner* request header in making S3 calls to the HyperStore S3 Service, identify the expected bucket owner by the **bucket owner's canonical user ID**. A user's canonical user ID can be obtained by retrieving the user's profile in the CMC or via the Admin API call *GET /user*.
> * As with AWS, if the destination bucket in an API request is owned by an account other than the expected bucket owner account, the request will fail with an HTTP 403 (Access Denied) error.

1.1.5.2. Common Response Headers

From the **"Common Response Headers" section** of the AWS S3 REST API specification, HyperStore supports the headers listed below. If a header from that specification section is not listed below, HyperStore does not support it.

- Content-Length
- Content-Type

- Connection
- Date
- ETag
- Server
- x-amz-delete-marker
- x-amz-request-id
- x-amz-version-id

1.1.6. S3 Error Responses

From the **"Error Responses" section** of the AWS S3 API specification, HyperStore supports the error codes listed below, in the same format as indicated in the specification. If an error code from that specification section is not listed below, HyperStore does not support it.

- AccessDenied
- AccountProblem
- AmbiguousGrantByEmailAddress
- BadDigest
- BucketAlreadyExists
- BucketAlreadyOwnedByYou
- BucketNotEmpty
- CrossLocationLoggingProhibited
- EntityTooLarge
- EntityTooSmall
- IllegalVersioningConfigurationException
- IncorrectNumberOfFilesInPostRequest
- InternalError
- InvalidAccessKeyId
- InvalidArgument
- InvalidBucketName
- InvalidBucketState
- InvalidDigest
- InvalidEncryptionAlgorithmError
- InvalidLocationConstraint
- InvalidObjectState
- InvalidPart
- InvalidPartOrder
- InvalidPolicyDocument
- InvalidRange
- InvalidRequest
- InvalidSecurity

- InvalidTargetBucketForLogging
- InvalidURI
- KeyTooLong
- MalformedACLError
- MalformedPOSTRequest
- MalformedXML
- MaxMessageLengthExceeded
- MaxPostPreDataLengthExceededError
- MetadataTooLarge
- MethodNotAllowed
- MissingContentLength
- MissingSecurityHeader
- NoSuchBucket
- NoSuchBucketPolicy
- NoSuchKey
- NoSuchLifecycleConfiguration
- NoSuchReplicationConfiguration
- NoSuchUpload
- NoSuchVersion
- NotImplemented
- PermanentRedirect
- PreconditionFailed
- Redirect
- RestoreAlreadyInProgress
- RequestIsNotMultiPartContent
- RequestTimeout
- RequestTimeTooSkewed
- SignatureDoesNotMatch
- ServiceUnavailable
- SlowDown
- TemporaryRedirect
- TooManyBuckets
- UnexpectedContent
- UnresolvableGrantByEmailAddress
- UserKeyMustBeSpecified

1.1.7. HyperStore Extensions to the S3 API

The HyperStore S3 Service supports the following extensions to the AWS S3 REST API. In each case the extensions take the form of additional supported headers for standard AWS S3 API methods.

Extension	Purpose	Detail
x-gmt-policyid as optional request header for "CreateBucket" and response header for "ListObjects", "ListObjectsV2", and "HeadBucket"	Specify the HyperStore storage policy to use for a new bucket	• **"CreateBucket"** (page 22)
x-gmt-tieringinfo and *x-gmt-compare* and *x-gmt-post-tier-copy* as optional request headers for "PutBucketLifecycle" and response headers for "GetBucketLifecycle"	Set up auto-tiering for a bucket	• **"PutBucketLifecycleConfiguration"** (page 57)
x-gmt-error-code and *x-gmt-message* as supported response headers for "GetObject" and "HeadObject"	Provide additional information about HTTP 4xx errors	• **"GetObject"** (page 37) • **"HeadObject"** (page 42)
x-gmt-crr-endpoint and *x-gmt-crr-credentials* as optional request headers for "PutBucketReplication" and response headers for "GetBucketReplication".	Use for cross-**system** replication	• **"PutBucketReplication"** (page 70) • Cross-System Replication

1.2. Supported S3 Operations

1.2.1. AbortMultipartUpload

This operation aborts a multipart upload.

Along with the **common headers**, HyperStore supports the operation-specific parameters listed below.

For operation details and examples see the AWS documentation: **AbortMultipartUpload**

Former operation name: Abort Multipart Upload

1.2.1.1. Query Parameters

- uploadId

1.2.2. CompleteMultipartUpload

Completes a multipart upload by assembling previously uploaded parts.

Along with the **common headers**, HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: **CompleteMultipartUpload**

Former operation name: Complete Multipart Upload

1.2.2.1. Request Elements

- CompleteMultipartUpload
    - Part
        - ETag
        - PartNumber

1.2.2.2. Response Headers

- x-amz-expiration
- x-amz-server-side-encryption
- x-amz-version-id

1.2.2.3. Response Elements

- Bucket
- CompleteMultipartUploadResult
- ETag
- Key
- Location

1.2.3. CopyObject

Creates a copy of an object that is already stored in HyperStore.

Along with the **common headers**, HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: **CopyObject**

Former operation name: PUT Object - Copy

1.2.3.1. Request Headers

- x-amz-acl
- x-amz-copy-source
- x-amz-copy-source-if-match
- x-amz-copy-source-if-modified-since
- x-amz-copy-source-if-none-match
- x-amz-copy-source-if-unmodified-since
- x-amz-copy-source-server-side-encryption-customer-algorithm
- x-amz-copy-source-server-side-encryption-customer-key
- x-amz-copy-source-server-side-encryption-customer-key-MD5
- x-amz-grant-full-control
- x-amz-grant-read

- x-amz-grant-read-acp
- x-amz-grant-write
- x-amz-grant-write-acp
- x-amz-meta-*
- x-amz-metadata-directive
- x-amz-object-lock-legal-hold
- x-amz-object-lock-mode
- x-amz-object-lock-retain-until-date
- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key
- x-amz-server-side-encryption-customer-key-MD5
- x-amz-storage-class

  Note HyperStore ignores the value of the *x-amz-storage-class* header and treats all requests as being for storage class STANDARD.

- x-amz-source-expected-bucket-owner
- x-amz-tagging
- x-amz-tagging-directive
- x-amz-website-redirect-location

1.2.3.2. Response Headers

- x-amz-copy-source-version-id
- x-amz-expiration
- x-amz-server-side-encryption
- x-amz-version-id

1.2.3.3. Response Elements

- CopyObjectResult
  - ETag
  - LastModified

1.2.4. CreateBucket

Creates a new bucket.

Along with the **common headers**, HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: **CreateBucket**

Former operation name: PUT Bucket

Note
* By default each user is allowed a maximum of 100 buckets.
* HyperStore enforces the same **bucket naming restrictions** as does Amazon S3. Also, **if you use an underscore in a bucket name you will not be able to enable auto-tiering** for the bucket (for transitioning objects to Amazon or other remote destinations on a configurable schedule). It's best not to use underscores when naming new buckets, in case you may want to enable auto-tiering on the bucket immediately or in the future.

1.2.4.1. Request Headers

- x-amz-acl
- x-amz-grant-full-control
- x-amz-grant-read
- x-amz-grant-read-acp
- x-amz-grant-write
- x-amz-grant-write-acp
- x-amz-object-lock-enabled
- x-amz-object-ownership

HyperStore Extension to the S3 API

The HyperStore system supports the following Request Header as an extension to the "PUT Bucket" operation:

Name	Description	Required
x-gmt-policyid	This header specifies the unique ID of the storage policy to assign to the newly created bucket. The storage policy determines how data in the bucket will be distributed and protected through either replication or erasure coding. System administrators can create multiple storage policies through the CMC and the system automatically assigns each a unique policy ID that becomes part of the policy definition. With the "x-gmt-policyid" request header for "PUT Bucket", you specify the ID of the desired storage policy when you create a new bucket. Note however that some policies may not be available to all user groups — a policy’s availability is specified by system administrators at the time of policy creation, and this information becomes part of the policy definition. When you specify an "x-gmt-policyid" value with a "PUT Bucket" request, the policy ID must be for a policy that is available to the group to which the bucket owner belongs. Also the policy ID must be for a storage policy from the service region that is specified in the "PUT Bucket" request’s LocationConstraint element. If the "PUT Bucket" request does not include the "x-gmt-policyid" request header, then the system will automatically assign the system default storage policy to the bucket during bucket creation. **Note** After a bucket is created, it cannot be assigned a different stor-	No

Name	Description	Required
	age policy. The storage policy assigned to the bucket at bucket creation time will continue to be bucket's storage policy for the life of the bucket. **Note** A 403 error response is returned if you specify a policy ID that does not exist, has been disabled, is not available to the region in which the bucket is being created, or is not available to the group to which the bucket owner belongs. A 403 is also returned if you do not specify an "x-gmt-policyid" header and the system does not yet have an established default storage policy. Example header: `x-gmt-policyid: 1bc90238f9f11cb32f5e4e901675d50b`	

1.2.4.2. Request Elements

- CreateBucketConfiguration
    - LocationConstraint

1.2.5. CreateMultipartUpload

This operation initiates a multipart upload and returns an upload ID.

Along with the **common headers**, HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: **CreateMultipartUpload**

Former operation name: Initiate Multipart Upload

1.2.5.1. Request Headers

- Cache-Control
- Content-Disposition
- Content-Encoding
- Content-Type
- Expires
- x-amz-acl
- x-amz-grant-full-control
- x-amz-grant-read
- x-amz-grant-read-acp
- x-amz-grant-write
- x-amz-grant-write-acp

- x-amz-meta-*
- x-amz-object-lock-legal-hold
- x-amz-object-lock-mode
- x-amz-object-lock-retain-until-date
- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key
- x-amz-server-side-encryption-customer-key-MD5
- x-amz-storage-class

  Note HyperStore ignores the value of the *x-amz-storage-class* header and treats all requests as being for storage class STANDARD.

- x-amz-website-redirect-location

1.2.5.2. Response Headers

- x-amz-abort-date
- x-amz-abort-rule-id
- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key-MD5

1.2.5.3. Response Elements

- InitiateMultipartUploadResult
  - Bucket
  - Key
  - UploadId

1.2.6. DeleteBucket

Deletes the bucket.

For this operation HyperStore supports the **S3 common headers**.

For operation details and examples see the AWS documentation: **DeleteBucket**

Former operation name: DELETE Bucket

1.2.7. DeleteBucketCors

Deletes the *cors* configuration information set for the bucket.

For this operation HyperStore supports the **S3 common headers**.

For operation details and examples see the AWS documentation: **DeleteBucketCors**

Former operation name: DELETE Bucket cors

1.2.8. DeleteBucketEncryption

This implementation of the DELETE operation removes default encryption from the bucket.

For this operation HyperStore supports the **S3 common headers**.

For operation details and examples see the AWS documentation: **DeleteBucketEncryption**

Former operation name: DELETE Bucket encryption

1.2.9. DeleteBucketInventoryConfiguration

Deletes an inventory configuration (identified by the inventory ID) from the bucket.

Along with the **common headers**, HyperStore supports the operation-specific parameter listed below.

For operation details and examples see the AWS documentation: **DeleteBucketInventoryConfiguration**

1.2.9.1. Query Parameter

- id

1.2.10. DeleteBucketLifecycle

Deletes the lifecycle configuration from the specified bucket.

For this operation HyperStore supports the **S3 common headers**.

For operation details and examples see the AWS documentation: **DeleteBucketLifecycle**

Former operation name: DELETE Bucket lifecycle

1.2.11. DeleteBucketOwnershipControls

Removes *OwnershipControls* for a bucket.

For this operation HyperStore supports the **S3 common headers**.

For operation details and examples see the AWS documentation: **DeleteBucketOwnershipControls**

1.2.12. DeleteBucketPolicy

This implementation of the DELETE operation uses the policy subresource to delete the policy of a specified bucket.

For this operation HyperStore supports the **S3 common headers**.

For operation details and examples see the AWS documentation: **DeleteBucketPolicy**

Former operation name: DELETE Bucket policy

1.2.13. DeleteBucketReplication

Deletes the replication configuration from the bucket.

For this operation HyperStore supports the **S3 common headers**.

For operation details and examples see the AWS documentation: **DeleteBucketReplication**

Former operation name: DELETE Bucket replication

1.2.14. DeleteBucketTagging

Deletes the tags from the bucket.

For this operation HyperStore supports the **S3 common headers**.

For operation details and examples see the AWS documentation: **DeleteBucketTagging**

Former operation name: DELETE Bucket tagging

1.2.15. DeleteBucketWebsite

This operation removes the website configuration for a bucket.

For this operation HyperStore supports the **S3 common headers**.

For operation details and examples see the AWS documentation: **DeleteBucketWebsite**

Former operation name: DELETE Bucket website

1.2.16. DeleteObject

Removes the null version (if there is one) of an object and inserts a delete marker, which becomes the latest version of the object.

Along with the **common headers**, HyperStore supports the operation-specific headers listed below.

For operation details and examples see the AWS documentation: **DeleteObject**

Former operation name: DELETE Object

Note Successful completion of a *DeleteObject* request results in the system marking the object as having been deleted. However the actual deletion of object data from disk will not occur until the next automatic running of the object deletion batch processing job. By default this batch processing of object data deletes runs hourly on each node.

IMPORTANT ! Do not attempt to delete more than 100,000 objects from a single bucket in less than an hour, if the bucket was created in a HyperStore version older than 7.4.

Buckets created in HyperStore 7.4 and later use a different metadata structure than older buckets. Starting in HyperStore 7.4.2, all buckets created in HyperStore 7.4 and later use an improved tombstone monitoring and cleanup mechanism that leverages the new metadata structure. Such buckets are able to support mass deletions without negative service impact (they are not subject to the 100,000 deletes per hour limit that older buckets are). If you have buckets that were created prior to HyperStore 7.4 and that need to be able to support mass deletes, contact Cloudian Support for assistance in upgrading those buckets to the newer metadata structure (known as "rules based partitioning").

1.2.16.1. Request Headers

- x-amz-bypass-governance-retention

1.2.16.2. Response Headers

- x-amz-delete-marker
- x-amz-version-id

1.2.17. DeleteObjects

This operation enables you to delete multiple objects from a bucket using a single HTTP request.

Along with the **common headers**, HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: **DeleteObjects**

Former operation name: Delete Multiple Objects

Note The HyperStore S3 Service allows a maximum of 1000 object deletes per *DeleteObjects* request.

Note Successful completion of a *DeleteObjects* request results in the system marking the objects as having been deleted. However the actual deletion of object data from disk will not occur until the next automatic running of the object deletion batch processing job. By default this batch processing of object data deletes runs hourly on each node.

IMPORTANT ! Do not attempt to delete more than 100,000 objects from a single bucket in less than an hour, if the bucket was created in a HyperStore version older than 7.4.

Buckets created in HyperStore 7.4 and later use a different metadata structure than older buckets. Starting in HyperStore 7.4.2, all buckets created in HyperStore 7.4 and later use an improved tombstone monitoring and cleanup mechanism that leverages the new metadata structure. Such buckets are able to support mass deletions without negative service impact (they are not subject to the 100,000 deletes per hour limit that older buckets are). If you have buckets that were created prior to HyperStore 7.4 and that need to be able to support mass deletes, contact Cloudian Support for assistance in upgrading those buckets to the newer metadata structure (known as "rules based partitioning").

1.2.17.1. Request Headers

- x-amz-bypass-governance-retention

1.2.17.2. Request Elements

- Delete
    - Object
        - Key
        - VersionId
    - Quiet

1.2.17.3. Response Elements

- DeleteResult
    - Deleted
        - DeleteMarker
        - DeleteMarkerVersionId
        - Key
        - VersionId
    - Error
        - Code
        - Key
        - Message
        - VersionId

1.2.18. DeleteObjectTagging

Removes the entire tag set from the specified object.

For this operation HyperStore supports the **S3 common headers**.

For operation details and examples see the AWS documentation: **DeleteObjectTagging**

Former operation name: DELETE Object tagging

1.2.19. DeletePublicAccessBlock

Removes the *PublicAccessBlock* configuration for a bucket.

For this operation HyperStore supports the **S3 common headers**.

For operation details and examples see the AWS documentation: **DeletePublicAccessBlock**

1.2.20. GetBucketAcl

This implementation of the GET operation uses the *acl* subresource to return the access control list (ACL) of a bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **GetBucketAcl**

Former operation name: GET Bucket acl

1.2.20.1. Response Elements

- AccessControlPolicy
  - Owner
    - DisplayName
    - ID
  - AccessControlList
    - Grant
      - Grantee
        - DisplayName
        - ID
        - Permission

1.2.21. GetBucketCors

Returns the *cors* configuration information set for the bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **GetBucketCors**

Former operation name: GET Bucket cors

1.2.21.1. Response Elements

- CORSConfiguration
  - CORSRule
    - AllowedHeader
    - AllowedMethod
    - AllowedOrigin
    - ExposeHeader
    - ID
    - MaxAgeSeconds

1.2.22. GetBucketEncryption

Returns the default encryption configuration for the bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **GetBucketEncryption**

Former operation name: GET Bucket encryption

1.2.22.1. Response Elements

- ServerSideEncryptionConfiguration
    - Rule
        - ApplyServerSideEncryptionByDefault
            - SSEAlgorithm

1.2.23. GetBucketInventoryConfiguration

Returns an inventory configuration (identified by the inventory configuration ID) from the bucket.

Along with the **common headers**, HyperStore supports the operation-specific parameter and elements listed below.

For operation details and examples see the AWS documentation: **GetBucketInventoryConfiguration**

1.2.23.1. Query Parameter

- id

1.2.23.2. Response Elements

- InventoryConfiguration
    - Destination
        - S3BucketDestination
            - AccountId
            - Bucket
            - Encryption
                - SSE-KMS
                    - KeyId
                - SSE-S3
            - Format
            - Prefix
    - Filter
        - Prefix
    - Id
    - IncludedObjectVersions
    - IsEnabled
    - OptionalFields
        - Field
    - Schedule
        - Frequency

1.2.24. GetBucketLifecycle

Returns the lifecycle configuration information set on the bucket.

For operation details and examples see the AWS documentation: **GetBucketLifecycle**

Note Though HyperStore supports this API operation for backward compatibility, AWS has deprecated this operation in favor of a newer version called **GetBucketLifecycleConfiguration** which HyperStore also supports. If you used **PutBucketLifecycleConfiguration** to create a lifecycle use **GetBuck-etLifecycleConfiguration** to retrieve the configuration.

1.2.25. GetBucketLifecycleConfiguration

Returns the lifecycle configuration information set on the bucket.

Along with the **common headers**, HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: **GetBucketLifecycleConfiguration**

Former operation name: GET Bucket lifecycle

1.2.25.1. Response Headers

HyperStore Extension to the S3 API

The HyperStore system supports the following Response Headers as extensions to the "GetBuck-etLifecycleConfiguration" operation:

Name	Description	Required
x-gmt-tieringinfo	See **"PutBucketLifecycleConfiguration"** (page 57).	No
x-gmt-compare		
x-gmt-post-tier-copy		

1.2.25.2. Response Elements

- LifecycleConfiguration
  - Rule
    - AbortIncompleteMultipartUpload
      - DaysAfterInitiation
    - Expiration
      - Date
      - Days
      - ExpiredObjectDeleteMarker

- Filter
  - And
    - ObjectSizeGreaterThan
    - ObjectSizeLessThan
    - Prefix
    - Tag
      - Key
      - Value
  - Prefix
  - Tag
    - Key
    - Value
- ID
- NoncurrentVersionExpiration
  - NoncurrentDays
- NoncurrentVersionTransition
  - NoncurrentDays
  - StorageClass
- Prefix
- Status
- Transition
  - Date
  - Days
  - StorageClass

1.2.26. GetBucketLocation

Returns the Region the bucket resides in.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **GetBucketLocation**

Former operation name: GET Bucket location

1.2.26.1. Response Elements

- LocationConstraint

1.2.26.2. GetBucketLocation Response for Buckets in the Default Service Region

The GetBucketLocation operation behaves as follows:

- If the bucket specified in the GetBucketLocation request resides in a non-default service region, the response indicates the name of the service region.

- If the bucket specified in the GetBucketLocation request resides in the default service region, the response returns a null/empty value.

HyperStore's behavior of returning a null/empty value if the bucket is in the default region is the same as Amazon Web Services' implementation of the GetBucketLocation operation. Some S3 client applications -- such as Veeam -- are unable to handle the return of a null/empty region value, and may display an error if the actual default region name is set within the client application. The work-around is to not set the region in the client application, or else set it to the AWS default region name: *us-east-1*.

1.2.27. GetBucketLogging

Returns the logging status of a bucket and the permissions users have to view and modify that status.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **GetBucketLogging**

Former operation name: GET Bucket logging

1.2.27.1. Response Elements

- BucketLoggingStatus
    - LoggingEnabled
        - TargetBucket
        - TargetGrants
            - Grant
                - Grantee
                - Permission
        - TargetPrefix

1.2.28. GetBucketNotificationConfiguration

Returns the notification configuration of a bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **GetBucketNotificationConfiguration**

1.2.28.1. Response Elements

- NotificationConfiguration
    - QueueConfiguration
    - Event
    - Filter
        - S3Key
            - FilterRule
                - Name
                - Value

    - Id
    - Queue

For Event types, HyperStore supports only the following:

- s3:ObjectCreated:*
- s3:ObjectCreated:Put
- s3:ObjectCreated:Post
- s3:ObjectCreated:Copy
- s3:ObjectCreated:CompleteMultipartUpload
- s3:ObjectRemoved:*
- s3:ObjectRemoved:Delete
- s3:ObjectRemoved:DeleteMarkerCreated

1.2.29. GetBucketOwnershipControls

Retrieves *OwnershipControls* for a bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **GetBucketOwnershipControls**

1.2.29.1. Response Elements

- OwnershipControls
    - Rule
        - ObjectOwnership

1.2.30. GetBucketPolicy

Returns the policy of a specified bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **GetBucketPolicy**

Former operation name: GET Bucket policy

1.2.30.1. Response Elements

The response contains the (JSON) policy of the specified bucket.

1.2.31. GetBucketPolicyStatus

Retrieves the policy status for a bucket, indicating whether the bucket is public.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **GetBucketPolicyStatus**

1.2.31.1. Response Elements

- PolicyStatus
    - IsPublic

Note HyperStore considers a bucket policy to be "public" if any statement in the policy is public. A statement is considered public if the Effect is Allow and the Principal has a wildcard -- unless there is an *IpAddress:{aws:SourceIp* condition associated with the statement that restricts the requesting source IP to one or more specified IP addresses.

1.2.32. GetBucketReplication

Returns the replication configuration of a bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **GetBucketReplication**

Former operation name: GET Bucket replication

1.2.32.1. Response Elements

- ReplicationConfiguration
    - Role
    - Rule

1.2.33. GetBucketTagging

Returns the tag set associated with the bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **GetBucketTagging**

Former operation name: GET Bucket tagging

1.2.33.1. Response Elements

- Tagging
    - TagSet
        - Tag
            - Key
            - Value

1.2.34. GetBucketVersioning

Returns the versioning state of a bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **GetBucketVersioning**

Former operation name: GET Bucket versioning

1.2.34.1. Response Elements

- VersioningConfiguration
    - Status

1.2.35. GetBucketWebsite

Returns the website configuration for a bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **GetBucketWebsite**

Former operation name: GET Bucket website

1.2.35.1. Response Elements

- WebsiteConfiguration
-
    - ErrorDocument
        - Key
    - IndexDocument
        - Suffix
    - RedirectAllRequestsTo
        - HostName
        - Protocol

1.2.36. GetObject

Retrieves objects from the S3 storage system.

Along with the **common headers**, HyperStore supports the operation-specific parameters, headers, and elements listed below.

For operation details and examples see the AWS documentation: **GetObject**

Former operation name: GET Object

1.2.36.1. Query Parameters

- partNumber

> **Note** Using the *partNumber* parameter may not work as expected if the object has been auto-tiered, or if the object has been auto-tiered and restored. This is because an object's number of parts when uploaded to HyperStore may be different than its number of parts when it is auto-tiered to a remote destination system.

- response-cache-control
- response-content-disposition
- response-content-encoding
- response-content-language
- response-content-type
- response-expires
- versionId

1.2.36.2. Request Headers

- If-Match
- If-Modified-Since
- If-None-Match
- If-Unmodified-Since
- Range
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key
- x-amz-server-side-encryption-customer-key-MD5

1.2.36.3. Response Headers

- Last-Modified
- x-amz-delete-marker
- x-amz-expiration
- x-amz-meta-*
- x-amz-object-lock-legal-hold
- x-amz-object-lock-mode
- x-amz-object-lock-retain-until-date
- x-amz-replication-status
- x-amz-restore
- x-amz-server-side-encryption

  Note For objects encrypted with the KMIP type of server-side encryption a "-KMIP" suffix will be appended to the *x-amz-server-side-encryption* response header value.

- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key-MD5
- x-amz-tagging-count
- x-amz-version-id
- x-amz-website-redirect-location

HyperStore Extension to the S3 API

The HyperStore system supports the following Response Headers as extensions to the "GET Object" operation. These headers are returned **only in the event of an HTTP 4xx response**. They are not returned with HTTP 2xx, 3xx, or 5xx responses.

Name	Description
x-gmt-error-code	In the event of an HTTP 4xx response, these two response headers provide additional information about the nature of the error. The *x-gmt-error-code* header values will be from among the list in **"S3 Error Responses"** (page 18).
x-gmt-message	

1.2.37. GetObjectAcl

Returns the access control list (ACL) of an object.

Along with the **common headers**, HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: **GetObjectAcl**

Former operation name: GET Object acl

1.2.37.1. Response Elements

- AccessControlPolicy
    - Owner
        - DisplayName
        - ID
    - AccessControlList
        - Grant
            - Grantee
                - DisplayName
                - ID
            - Permission

1.2.38. GetObjectLegalHold

Gets an object's current Legal Hold status.

Along with the **common headers**, HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: **GetObjectLegalHold**

Former operation name: GET Object legal hold

1.2.38.1. Query Parameters

- versionId

1.2.38.2. Response Elements

- LegalHold
  - Status

1.2.39. GetObjectLockConfiguration

Gets the Object Lock configuration for a bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **GetObjectLockConfiguration**

Former operation name: GET Bucket object lock configuration

1.2.39.1. Response Elements

- ObjectLockConfiguration
  - ObjectLockEnabled
  - Rule
    - DefaultRetention
      - Days
      - Mode
      - Years

1.2.40. GetObjectRetention

Retrieves an object's retention settings.

Along with the **common headers**, HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: **GetObjectRetention**

Former operation name: GET Object retention

1.2.40.1. Query Parameters

- versionId

1.2.40.2. Response Elements

- Retention
  - Mode
  - RetainUntilDate

1.2.41. GetObjectTagging

Returns the tag-set of an object.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **GetObjectTagging**

Former operation name: GET Object tagging

1.2.41.1. Response Elements

- Tagging
  - TagSet
    - Tag
      - Key
      - Value

1.2.42. GetObjectTorrent

Return torrent files from a bucket.

For operation details and examples see the AWS documentation: **GetObjectTorrent**

Former operation name: GET Object torrent

1.2.42.1. Implementation Notes

- HyperStore does not provide a BitTorrent "tracker". You must either provide your own tracker or use one of the many publicly available trackers. System administrators must configure the system with the URL of the tracker that you are using.
- HyperStore implements BitTorrent HTTP seeding for in accordance with the BEP19 specification (**http://www.bittorrent.org/beps/bep_0019.html**). Therefore torrent files returned by HyperStore in response to *GetObjectTorrent* requests will include a "url-list" key and the value of that key will be the URL of the object in HyperStore.
- HyperStore objects that have been auto-tiered to a destination S3 system cannot be retrieved via BitTorrent, unless the objects are first restored to local HyperStore storage (via the S3 **"RestoreObject"** (page 78) method). Restored objects can be retrieved from HyperStore via BitTorrent.
- Like with Amazon S3, with HyperStore only publicly readable objects are eligible for BitTorrent retrieval. And like with Amazon S3, the following types of objects are **not** retrievable via BitTorrent:
  - Objects larger than 5GB
  - Non-current versions of versioned objects
  - Objects encrypted via SSE-C (SSE with Customer-managed key; by contrast, BitTorrent retrieval is supported for objects encrypted with regular SSE)

1.2.43. GetPublicAccessBlock

Retrieves the *PublicAccessBlock* configuration for a bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **GetPublicAccessBlock**

1.2.43.1. Response Elements

- PublicAccessBlockConfiguration
    - BlockPublicAcls
    - IgnorePublicAcls
    - BlockPublicPolicy
    - RestrictPublicBuckets

1.2.44. HeadBucket

This operation is useful to determine if a bucket exists and you have permission to access it.

Along with the **common headers**, HyperStore supports the operation-specific headers listed below.

For operation details and examples see the AWS documentation: **HeadBucket**

Former operation name: HEAD Bucket

1.2.44.1. Response Headers

- x-amz-bucket-region

HyperStore Extension to the S3 API

The HyperStore system supports the following Response Header as an extension to the "HeadBucket" operation:

Parameter	Description
x-gmt-policyid	This header specifies the unique ID of the storage policy assigned to the bucket. For more information see **"CreateBucket"** (page 22).

1.2.45. HeadObject

The HEAD operation retrieves metadata from an object without returning the object itself.

Along with the **common headers**, HyperStore supports the operation-specific parameters and headers listed below.

For operation details and examples see the AWS documentation: **HeadObject**

Former operation name: HEAD Object

1.2.45.1. Query Parameters

- partNumber

Note Using the *partNumber* parameter may not work as expected if the object has been auto-tiered, or if the object has been auto-tiered and restored. This is because an object's number of parts when uploaded to HyperStore may be different than its number of parts when it is auto-tiered to a remote destination system.

- versionId

1.2.45.2. Request Headers

- If-Match
- If-Modified-Since
- If-None-Match
- If-Unmodified-Since
- Range
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key
- x-amz-server-side-encryption-customer-key-MD5

1.2.45.3. Response Headers

- Last-Modified
- x-amz-expiration
- x-amz-meta-*
- x-amz-object-lock-legal-hold
- x-amz-object-lock-mode
- x-amz-object-lock-retain-until-date
- x-amz-replication-status
- x-amz-restore
- x-amz-server-side-encryption

  Note For objects encrypted with the KMIP type of server-side encryption a "-KMIP" suffix will be appended to the *x-amz-server-side-encryption* response header value.

- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key-MD5
- x-amz-tagging-count
- x-amz-version-id

HyperStore Extension to the S3 API

The HyperStore system supports the following Response Headers as extensions to the "HeadObject" operation. These headers are returned **only in the event of an HTTP 4xx response**. They are not returned with HTTP 2xx, 3xx, or 5xx responses.

Name	Description
x-gmt-error-code	In the event of an HTTP 4xx response, these two response headers provide additional information about the nature of the error. The *x-gmt-error-code* header values will be from among the list in **"S3 Error Responses"** (page 18).
x-gmt-message	

1.2.46. ListBucketInventoryConfigurations

Returns a list of inventory configurations for the bucket.

Along with the **common headers**, HyperStore supports the operation-specific parameter and elements listed below.

For operation details and examples see the AWS documentation: **ListBucketInventoryConfigurations**

1.2.46.1. Query Parameter

- continuation-token

1.2.46.2. Response Elements

- ListInventoryConfigurationsResult
    - ContinuationToken
    - InventoryConfiguration
        - Destination
            - S3BucketDestination
                - AccountId
                - Bucket
                - Encryption
                    - SSE-KMS
                        - KeyId
                    - SSE-S3
                - Format
                - Prefix
        - Filter
            - Prefix
        - Id
        - IncludedObjectVersions
        - IsEnabled
        - OptionalFields
            - Field
        - Schedule
            - Frequency

- IsTruncated
- NextContinuationToken

1.2.47. ListBuckets

Returns a list of all buckets owned by the authenticated sender of the request.

Along with the **common headers**, HyperStore supports the operation-specific parameter listed below.

For operation details and examples see the AWS documentation: **ListBuckets**

Former operation name: GET Service

HyperStore Extension to the S3 API

The HyperStore system supports the following Query Parameter as an extension to the "ListBuckets" operation:

Note . This extension is supported only if it has been enabled by system administrators.

Name	Description	Required

shared	If the *shared* parameter is included in the request, the ListBuckets operation returns a list of buckets that other users have shared with the requesting user. This will be buckets that have been shared specifically with the requesting user, plus buckets that have been shared with the group to which the requesting user belongs, plus buckets that have been shared with everyone.
	
	Example:
	
	`GET /?shared HTTP/1.1.`
	`Host: s3-region1.enterprise4.mobi-cloud.com.`
	`Accept-Encoding: identity.`
	`Date: Fri, 05 Apr 2019 15:34:01 GMT.`
	`Content-Length: 0.`
	`Authorization: AWS akey2:jTcwd1Ta+5sZftVHGtEEyweojdk=.`
	`User-Agent: Boto/2.42.0 Python/2.7.5 Linux/3.10.0-693.el7.x86_64.`
	
	`HTTP/1.1 200 OK.`
	`Date: Fri, 05 Apr 2019 15:34:01 GMT.`
	`x-amz-request-id: 1721b414-267b-1341-93e6-d4ae52ce5402.`
	`Content-Type: application/xml;charset=UTF-8.`
	`Content-Length: 432.`
	`Server: CloudianS3.`
	
	`<?xml version="1.0" encoding="UTF-8"?><ListAllMyBucketsResult`
	`xmlns="http://s3.amazonaws.com/doc/2006-03-01/">`
	`<Owner><ID>8ce1c49e532edc91b0a43e0c7e7d5975</ID>`
	`<DisplayName>robot1</DisplayName></Owner>`
	`<Buckets><Bucket><Name>sharedbucket1</Name>`
	`<CreationDate>2019-04-05T15:30:03.897Z</CreationDate></Bucket>`
	`<Bucket><Name>sharedbucket2</Name>`
	`<CreationDate>2019-04-05T15:27:26.300Z</CreationDate>`
	`</Bucket></Buckets></ListAllMyBucketsResult>`
		No

Name	Description	Required
	Note When the 'shared' parameter is used, the *ListBuckets* call returns only buckets that have been shared with the requesting user -- **not buckets owned by the requesting user**. So to retrieve all buckets that a user has access to, an S3 client application must submit two *ListBuckets* calls -- one without the 'shared' parameter (to retrieve the user's own buckets) and one with the 'shared' parameter (to retrieve buckets that have been shared with the user). **Note** When the 'shared' parameter is used, in the *ListBuckets* response body the "Owner" is the requesting user, not the actual owner(s) of the shared bucket(s).	

1.2.48. ListMultipartUploads

This operation lists in-progress multipart uploads.

Along with the **common headers**, HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: **ListMultipartUploads**

Former operation name: List Multipart Uploads

1.2.48.1. Query Parameters

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

1.2.48.2. Response Elements

- ListMultipartUploadsResult
  - Bucket
  - KeyMarker
  - UploadIdMarker
  - NextKeyMarker
  - Prefix
  - Delimiter
  - NextUploadIdMarker
  - MaxUploads

- IsTruncated
- Upload
  - Initiated
  - Initiator
    - DisplayName
    - ID
  - Key
  - Owner
    - DisplayName
    - ID
  - StorageClass
  - UploadId
- CommonPrefixes
  - Prefix
- EncodingType

1.2.49. ListObjects

Returns some or all of the objects in a bucket.

Along with the **common headers**, HyperStore supports the operation-specific parameters, headers, and elements listed below.

For operation details and examples see the AWS documentation: **ListObjects**

Former operation name: GET Bucket (List Objects) Version 1

Note HyperStore also supports the newer version of this API operation, **ListObjectsV2**.

Note When using ListObjects, use the *marker* request parameter to improve performance in listing the content of buckets that contain many objects. For detail see the AWS documentation for this API operation.

1.2.49.1. Query Parameters

- delimiter

  Note The HyperStore system does not support %c2%85(U+0085) as a delimiter value

- encoding-type
- marker
- max-keys
- prefix

Note The HyperStore S3 extension request parameter *meta=true* is no longer supported.

1.2.49.2. Response Headers

HyperStore Extension to the S3 API

The HyperStore system supports the following Response Header as an extension to the "ListObjects" operation:

Name	Description	Required
x-gmt-policyid	This header specifies the unique ID of the storage policy assigned to the bucket. For more information see **"CreateBucket"** (page 22).	No

1.2.49.3. Response Elements

- ListBucketResult
  - IsTruncated
  - Marker
  - NextMarker
  - Contents
    - ETag
    - Key
    - LastModified
    - Owner
      - DisplayName
      - ID
    - Size
    - StorageClass (values STANDARD and GLACIER only)
  - Name
  - Prefix
  - Delimiter
  - MaxKeys
  - CommonPrefixes
    - Prefix
  - Encoding-Type

1.2.50. ListObjectsV2

Returns some or all of the objects in a bucket.

Along with the **common headers**, HyperStore supports the operation-specific parameters, headers, and elements listed below.

For operation details and examples see the AWS documentation: **ListObjectsV2**

Former operation name: GET Bucket (List Objects) Version 2

Note For backward-compatibility HyperStore continues to also support the older version of this API operation, **ListObjects**.

Note When using ListObjectsV2, use the *continuation-token* request parameter to improve performance in listing the content of buckets that contain many objects. For detail see the Amazon documentation for ListObjectsV2.

1.2.50.1. Query Parameters

- continuation-token
- delimiter

  Note The HyperStore system does not support %c2%85(U+0085) as a delimiter value

- encoding-type
- fetch-owner
- list-type
- max-keys
- prefix
- start-after

Note The HyperStore S3 extension request parameter *meta=true* is no longer supported.

1.2.50.2. Response Headers

HyperStore Extension to the S3 API

The HyperStore system supports the following Response Header as an extension to the "ListObjectsV2" operation:

Name	Description	Required
x-gmt-policyid	This header specifies the unique ID of the storage policy assigned to the bucket. For more information see **"CreateBucket"** (page 22).	No

1.2.50.3. Response Elements

- ListBucketResult
  - IsTruncated
  - Contents
    - ETag
    - Key

        - LastModified
        - Owner
            - DisplayName
            - ID
        - Size
        - StorageClass (values STANDARD and GLACIER only)
    - Name
    - Prefix
    - Delimiter
    - MaxKeys
    - CommonPrefixes
        - Prefix
    - Encoding-Type
    - KeyCount
    - ContinuationToken
    - NextContinuationToken
    - StartAfter

1.2.51. ListObjectVersions

Returns metadata about all of the versions of objects in a bucket.

Along with the **common headers**, HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: **ListObjectVersions**

Former operation name: GET Bucket Object versions

1.2.51.1. Query Parameters

- delimiter
- encoding-type
- key-marker
- max-keys
- prefix
- version-id-marker

1.2.51.2. Response Elements

- ListVersionsResult
    - IsTruncated
    - KeyMarker
    - VersionIdMarker

- NextKeyMarker
- NextVersionIdMarker
- Version
    - ETag
    - IsLatest
    - Key
    - LastModified
    - Owner
        - DisplayName
        - ID
    - Size
    - StorageClass
    - VersionId
- DeleteMarker
    - IsLatest
    - Key
    - LastModified
    - Owner
        - DisplayName
        - ID
    - VersionId
- Name
- Prefix
- Delimiter
- MaxKeys
- Encoding-Type

1.2.52. ListParts

Lists the parts that have been uploaded for a specific multipart upload.

Along with the **common headers**, HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: **ListParts**

Former operation name: List Parts

1.2.52.1. Query Parameters

- key
- max-parts
- part-number-marker
- uploadId

1.2.52.2. Response Headers

- x-amz-abort-date
- x-amz-abort-rule-id

1.2.52.3. Response Elements

- ListPartsResult
  - Bucket
  - Key
  - UploadId
  - PartNumberMarker
  - NextPartNumberMarker
  - MaxParts
  - IsTruncated
  - Part
    - ETag
    - LastModified
    - PartNumber
    - Size
  - Initiator
    - DisplayName
    - ID
  - Owner
    - DisplayName
    - ID
  - StorageClass

1.2.53. OPTIONS Object

A browser can send this preflight request to HyperStore to determine if it can send an actual request with the specific origin, HTTP method, and headers.

Along with the **common headers**, HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: **OPTIONS Object**

Former operation name: OPTIONS Object (no change)

1.2.53.1. Request Headers

- Access-Control-Request-Headers
- Access-Control-Request-Method
- Origin

1.2.53.2. Response Headers

- Access-Control-Allow-Headers
- Access-Control-Allow-Methods
- Access-Control-Allow-Origin
- Access-Control-Expose-Headers
- Access-Control-Max-Age

1.2.54. POST Object

The POST operation adds an object to a specified bucket using HTML forms.

Along with the **common headers**, HyperStore supports the operation-specific form fields listed below.

For operation details and examples see the AWS documentation: **POST Object**

Former operation name: POST Object (no change)

1.2.54.1. Form Fields

- AWSAccessKeyId
- acl
- Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires
- file
- key
- policy
- success_action_redirect, redirect
- success_action_status
- tagging
- x-amz-storage-class

  Note HyperStore ignores the value of the *x-amz-storage-class* field and treats all requests as being for storage class STANDARD.

- x-amz-meta-*
- x-amz-website-redirect-location
- x-amz-object-lock-mode
- x-amz-object-lock-retain-until-date
- x-amz-object-lock-legal-hold
- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key
- x-amz-server-side-encryption-customer-key-MD5

1.2.54.2. Response Headers

- success_action_redirect, redirect
- x-amz-expiration
- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key-MD5
- x-amz-version-id

1.2.54.3. Response Elements

- Bucket
- ETag
- Key
- Location

1.2.55. PutBucketAcl

Sets the permissions on an existing bucket using access control lists (ACL).

Along with the **common headers**, HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: **PutBucketAcl**

Former operation name: PUT Bucket acl

1.2.55.1. Request Headers

- x-amz-acl
- x-amz-grant-full-control
- x-amz-grant-read
- x-amz-grant-read-acp
- x-amz-grant-write
- x-amz-grant-write-acp

1.2.55.2. Request Elements

- AccessControlPolicy
    - AccessControlList
        - Grant
            - Grantee
                - DisplayName
                - ID
        - Permission

    - Owner
        - DisplayName
        - ID

1.2.56. PutBucketCors

Sets the *cors* configuration for your bucket.

Along with the **common headers**, HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: **PutBucketCors**

Former operation name: PUT Bucket cors

1.2.56.1. Request Headers

- Content-MD5

1.2.56.2. Request Elements

- CORSConfiguration
    - CORSRule
        - AllowedHeader
        - AllowedMethod
        - AllowedOrigin
        - ExposeHeader
        - ID
        - MaxAgeSeconds

1.2.57. PutBucketEncryption

This implementation of the PUT operation uses the *encryption* subresource to set the default encryption state of an existing bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **PutBucketEncryption**

Former operation name: PUT Bucket encryption

Note In the current HyperStore release, **only the bucket owner is allowed to perform operations relating to bucket encryption**. HyperStore does not currently support the use of bucket policies to extend bucket encryption permissions to users other than the bucket owner. Specifically, with regard to **"PutBucketPolicy"** (page 65), HyperStore does not currently support the "s3:PutEn-cryptionConfiguration" or "s3:GetEncryptionConfiguration" actions.

1.2.57.1. Request Elements

- ServerSideEncryptionConfiguration
    - Rule
        - ApplyServerSideEncryptionByDefault
            - KMSMasterKeyID
            - SSEAlgorithm

1.2.58. PutBucketInventoryConfiguration

This implementation of the PUT action adds an inventory configuration (identified by the inventory ID) to the bucket.

Along with the **common headers**, HyperStore supports the operation-specific parameter and elements listed below.

For operation details and examples see the AWS documentation: **PutBucketInventoryConfiguration**

Note Unlike AWS, HyperStore does not use a system account to write inventory reports to the destination bucket. Instead, reports are written by source bucket owner's account. In the current version of HyperStore, the **report destination bucket must be a bucket that is owned by the source bucket owner**.

1.2.58.1. Query Parameter

- id

1.2.58.2. Request Elements

- InventoryConfiguration
    - Destination
        - S3BucketDestination
            - AccountId
            - Bucket
            - Encryption

              Note HyperStore currently does not support encryption of bucket inventory reports. If you include the optional "Encryption" element in the request body HyperStore will ignore it.
            - Format

              Note HyperStore currently only supports CSV format.
            - Prefix

- Filter
  - Prefix
- Id
- IncludedObjectVersions
- IsEnabled
- OptionalFields
  - Field
- Schedule
  - Frequency

1.2.59. PutBucketLifecycle

Creates a new lifecycle configuration for the bucket or replaces an existing lifecycle configuration.

For operation details and examples see the AWS documentation: **PutBucketLifecycle**

Note Though HyperStore supports this API operation for backward compatibility, AWS has deprecated this operation in favor of a newer version called **PutBucketLifecycleConfiguration** which HyperStore also supports. For new lifecycle configurations use the new version.

1.2.60. PutBucketLifecycleConfiguration

Creates a new lifecycle configuration for the bucket or replaces an existing lifecycle configuration.

Along with the **common headers**, HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: **PutBucketLifecycleConfiguration**

Former operation name: PUT Bucket lifecycle

Note With the HyperStore system, only the bucket owner can create bucket lifecycle rules.

1.2.60.1. Request Headers

HyperStore Extension to the S3 API

The HyperStore system supports the following Request Headers as extensions to the "PutBucketLifecycleConfiguration" operation:

Note Do not set an auto-tiering lifecycle rule and a **cross-region replication** configuration on the same source bucket.

Name	Description	Require-d
x-gmt-tieringinf-	The *x-gmt-tieringinfo* request header enables you to configure a bucket for schedule-based automatic transitioning of objects from local HyperStore storage to a remote stor-	No

Name	Description	Require-d
o	age system. The *x-gmt-tieringinfo* header is formatted as follows: `x-gmt-tieringinfo: PROTOCOL|EndPoint:Endpoint,Action:Action [,Mode:proxy][,Region:Region][,TieringBucket:TieringBucket]` • *PROTOCOL* (mandatory) — Specify one of these values, in all caps: ◦ S3 -- Transition the objects to Amazon S3 storage. ◦ S3GLACIER -- Transition the objects to Amazon Glacier. ◦ GCS -- Transition the objects to Google Cloud Storage. ◦ AZURE -- Transition the objects to Microsoft Azure. ◦ SPECTRA -- Transition the objects to a Spectra Logic BlackPearl destination. **Note** If you are tiering to an S3-compliant system other than Amazon S3, Glacier, or Google Cloud Storage, use "S3" as the protocol. This would include, for instance, **tiering to a remote HyperStore region or system**. **Note** Auto-tiering restrictions based on destination type: * Tiering to Azure, Google Cloud, or Spectra BlackPearl is not supported for source buckets that have versioning enabled or that have had versioning enabled in the past. * When auto-tiering to Spectra BlackPearl is used for a bucket, objects in the bucket will not be auto-tiered unless they are larger than 5MB. Objects 5MB or smaller will remain in HyperStore. • EndPoint:*EndPoint* (mandatory) — The service endpoint URL to use as your auto-tiering destination. For example with Amazon S3, choose the region endpoint that’s most suitable for your location (such as *s3-us-west-1.amazonaws.com* if your organization is in northern California). Or in the case of Spectra BlackPearl, specify the URL for your Spectra BlackPearl destination. If your ultimate tiering destination is Glacier, you must specify an Amazon S3 endpoint here, **not** a Glacier endpoint. The HyperStore system will first transition the objects to your specified Amazon S3 endpoint and then from there they will be transitioned to the corresponding Glacier location. **Note You must use nested URL encoding**. First URL encode the Endpoint value (the endpoint itself), and then URL encode the whole *x-gmt-tieringinfo* value. **Note** Once you've configured an auto-tiering lifecycle on a source bucket you cannot subsequently change the tiering endpoint for that	

Name	Description	Require-d
	source bucket. • Action:*Action* (mandatory) — This parameter specifies how the HyperStore system will handle S3 *GET Object* requests for objects that have been transitioned to the tiering destination. The choices are: ◦ *stream* — If the client submits a *GET Object* request to HyperStore, the HyperStore system retrieves the object from the destination and streams it through to the client. This method is supported only if the *Protocol* is S3, GCS, or AZURE. ◦ *nostream* — If the client submits a *GET Object* request to HyperStore, the HyperStore system rejects the GET request. Instead, clients must submit a *POST Object restore* request in order to temporarily restore a copy of the object to local HyperStore storage. ◦ *cache* -- The local system GETs the object from the tiering destination system and immediately streams it through to the client, and simultaneous saves a copy of the object to local storage. The local copy is kept in local storage -- cached -- for a period that depends on how the auto-tiering policy is configured: ▪ If the tiering policy uses a "Days After..." configuration, the caching retention period is the same as the number of days that triggers the tiering of objects in the first place. For example, if the auto-tiering policy is configured to tier objects 30 days after object creation, and "Cache (Stream & Restore)" mode is selected for handling GETs of tiered objects, then when there's a GET request for a tiered object the object is streamed to the client and also cached locally for 30 days. ▪ If the tiering policy uses a "After Date..." configuration, the cached local copy is retained only until the next running of the auto-tiering / auto-expiration cron job (which runs once a day). The same caching behavior applies to GETs of objects that were tiered by "Bridge Mode" (proxy mode). During the period while the object is cached locally, subsequent GETs of the object can be served from local storage. After the cache period expires, the local copy is automatically deleted by the next run of the daily auto-tiering / auto-expiration cron job. Following deletion of the cached copy, the next GET of the object will be served from the tiering destination site (and a copy of the object will be once again be cached). If the *Protocol* is S3, GCS, or AZURE you can use either "stream" or "nostream" or "cache". If the *Protocol* is S3GLACIER or SPECTRA you must use "nostream" (the "stream" and "cache" options are not supported for those destinations). • Mode:proxy (optional) — If you specify this option, then: ◦ **All objects uploaded to the bucket from this time forward** (all objects uploaded after you successfully submit the *PUT Bucket lifecycle* request) will be **immediately** transitioned to the destination system.	

Name	Description	Require-d

	◦ Any objects already in the bucket at the time that you submit the *PUT Bucket lifecycle* request will be subject to the transition schedule that you define in the request body.
	
	Proxy mode is supported only if the *Protocol* is S3, GCS, or AZURE (proxy mode is not supported for S3GLACIER or SPECTRA tiering).
	
	• Region: *Region* (optional) — If you are tiering to an *S3* protocol endpoint other than Amazon (for example a remote HyperStore destination) and you are having the system create a destination bucket for you to tier to -- meaning you are leaving the "TieringBucket" field empty or you are specifying the name of a bucket that does not yet exist -- use the "Region" field to indicate the destination service region in which you want the bucket created.
	• TieringBucket:*TieringBucket* (optional) — The name of the bucket to transition objects into, in the tiering destination system. This can be either:
	◦ The **name of a bucket that already exists in the destination system**, and for which you are the bucket owner. In this case HyperStore will use this existing bucket as the tiering destination.
	◦ The **name of a bucket that you want HyperStore to create in the destination system**, to use as the tiering destination. Be sure to choose a bucket name that is very likely to be unique in the destination system. If your supplied bucket name is not unique in the destination system, HyperStore will be unable to create the bucket and the *PUT Bucket lifecycle* request will fail.
	
	If you **omit** the tiering bucket parameter, then in the destination system HyperStore will create a tiering bucket named as follows:
	
	<origin-bucket-name-truncated-to-34-characters>-<28-character-random-string>
	
	Example *x-gmt-tieringinfo* request headers:
	`# Example 1 (before URL encoding) Tiering to Amazon S3, into target bucket`
	`# named 'bucket12'. Streaming for local GETs will be supported.`
	
	`x-gmt-tieringinfo: S3|EndPoint:http://s3.amazonaws.com,Action:stream,`
	`TieringBucket:bucket12`
	
	`# Example 1 after nested URL encoding (endpoint value first, then whole`
	`# header value)`
	
	`x-gmt-tieringinfo: S3%7CEndPoint%3Ahttp%253A%252F%252Fs3.amazonaws.com`
	`%2CAction%3Astream%2CTieringBucket%3Abucket12`
	
	`# Example 2 (before URL encoding) Tiering to Azure. HyperStore will derive`
	`target`
	`# bucket name from source bucket name. Streamed local GETs will not be`
	`supported,`	

Name	Description	Require-d
	```
	# clients must use Restore.

	x-gmt-tieringinfo:
	AZURE|EndPoint:https://blob.core.windows.net,Action:nostream

	# Example 2 after nested URL encoding (endpoint value first, then whole
	# header value)

	x-gmt-tieringinfo:
	AZURE%7CEndPoint%3Ahttps%253A%252F%252Fblob.core.windows.net
	%2CAction%3Anostream
	```
x-gmt-com-pare	If you include this header in your "PUT Bucket lifecycle" request and set the header value to "LAT", then in lifecycle rules that you configure with the "Days" comparator the rule will be implemented as number of days since the object's **Last Access Time**. If you do not use this extension header, or if you include the header but assign it no value or any value other than "LAT", then "Days" based lifecycle rules will be imple-mented as number of days since the object's Creation Time (the default Amazon S3 behavior). You can use this header to create: • Last Access Time based auto-tiering rules (use this header and also the *x-gmt-tierinfo* header). • Last Access Time based expiration rules (use this header but not the *x-gmt-tier-info* header). **Note** Last Access Time is subject to updating only for objects in buckets for which a lifecycle is configured. For such objects, Last Access Time is updated if the object is accessed either for retrieval (GET or HEAD) or modification (PUT/POST/Copy). If an object is created and then never accessed, its Last Access Time will be its Creation Time. **Note** If you use the *x-gmt-compare* header and set it to "LAT", it does not apply to any in *NoncurrentVersionTransition* or *NoncurrentVersionExpiration* rules within the lifecycle policy (for non-current versions of versioned objects). These types of rules are always based on the time elapsed since an object ver-sion became non-current (was replaced by a new version of the object).	No
x-gmt-post-tier-copy	If you use the *x-gmt-tieringinfo* request header to configure auto-tiering for a bucket, you can optionally also use the *x-gmt-post-tier-copy* request header to **specify a num-ber of days for which a local copy of auto-tiered objects should be retained**. For example if you set *x-gmt-post-tier-copy: 7* then after each object is auto-tiered to the tiering destination, a copy of the object will be kept in the HyperStore source bucket for 7 days. After that the local copy will be deleted and only object metadata will be retained locally. There is no upper limit on this value. So if you want the local copy retention period to	No

Name	Description	Require-d
	be practically limitless, you could for example set this header to 36500 to indicate a local copy retention period of 100 years. If you **omit** the *x-gmt-post-tier-copy* request header, then by default local objects are deleted after they are successfully auto-tiered to the tiering destination system, and only object metadata is retained locally.	

1.2.60.2. Request Elements

- LifecycleConfiguration
  - Rule
    - AbortIncompleteMultipartUpload
      - DaysAfterInitiation
    - Expiration
      - Date
      - Days
      - ExpiredObjectDeleteMarker
    - Filter
      - And
        - ObjectSizeGreaterThan
        - ObjectSizeLessThan
        - Prefix
        - Tag
          - Key
          - Value
      - Prefix
      - Tag
        - Key
        - Value
    - ID
    - NoncurrentVersionExpiration
      - NoncurrentDays
    - NoncurrentVersionTransition
      - NoncurrentDays
      - StorageClass
    - Prefix
    - Status

            - Transition
                - Date
                - Days
                - StorageClass

Note If you are using "Bridge Mode" transition (whereby objects are auto-tiered immediately after being uploaded to HyperStore), leave the "Prefix" attribute empty. Bridge Mode does not support filtering by prefix. Also, Bridge Mode does not support filtering by tag(s).

1.2.61. PutBucketLogging

Set the logging parameters for a bucket and to specify permissions for who can view and modify the logging parameters.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **PutBucketLogging**

Former operation name: PUT Bucket logging

Note For a bucket that has bucket logging enabled, bucket logs (server access logs) are generated every 10 minutes by a HyperStore system cron job, if there was activity for that bucket during that interval.

Note If you are using bucket logging in your service, and if you use a load balancer in front of your S3 Service nodes, you should configure your S3 Service to support the HTTP X-Forwarded-For header. This will enable bucket logs to record the true originating IP address of S3 requests, rather than the load balancer IP address. By default the S3 Service does not support the X-Forwarded-For header. You can enable support for this header using the system configuration file *s3.xml.erb*.

1.2.61.1. Request Elements

- BucketLoggingStatus
    - LoggingEnabled
        - TargetBucket
        - TargetGrants
            - Grant
                - Grantee
                    - DisplayName
                    - EmailAddress
                    - ID
                - Permission
        - TargetPrefix

1.2.62. PutBucketNotificationConfiguration

Enables notifications of specified events for a bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **PutBucketNotificationConfiguration**

Note In the current HyperStore release, only the bucket owner is allowed to submit this request and **the bucket owner must also be the owner of the destination Queue**.

Note HyperStore's bucket notification feature and its SQS Service (for notification message queueing and delivery) are **disabled by default**. Before you can use this feature it must be enabled by a system administrator.

1.2.62.1. Request Elements

- NotificationConfiguration
  - QueueConfiguration
    - Event
    - Filter
      - S3Key
        - FilterRule
          - Name
          - Value
    - Id
    - Queue

For Event types, HyperStore supports only the following:

- s3:ObjectCreated:*
- s3:ObjectCreated:Put
- s3:ObjectCreated:Post
- s3:ObjectCreated:Copy
- s3:ObjectCreated:CompleteMultipartUpload
- s3:ObjectRemoved:*
- s3:ObjectRemoved:Delete
- s3:ObjectRemoved:DeleteMarkerCreated

1.2.63. PutBucketOwnershipControls

Creates or modifies *OwnershipControls* for a bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **PutBucketOwnershipControls**

1.2.63.1. Request Elements

- OwnershipControls
  - Rule
    - ObjectOwnership

1.2.64. PutBucketPolicy

Applies an S3 bucket policy to an S3 bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **PutBucketPolicy**

For examples of the kinds of things you can do with bucket policies, see the AWS documentation: **Bucket Policy Examples**

Former operation name: PUT Bucket policy

1.2.64.1. Request Elements

The request body is a JSON-formatted bucket policy containing one or more policy statements.

- Version (**must be** "2012-10-17")
- Id
- Statement

  Within a policy's *Statement* block(s), HyperStore support for policy statement elements and their values is as follows:

  - Sid -- Same as AWS: Custom string identifying the statement, for example "Statement1" or "Only allow access from partner source IPs"
  - Effect -- Same as AWS: "Allow" or "Deny"
  - Principal -- The following formats are supported:
    - "*" -- Statement applies to all users (also known as "anonymous access").
    - {**"CanonicalUser":** "<canonicalUserId>"} -- Statement applies to the specified HyperStore account root user and to IAM users under that account.
    - {**"CanonicalUser":** ["<canonicalUserId>", "<canonicalUserId>",...]} -- Statement applies to the specified HyperStore account root users and to IAM users under those accounts.
    - {**"AWS":"arn:aws:iam::**<canonicalUserId>:root"} -- Statement applies to the specified HyperStore account root user and to IAM users under that account.
    - {**"AWS":"arn:aws:iam::**<canonicalUserId>:user/<iamUserName>"} -- Statement applies to the specified IAM user. In this format the <canonicalUserId> is that of the parent account root user.

      Note
      * Be careful to correctly spell the principal type (**highlighted in bold above**).

      * A HyperStore user's canonical ID can be obtained by using the Admin API

method *GET /user*. Details are in the Cloudian HyperStore Admin API Reference.
* In formats of the "AWS":"arn:aws:iam::...." type, AWS uses "Account Id" to identify the account root user. In HyperStore the canonical user ID is used for this purpose, since in HyperStore there is not a separate account ID that's different than the canonical user ID.
* To access a resource, an **IAM user** must also be granted access to the resource by an **IAM policy**. A bucket policy by itself is not sufficient to grant an IAM user access to resources within the bucket. This is consistent with AWS behavior.

- NotPrincipal -- Same usage as AWS.
- Action -- See details below.
- NotAction -- Same usage as AWS.
- Resource -- Same as AWS; must be one of:
  - "arn:aws:s3:::<bucketName>" -- For bucket actions (such as "s3:ListBucket") and bucket subresource actions (such as "s3:GetBucketAcl").
  - "arn:aws:s3:::<bucketName>/*" or "arn:aws:s3:::<bucketName>/<objectName>" -- For object actions (such as "s3:PutObject").
- NotResource -- Same usage as AWS.
- Condition -- See details below.

Supported "Action" Values

Within bucket policy statements, HyperStore supports **only** the following *Action* values (also known as permission keywords).

Note For information about how to use *Action* values in a bucket policy, see the AWS documentation on **Specifying Permissions in a Policy**.

Object Actions

- s3:AbortMultipartUpload
- s3:BypassGovernanceRetention
- s3:DeleteObject
- s3:DeleteObjectTagging
- s3:DeleteObjectVersion
- s3:DeleteObjectVersionTagging
- s3:GetObject
- s3:GetObjectAcl
- s3:GetObjectLegalHold
- s3:GetObjectRetention
- s3:GetObjectTagging
- x3:GetObjectTorrent
- s3:GetObjectVersion

- s3:GetObjectVersionAcl
- s3:GetObjectVersionTagging
- s3:ListMultipartUploadParts
- s3:PutObject
- s3:PutObjectAcl
- s3:PutObjectLegalHold
- s3:PutObjectRetention
- s3:PutObjectTagging
- s3:PutObjectVersionAcl
- s3:PutObjectVersionTagging
- s3:RestoreObject

Bucket Actions

- s3:CreateBucket
- s3:DeleteBucket
- s3:ListBucket
- s3:ListBucketMultipartUploads
- s3:ListBucketVersions

Bucket Subresource Actions

- s3:DeleteBucketPolicy
- s3:DeleteBucketWebsite
- s3:GetBucketAcl
- s3:GetBucketCORS
- s3:GetBucketLocation
- s3:GetBucketLogging
- s3:GetBucketNotification
- s3:GetBucketObjectLockConfiguration
- s3:GetBucketPolicy
- s3:GetBucketRequestPayment
- s3:GetBucketTagging
- s3:GetBucketVersioning
- s3:GetBucketWebsite
- s3:GetInventoryConfiguration
- s3:GetLifecycleConfiguration
- s3:GetReplicationConfiguration
- s3:PutBucketAcl
- s3:PutBucketCORS
- s3:PutBucketLogging
- s3:PutBucketNotification

- s3:PutBucketObjectLockConfiguration
- s3:PutBucketPolicy
- s3:PutBucketRequestPayment
- s3:PutBucketTagging
- s3:PutBucketVersioning
- s3:PutBucketWebsite
- s3:PutInventoryConfiguration
- s3:PutLifecycleConfiguration
- s3:PutReplicationConfiguration

Note Like AWS, the HyperStore system supports the use of a wildcard in your Action configuration (*"Action":["s3:*"]*). When an Action wildcard is used together with an object-level Resource element (*"arn:aws:s3:::<bucketName>/*"* or *"arn:aws:s3:::<bucketName>/<objectName>"*), the wildcard denotes all the Object actions **that HyperStore supports**. When an Action wildcard is used together with bucket-level Resource element (*"arn:aws:s3:::<bucketName>"*), the wildcard denotes all the Bucket actions and Bucket Subresource actions **that HyperStore supports**.

Supported "Condition" Values

Within bucket policy statements, HyperStore supports **only** the following *Condition* operators and keys.

Note For information about how to use condition operators and keys in a bucket policy, see the AWS documentation on **Specifying Conditions in a Policy**.

Condition Operators

- ForAllValues:StringLike
- ForAnyValue:StringLike
- IpAddress

  Note If the HyperStore system uses load balancers in front of the HyperStore S3 Service, then IP address based bucket policies will only work if system administrators have enabled the use of PROXY Protocol between the load balancers and the S3 Service.

- NotIpAddress
- NumericEquals
- NumericNotEquals
- NumericLessThan
- NumericLessThanEquals
- NumericGreaterThan
- NumericGreaterThanEquals
- StringEquals
- StringNotEquals

- StringEqualsIgnoreCase
- StringNotEqualsIgnoreCase
- StringLike
- StringNotLike

Condition Keys

- "aws:CurrentTime"
- "aws:EpochTime"
- "aws:Referer"
- "aws:SecureTransport"
- "aws:SourceIp"
- "aws:TokenIssueTime"
- "aws:UserAgent"
- "s3:delimiter"
- "s3:ExistingObjectTag"
- "s3:LocationConstraint"
- "s3:max-keys"
- "s3:object-lock-legal-hold"
- "s3:object-lock-mode"
- "s3:object-lock-remaining-retention-days"
- "s3:object-lock-retain-until-date"
- "s3:prefix"
- "s3:RequestObjectTag"
- "s3:RequestObjectTagKeys"
- "s3:VersionId"
- "s3:x-amz-acl"
- s3:x-amz-content-sha256

  Note To deny access via pre-signed URLs you can create a Deny policy statement in which the condition denied is "s3:x-amz-content-sha256": "UNSIGNED-PAYLOAD".

- "s3:x-amz-copy-source"
- "s3:x-amz-grant-full-control"
- "s3:x-amz-grant-read"
- "s3:x-amz-grant-read-acp"
- "s3:x-amz-grant-write"
- "s3:x-amz-grant-write-acp"
- "s3:x-amz-metadata-directive"
- "s3:x-amz-object-ownership"
- "s3:x-amz-server-side-encryption"

- "s3:x-amz-server-side-encryption-aws-kms-key-id"
- "saml:namequalifier"
- "saml:sub"
- "saml:sub_type"

1.2.65. PutBucketReplication

Creates a replication configuration or replaces an existing one.

Along with the **common headers**, HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: **PutBucketReplication**

Former operation name: PUT Bucket replication

Note
* Unlike Amazon S3, HyperStore does not require that you set up an IAM Role (or anything analogous) in order to use bucket replication. Also, HyperStore does not require that the destination bucket be in a different region than the source bucket. With HyperStore you can replicate to a destination bucket that's in the same region as the source bucket, if you want to.
* Like Amazon S3, HyperStore bucket replication requires that **versioning must be enabled** (using the **"PutBucketVersioning"** (page 72) operation) on both the source bucket and the destination bucket.
* Do not set a cross-region replication configuration and a **bucket lifecycle rule for auto-tiering** on the same source bucket.

1.2.65.1. Request Headers

- Content-MD5

HyperStore Extension to the S3 API

The HyperStore system supports the following Request Headers as extensions to the "PUT Bucket replication" operation. Typically these headers are not needed for bucket replication. These headers are required only in a scenario where you want data to be replicated to a destination bucket in an external S3-compatible system (rather than in a service region within the same HyperStore system as the source bucket). Before using these extensions you should review (or have a system administrator provide to you) the "Cross-System Replication" section in the Cloudian HyperStore Administrator's Guide, including the limitations and caveats noted in that section.

Name	Description	Required
x-gmt-crr-end-point	Service endpoint of the destination S3 service, in format *<protocol>://<endpoint>:<port>*. For example *https://s3.amazonaws:443*. Since security credentials will be transmitted in this request (see "x-gmt-crr-credentials" below), HTTPS is the recommended protocol rather than regular HTTP. HyperStore does not force HTTPS use here, but for security HTTPS is advisable. This header is required only if the destination bucket is not in the same HyperStore system as the source bucket. Do not use this header if the destination bucket is in the same HyperStore system as the source bucket.	See description

Name	Description	Required
x-gmt-crr-credentials	Access key and secret key for the user account that HyperStore should use to write to the destination bucket in the destination S3 system, in format *<access-key>:<secret-key>*. For example, *00caf3940d-c923c59406:Ku0bMR0H5nSA7t8N+ngP6uPPTINSxJ/Q2olCMexx*. This user account must have write permissions on the destination bucket. For example, if the destination bucket is in the Amazon S3 system, this header is used to specify the Amazon S3 access key and secret key for an account that has write permissions on the destination bucket. This header is required only if the destination bucket is not in the same HyperStore system as the source bucket. Do not use this header if the destination bucket is in the same HyperStore system as the source bucket.	See description

1.2.65.2. Request Elements

- ReplicationConfiguration
    - Role
    - Rule
        - Destination
            - Bucket
            - StorageClass
        - ID
        - Prefix
        - Status

Note
* Use the same "Bucket" value formatting as in the Amazon S3 API spec, i.e. *arn:aws:s3:::<bucketname>*.
* As with the Amazon S3 API specification, for HyperStore the "Role" element must be included in the PUT Bucket replication request. However, HyperStore ignores the "Role" element's value (so, you can use any random string as its value). HyperStore does not use an IAM role or anything analogous when implementing cross-region replication.
* If you include the "StorageClass element" in the request, HyperStore ignores its value.

1.2.66. PutBucketTagging

Sets the tags for a bucket.

Along with the **common headers**, HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: **PutBucketTagging**

Former operation name: PUT Bucket tagging

Bucket tagging is subject to these restrictions:

- For each bucket, each tag key must be unique, and each tag key can have only one value.
- Maximum number of tags per bucket – 50
- Maximum key length – 128 Unicode characters in UTF-8
- Maximum value length – 256 Unicode characters in UTF-8

1.2.66.1. Request Headers

- Content-MD5

1.2.66.2. Request Elements

- Tagging
  - TagSet
    - Tag
      - Key
      - Value

1.2.67. PutBucketVersioning

Sets the versioning state of an existing bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **PutBucketVersioning**

Former operation name: PUT Bucket versioning

Note Do not enable versioning on a bucket that is configured for auto-tiering to Azure, Google Cloud, or Spectra BlackPearl. Auto-tiering to these destinations will not work properly for buckets that have versioning enabled.

1.2.67.1. Request Elements

- VersioningConfiguration
  - Status

1.2.68. PutBucketWebsite

Sets the configuration of the website that is specified in the *website* subresource.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **PutBucketWebsite**

Former operation name: PUT Bucket website

1.2.68.1. Request Elements

- WebsiteConfiguration
    - ErrorDocument
        - Key
    - IndexDocument
        - Suffix
    - RedirectAllRequestsTo
        - HostName
        - Protocol

1.2.69. PutObject

Adds an object to a bucket.

Along with the **common headers**, HyperStore supports the operation-specific headers listed below.

For operation details and examples see the AWS documentation: **PutObject**

Former operation name: PUT Object

1.2.69.1. Request Headers

- Cache-Control
- Content-Disposition
- Content-Encoding
- Expires
- x-amz-acl
- x-amz-grant-full-control
- x-amz-grant-read
- x-amz-grant-read-acp
- x-amz-grant-write
- x-amz-grant-write-acp
- x-amz-meta-*
- x-amz-object-lock-legal-hold
- x-amz-object-lock-mode
- x-amz-object-lock-retain-until-date
- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key
- x-amz-server-side-encryption-customer-key-MD5
- x-amz-storage-class

Note HyperStore ignores the value of the *x-amz-storage-class* header and treats all requests as being for storage class STANDARD.

- x-amz-tagging
- x-amz-website-redirect-location

HyperStore Restrictions on Object Names

The following control characters cannot be used anywhere in an object name and will result in a 400 Bad Request response: 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A ("\n"), 0x0B, 0x0C, 0x0D ("\r"), 0x0E, 0x0F, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1A, 0x1B, 0x1C, 0x1D, 0x1E, 0x1F

Also unsupported are:

- 0x09 ("\t") at the beginning of an object name
- 0xBF (inverted question mark) at the end of an object name
- Object names consisting **only** of "." or only of ".."
- Object names containing a **combination** of "." and "/", or a combination of ".." and "/"

1.2.69.2. Response Headers

- x-amz-expiration
- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key-MD5
- x-amz-version-id

1.2.70. PutObjectAcl

Uses the *acl* subresource to set the access control list (ACL) permissions for an object that already exists in a bucket.

Along with the **common headers**, HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: **PutObjectAcl**

Former operation name: PUT Object acl

1.2.70.1. Request Headers

- x-amz-acl
- x-amz-grant-full-control
- x-amz-grant-read
- x-amz-grant-read-acp
- x-amz-grant-write
- x-amz-grant-write-acp

1.2.70.2. Request Elements

- AccessControlPolicy
    - AccessControlList
        - Grant
            - Grantee
                - DisplayName
                - ID
            - Permission
    - Owner
        - DisplayName
        - ID

1.2.70.3. Response Headers

- x-amz-version-id

1.2.71. PutObjectLegalHold

Applies a Legal Hold configuration to the specified object.

Along with the **common headers**, HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: **PutObjectLegalHold**

Former operation name: PUT Object legal hold

1.2.71.1. Query Parameters

- versionId

1.2.71.2. Request Elements

- LegalHold
    - Status

1.2.72. PutObjectLockConfiguration

Places an Object Lock configuration on the specified bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **PutObjectLockConfiguration**

Former operation name: PUT Bucket object lock configuration

1.2.72.1. Request Elements

- ObjectLockConfiguration
    - ObjectLockEnabled
    - Rule
        - DefaultRetention
            - Days
            - Mode
            - Years

1.2.73. PutObjectRetention

Places an Object Retention configuration on an object.

Along with the **common headers**, HyperStore supports the operation-specific parameters, headers, and elements listed below.

For operation details and examples see the AWS documentation: **PutObjectRetention**

Former operation name: PUT Object retention

1.2.73.1. Query Parameters

- versionId

1.2.73.2. Request Headers

- x-amz-bypass-governance-retention

1.2.73.3. Request Elements

- Retention
    - Mode
    - RetainUntilDate

1.2.74. PutObjectTagging

Sets the supplied tag-set to an object that already exists in a bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **PutObjectTagging**

Former operation name: PUT Object tagging

Object tagging is subject to these restrictions:

- For each object, each tag key must be unique, and each tag key can have only one value.
- Maximum number of tags per object – 50

- Maximum key length – 128 Unicode characters in UTF-8
- Maximum value length – 256 Unicode characters in UTF-8

1.2.74.1. Request Elements

- Tagging
    - TagSet
        - Tag
            - Key
            - Value

1.2.75. PutPublicAccessBlock

Creates or modifies the *PublicAccessBlock* configuration for a bucket.

Along with the **common headers**, HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: **PutPublicAccessBlock**

1.2.75.1. Request Elements

- PublicAccessBlockConfiguration
    - BlockPublicAcls
    - IgnorePublicAcls
    - BlockPublicPolicy
    - RestrictPublicBuckets

HyperStore Implementation Notes

- If *BlockPublicAcls* is set to true, then:
    - Put object/bucket ACL calls will fail with access denied if any grant in the ACL is public (grantee is AllUsers or AuthenticatedUsers)
    - Put object calls will fail with access denied if any grant in the ACL is public (or if canned ACL is public)
- If *IgnorePublicAcls* is set to true, then when performing ACL permission checks during operations on the bucket or its objects, public grants are ignored (and therefore public accounts will be denied access).
- If *BlockPublicPolicy* is set to true, then a PutBucketPolicy call will fail if any statement in the policy is public. A statement is considered public if the Effect is Allow and the Principal has a wildcard -- unless there is an *IpAddress:{aws:SourceIp* condition associated with the statement that restricts the requesting source IP to one or more specified IP addresses.
- If *RestrictPublicBuckets* is set to true, then the bucket's bucket policy will be replaced with a policy that allows access to the bucket and its objects only to the bucket owner. Note that *RestrictPublicBuckets* does not restrict the use of ACLs on the bucket or its objects. To restrict ACLs use the *BlockPublicAcls* and/or *IgnorePublicAcls* settings.

1.2.76. RestoreObject

Restores a tiered object back into HyperStore.

Along with the **common headers**, HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: **RestoreObject**

Former operation name: POST Object restore

Note In the context of the HyperStore system, this standard S3 operation is for temporarily restoring a copy of an object that has been auto-tiered to a tiering destination, such as Amazon S3 or Amazon Glacier.

1.2.76.1. Request Headers

- Content-MD5

1.2.76.2. Request Elements

- RestoreRequest
  - Days
  - GlacierJobParameters
    - Tier

Note For the sake of S3 API compatibility, HyperStore's S3 Service allows the request elements *GlacierJobParameters* and *Tier* to be included in a "RestoreObject" request -- but in the current HyperStore release these elements will have no effect on how the restore request is implemented. These elements pertain only to objects tiered to Glacier, and specifically the way in which tiered objects in AWS Glacier will be temporarily retrieved into AWS S3 (for one day) so that they then can be restored from AWS S3 to your HyperStore system for your specified number of Days. For this purpose Standard retrieval (the default) from Glacier will always used.

1.2.77. UploadPart

Uploads a part in a multipart upload.

Along with the **common headers**, HyperStore supports the operation-specific headers listed below.

For operation details and examples see the AWS documentation: **UploadPart**

Former operation name: Upload Part

1.2.77.1. Request Headers

- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm

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

1.2.77.2. Response Headers

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

1.2.78. UploadPartCopy

Uploads a part by copying data from an existing object as data source.

Along with the **common headers**, HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: **UploadPartCopy**

Former operation name: Upload Part - Copy

1.2.78.1. Request Headers

- x-amz-copy-source
- x-amz-copy-source-if-match
- x-amz-copy-source-if-modified-since
- x-amz-copy-source-if-none-match
- x-amz-copy-source-if-unmodified-since
- x-amz-copy-source-range
- x-amz-source-expected-bucket-owner

1.2.78.2. Response Headers

- x-amz-copy-source-version-id
- x-amz-server-side-encryption

1.2.78.3. Response Elements

- CopyPartResult
    - ETag
    - LastModified

This page left intentionally blank

Chapter 2. IAM API

2.1. Introduction

2.1.1. HyperStore Support for the AWS IAM API

HyperStore provides **limited support** for the Amazon Web Services Identity and Access Management (IAM) API. This support enables each HyperStore user, under his or her HyperStore user account, to create IAM groups and IAM users and IAM roles. The HyperStore user -- also known as the "account root user" -- can then grant those IAM groups, users, and roles permissions to perform certain actions (such as reading or writing objects in a particular bucket or buckets). As with Amazon, the means by which a HyperStore account root user grants such permissions to IAM groups, users, and roles is by creating and attaching "managed" IAM policies to IAM groups, users, and roles, and/or by creating and embedding "inline" IAM policies for IAM groups, users, and roles. By default **newly created IAM entities have no permissions**; they gain permissions only by their association with managed or inline IAM policies.

In the HyperStore system **all S3 object data in IAM users' buckets belongs to the parent HyperStore user account**. Consequently, if an IAM user is deleted by their HyperStore parent user, the IAM user's data is not deleted from the system.

To access the IAM Service, HyperStore users **need S3 access credentials**. When users are created in Hyper-Store, S3 access credentials are automatically created for the users.

If users are using a third party IAM client to access the HyperStore IAM Service, the users can obtain their S3 access credentials by logging in to the CMC and going to the **Security Credentials** page (via the drop-down menu under the user login name). They can then supply those credentials to the third party IAM client applic-ation.

If users are using the CMC's built-in IAM client, the CMC automatically uses the user's S3 credentials to access the IAM service. Through the CMC's **IAM** section, HyperStore users can create IAM groups and users and so on.

2.1.1.1. Restrictions and Limitations in HyperStore's IAM Support

- HyperStore supports most but not all of the Amazon IAM API "Actions". For the list of supported actions see the "**Supported IAM Actions**" section.
- HyperStore supports most but not all of the Amazon IAM API policy elements, actions, resources, and condition keys. For more information see **"Supported IAM Policy Elements"** (page 121).
- Only HyperStore account root users can log into the Cloudian Management Console (CMC). The IAM users that HyperStore account root users create cannot login to the CMC, and cannot use the CMC as their S3 client application. To access the HyperStore S3 Service, IAM users must use a third party S3 cli-ent application.
- IAM users cannot use the HyperStore **SQS Service**.

2.1.1.2. IAM API Notes for HyperStore Administrators

Default 'admin' User Does Not Have S3 Access Credentials

As noted above, to access the IAM Service a user needs S3 access credentials. The pre-configured default system administrative user -- the user named "admin" -- does not have S3 access credentials by default. Consequently, if you are logged into the CMC as the "admin" user and you go to the CMC's **IAM** section you will see the following error displayed:

"No valid Access Key detected. Cannot connect to the IAM Service."

If you want to use the functionality in the CMC's **IAM** section as the "admin" user, you must create S3 credentials for this user. While logged into the CMC as the "admin" user, go to the **Security Credentials** page (via the drop-down menu under the login name). Then in the **S3 Access Credentials** section of the page, click **Create New Key**. This creates S3 access credentials for the "admin" user. Now you can use the CMC's **IAM** section without getting an access key error.

Note IAM users created by the "admin" user cannot create buckets or perform other S3 operations. Such IAM users can only perform RBAC based administrative operations (see the section below), if granted permissions by the "admin" user.

HyperStore IAM Extensions to Support RBAC for Admin Functions

The HyperStore implementation of the IAM API includes extensions that:

- Allow HyperStore system admins, group admins, or regular users to execute certain read-only HyperStore administrative functions by submitting a request to the IAM Service.
- Allow HyperStore system admins, group admins, or regular users to grant their IAM users permission to execute those same read-only HyperStore administrative functions.

For more information, including information about the client tool that HyperStore provides to help you use this feature, see "Role-Based Access to Admin API Operations" in the ***Cloudian HyperStore Admin API Reference***.

Deleting or Suspending HyperStore Users Who Have Created IAM Users

If a HyperStore user creates IAM groups, users, and/or roles, and then subsequently you **delete** that HyperStore user from the system, all IAM resources associated with that HyperStore user will also be deleted from the system. That includes IAM groups, users, roles, and policies that the HyperStore user created, the security credentials of those IAM users, and any object data that those IAM users have stored in the system.

If rather than deleting the HyperStore user you **suspend** the HyperStore user (make the user inactive), then any IAM groups, users, and/or roles that the HyperStore user created will be unable to access any HyperStore services (just like the suspended HyperStore user will be unable to access HyperStore services). If you subsequently make the HyperStore user active again, then IAM groups, users, and roles under that HyperStore user will again be able to access HyperStore services.

Disabling the HyperStore IAM Service

HyperStore's IAM Service is enabled by default, and IAM Service functionality can be accessed either through the CMC or by your third party or custom client applications. If you want to disable the IAM Service, do the following:

1. On the Configuration Master node open this configuration file in a text editor:

   ```
   /etc/cloudian-<version>-puppet/manifests/extdata/common.csv
   ```

   Note If you have a HyperStore Single-Node system, your one node is the Configuration Master.

2. Change the setting *iam_service_enabled,true* to *iam_service_enabled,false* and save your change.
3. In the installation staging directory (*/opt/cloudian-staging/7.5.2*) launch the installer.

   ```
   ./cloudianInstall.sh
   ```

4. Enter **2** for Cluster Management then **a** for Review Cluster Configuration, then enter **yes** when asked if you want to update the Puppet server with your changes.
5. Still in the installer's Cluster Management section, push your change and restart the S3 Service.

If you disable the HyperStore IAM Service, then IAM functions will no longer display in the CMC and the IAM Service will no longer accept requests from IAM client applications.

The IAM Service in Multi-Region Systems

In a multi-region HyperStore system, the IAM Service runs only on nodes in the default service region.

DNS

Be sure to configure the IAM endpoint domain in your DNS environment. For more information see the "DNS Set-Up" section of the ***Cloudian HyperStore Administrator's Guide***.

IAM Request Logging

Information about requests processed by the IAM Service are logged to *cloudian-iam-request-info.log*, which exists on each node in the default service region.For more information see the "Logging" section of the ***Cloudian HyperStore Administrator's Guide***.

2.1.2. IAM Client Application Options

Subjects covered in this section:

- *Introduction (immediately below)*
- ***"CMC Support for IAM Functions"*** *(page 83)*
- ***"Accessing the IAM Service with a Third Party Client Application"*** *(page 84)*

Users can access and use the HyperStore IAM Service either through the Cloudian Management Console (CMC) or a third party client application that supports IAM calls. Whether using the CMC or a third party client application, application users must have S3 access credentials (access key ID and secret key) in order to use the HyperStore IAM Service.

2.1.2.1. CMC Support for IAM Functions

Through the CMC, HyperStore account root users can:

- Add and manage IAM users and groups
- Add and manage IAM roles

- Add and manage IAM policies
- Add and manage SAML identity providers

2.1.2.2. Accessing the IAM Service with a Third Party Client Application

Third party or custom client applications can access the HyperStore IAM Service at these service endpoints:

```
http://iam.<organization-domain>:16080
```

```
https://iam.<organization-domain>:16443
```

HyperStore supports the standard IAM request line formatting, for example:

```
http://iam.enterprise.com:16080/?Action=<action-name>&<Parameter-name>=<value>
```

Note that:

- These are the **default** service endpoints for the HyperStore IAM Service. System administrators can customize the endpoints.
- The HyperStore IAM Service by default uses a self-signed certificate for its HTTPS listener, so if you are using HTTPS to access the service your client application must be configured to allow self-signed certificates.System administrators can replace the default self-signed certificate with a CA-signed certificate.

2.1.3. IAM Common Request Parameters

From the **"Common Parameters" section** of the AWS IAM API specification, HyperStore supports the parameters listed below. If a common parameter from that specification section is not listed below, HyperStore does not support it.

- Action
- Version

  Note Unlike Amazon's IAM implementation, in HyperStore's IAM implementation the "Version" request parameter is not required.

- X-Amz-Algorithm
- X-Amz-Credential
- X-Amz-Date
- X-Amz-Signature
- X-Amz-SignedHeaders

Note Like Amazon's IAM implementation, in HyperStore's IAM implementation you can either use query parameters or the HTTP header *Authorization* to submit the authentication data required by the Signature Version 2 or Signature Version 4 protocol. For more information on this topic see the Amazon documentation topic **Task 4: Add the Signature to the HTTP Request**.

2.1.4. IAM Common Errors

From the **"Common Errors" section** of the AWS IAM API specification, HyperStore supports the errors listed below. If a common error from that specification section is not listed below, HyperStore does not support it.

- AccessDenied
- IncompleteSignature
- InternalFailure
- InvalidAction
- InvalidClientTokenId
- InvalidParameterCombination
- InvalidParameterValue
- InvalidQueryParameter
- MalformedQueryString
- MissingAction
- MissingAuthenticationToken
- MissingParameter
- OptInRequired
- RequestExpired
- ServiceUnavailable
- ThrottlingException
- ValidationError

2.2. Supported IAM Actions

The HyperStore implementation of the AWS IAM API supports the Actions listed in this section. If an IAM Action is not listed in this section, HyperStore does not support it. For each Action, the documentation here lists the request parameters and request or response elements that HyperStore supports. If a parameter or element is not listed in this documentation, HyperStore does not support it.

For detailed descriptions of each Action and its associated parameters and elements, see the AWS documentation links.

Note For all "List" actions (such as "ListAccessKeys", "ListGroups" and so on): The HyperStore IAM Service does not support truncation. If the client request includes the "MaxItems" and "Marker" request parameters, the HyperStore IAM Service ignores those parameters. Accordingly, in the response bodies the "IsTruncated" response element will always be "false".

2.2.1. AddUserToGroup

Adds the specified user to the specified group.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **AddUserToGroup**

2.2.1.1. Request Parameters

- GroupName
- UserName

2.2.1.2. Errors

- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.2. AttachGroupPolicy

Attaches the specified managed policy to the specified IAM group.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **AttachGroupPolicy**

2.2.2.1. Request Parameters

- GroupName
- PolicyArn

2.2.2.2. Errors

- InvalidInput
- LimitExceeded
- NoSuchEntity
- PolicyNotAttachable
- ServiceFailure

2.2.3. AttachRolePolicy

Attaches the specified managed policy to the specified IAM role.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **AttachRolePolicy**

2.2.3.1. Request Parameters

- PolicyArn
- RoleName

2.2.3.2. Errors

- InvalidInput
- LimitExceeded
- NoSuchEntity
- PolicyNotAttachable
- ServiceFailure
- UnmodifiableEntity

2.2.4. AttachUserPolicy

Attaches the specified managed policy to the specified user.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **AttachUserPolicy**

2.2.4.1. Request Parameters

- PolicyArn
- UserName

2.2.4.2. Errors

- InvalidInput
- LimitExceeded
- NoSuchEntity
- PolicyNotAttachable
- ServiceFailure

2.2.5. CreateAccessKey

Creates a new secret access key and corresponding access key ID for the specified user.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **CreateAccessKey**

2.2.5.1. Request Parameters

- UserName

2.2.5.2. Response Elements

- AccessKey

2.2.5.3. Errors

- LimitExceeded
- NoSuchEntity
- ServiceFailure

Note By default the HyperStore system allows only two key pairs per IAM user. Note that an IAM user's inactive credentials (if any) count toward this limit, as well as active credentials.

2.2.6. CreateGroup

Creates a new group.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **CreateGroup**

2.2.6.1. Request Parameters

- GroupName
- Path

2.2.6.2. Response Elements

- Group

Note For HyperStore, within the "Group" object the system-generated "GroupId" attribute value will be in this format: *<Canonical-UID-of-HyperStore-User>|<IAM-groupname>*

For example: *e97eb4557aea18781f53eb2b8f7e282e|iamgroup2*

The canonical user ID is that of the HyperStore user account under which the IAM group is created. The IAM group name will be preceded by the path if any is specified when the group is created.

Similarly, the "Arn" attribute value will be in this format:
arn:aws:iam::<Canonical-UID-of-HyperStore-User>:group/<IAM-groupname>

2.2.6.3. Errors

- EntityAlreadyExists
- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.7. CreatePolicy

Creates a new managed policy under your HyperStore account.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **CreatePolicy**

2.2.7.1. Request Parameters

- Description
- Path
- PolicyDocument

  Note For information about HyperStore's IAM policy document support see **"Supported IAM Policy Elements"** (page 121).

- PolicyName

2.2.7.2. Response Elements

- Policy

2.2.7.3. Errors

- EntityAlreadyExists
- InvalidInput
- LimitExceeded
- MalformedPolicyDocument
- ServiceFailure

2.2.8. CreatePolicyVersion

Creates a new version of the specified managed policy.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **CreatePolicyVersion**

2.2.8.1. Request Parameters

- PolicyArn
- PolicyDocument

  Note For information about HyperStore's IAM policy document support see **"Supported IAM Policy Elements"** (page 121).

- SetAsDefault

2.2.8.2. Response Elements

- PolicyVersion
  - CreateDate
  - Document
  - IsDefaultVersion
  - VersionId

2.2.8.3. Errors

- InvalidInput
- LimitExceeded
- MalformedPolicyDocument
- NoSuchEntity
- ServiceFailure

2.2.9. CreateRole

Creates a new role under your account.

HyperStore supports the parameters, parameters, and errors listed below.

For action details and examples see the AWS documentation: **CreateRole**

See also the AWS documentation: **IAM Roles**

2.2.9.1. Request Parameters

- AssumeRolePolicyDocument (also known as the "trust policy")
- Description
- MaxSessionDuration
- Path
- PermissionsBoundary
- RoleName

Example AssumeRolePolicyDocument (trust policy) for a federated/SAML principal:

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRoleWithSAML",
    "Principal": {"Federated": "arn:aws:iam::123456789012:saml-provider/adfs"},
    "Condition": {"StringEquals": {"saml:aud":
"https://cmc.mycloudianhyperstore.com:8443/saml/sso"}}
  }
}
```

The example policy above says to trust SAML assertions from the "adfs" SAML Provider to use *STS:AssumeRoleWithSAML* for this role but only if the SAML assertion contains the recipient string matching *https://cmc.mycloudianhyperstore.com:8443/saml/sso*.

IMPORTANT ! The "saml:aud" value must exactly match what is submitted by the identity provider system (IdP). This value will come from this element in the **HyperStore SAML Metadata Document for IdP Setup** (emphasis added):

<AssertionConsumerService index="1" isDefault="true" Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location =**"https://<cmc FQDN or IP>:<port>/saml/sso"**/>

For *Condition* in a trust policy, HyperStore supports only the *StringEquals* condition operator and only the following condition keys:

- aws:TokenIssueTime
- sts:ExternalId
- saml:aud
- saml:doc
- saml:iss
- saml:namequalifier
- saml:sub
- saml:sub_type

2.2.9.2. Response Elements

- Role

2.2.9.3. Errors

- ConcurrentModification
- EntityAlreadyExists
- InvalidInput
- LimitExceeded
- MalformedPolicyDocument
- ServiceFailure

2.2.10. CreateSAMLProvider

Creates an IAM resource that describes an identity provider (IdP) that supports SAML 2.0.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **CreateSAMLProvider**

2.2.10.1. Request Parameters

- Name
- SAMLMetadataDocument

2.2.10.2. Response Elements

- SAMLProviderArn

2.2.10.3. Errors

- EntityAlreadyExists
- LimitExceeded
- ServiceFailure

2.2.11. CreateUser

Creates a new IAM user under your account.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **CreateUser**

2.2.11.1. Request Parameters

- Path
- UserName

2.2.11.2. Response Elements

- User

Note For HyperStore, within the "User" object the system-generated "UserId" attribute value will be in this format: *<Canonical-UID-of-HyperStore-User>|<IAM-username>*

For example: *e97eb4557aea18781f53eb2b8f7e282e|iamuser2*

The canonical user ID is that of the HyperStore user account under which the IAM user is created. The IAM user name will be preceded by the path if any is specified when the user is created.

Similarly, the "Arn" attribute value will be in this format:
arn:aws:iam::<Canonical-UID-of-HyperStore-User>:user/<IAM-username>

2.2.11.3. Errors

- EntityAlreadyExists
- LimitExceeded
- NoSuchEntity
- ServiceFailure

Note IAM users that you create under your HyperStore user account **will not be allowed to log into the CMC** or to use the CMC as their S3 client application. IAM users will need to use an S3 client application other than the CMC to access the HyperStore S3 Service.

2.2.12. CreateVirtualMFADevice

Creates a new virtual MFA device for the HyperStore account root. After creating the virtual MFA, use **"EnableMFADevice"** (page 100) to attach the MFA device to an IAM user.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **CreateVirtualMFADevice**

2.2.12.1. Request Parameters

- Path
- Tags.member.N
  - Tag
    - Key
    - Value
- VirtualMFADeviceName

2.2.12.2. Response Elements

- VirtualMFADevice
  - Base32StringSeed
  - EnableDate
  - QRCodePNG
  - SerialNumber
  - Tags.member.N
    - Tag
      - Key
      - Value
  - User
    - Arn
    - CreateDate

        - PasswordLastUsed
        - Path
        - PermissionsBoundary
            - PermissionsBoundaryArn
            - PermissionsBoundaryType
        - Tags.member.N
            - Tag
                - Key
                - Value
        - UserId
        - UserName

2.2.12.3. Errors

- ConcurrentModification
- EntityAlreadyExists
- InvalidInput
- LimitExceeded
- ServiceFailure

2.2.13. DeactivateMFADevice

Deactivates the specified MFA device and removes it from association with the user name for which it was originally enabled.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **DeactivateMFADevice**

2.2.13.1. Request Parameters

- SerialNumber
- UserName

2.2.13.2. Errors

- EntityTemporarilyUnmodifiable
- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.14. DeleteAccessKey

Deletes the access key pair associated with the specified IAM user.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **DeleteAccessKey**

2.2.14.1. Request Parameters

- AccessKeyId
- UserName

2.2.14.2. Errors

- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.15. DeleteGroup

Deletes the specified IAM group.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **DeleteGroup**

2.2.15.1. Request Parameters

- GroupName

2.2.15.2. Errors

- DeleteConflict
- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.16. DeleteGroupPolicy

Deletes the specified inline policy that is embedded in the specified IAM group.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **DeleteGroupPolicy**

2.2.16.1. Request Parameters

- GroupName
- PolicyName

2.2.16.2. Errors

- LimitExceeded
- NoSuchEntity

- ServiceFailure

2.2.17. DeletePolicy

Deletes the specified managed policy.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **DeletePolicy**

2.2.17.1. Request Parameters

- PolicyArn

2.2.17.2. Errors

- DeleteConflict
- InvalidInput
- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.18. DeletePolicyVersion

Deletes the specified version from the specified managed policy.

Note You cannot delete the default version of a policy using this operation. To delete the default version a policy, use **"DeletePolicy"** (page 96).

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **DeletePolicyVersion**

2.2.18.1. Request Parameters

- PolicyArn
- VersionId

2.2.18.2. Errors

- DeleteConflict
- InvalidInput
- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.19. DeleteRole

Deletes the specified role.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **DeleteRole**

2.2.19.1. Request Parameters

- RoleName

2.2.19.2. Errors

- ConcurrentModification
- DeleteConflict
- LimitExceeded
- NoSuchEntity
- ServiceFailure
- UnmodifiableEntity

2.2.20. DeleteRolePolicy

Deletes the specified inline policy that is embedded in the specified IAM role.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **DeleteRolePolicy**

2.2.20.1. Request Parameters

- PolicyName
- RoleName

2.2.20.2. Errors

- LimitExceeded
- NoSuchEntity
- ServiceFailure
- UnmodifiableEntity

2.2.21. DeleteSAMLProvider

Deletes a SAML provider resource in IAM.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **DeleteSAMLProvider**

2.2.21.1. Request Parameters

- SAMLProviderArn

2.2.21.2. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

2.2.22. DeleteUser

Deletes the specified IAM user.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **DeleteUser**

2.2.22.1. Request Parameters

- UserName

2.2.22.2. Errors

- DeleteConflict
- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.23. DeleteUserPolicy

Deletes the specified inline policy that is embedded in the specified IAM user.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **DeleteUserPolicy**

2.2.23.1. Request Parameters

- PolicyName
- UserName

2.2.23.2. Errors

- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.24. DeleteVirtualMFADevice

Deletes a virtual MFA device.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **DeleteVirtualMFADevice**

2.2.24.1. Request Parameters

- SerialNumber

2.2.24.2. Errors

- DeleteConflict
- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.25. DetachGroupPolicy

Removes the specified managed policy from the specified IAM group.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **DetachGroupPolicy**

2.2.25.1. Request Parameters

- GroupName
- PolicyArn

2.2.25.2. Errors

- InvalidInput
- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.26. DetachRolePolicy

Removes the specified managed policy from the specified role.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **DetachRolePolicy**

2.2.26.1. Request Parameters

- PolicyArn
- RoleName

2.2.26.2. Errors

- InvalidInput
- LimitExceeded
- NoSuchEntity
- ServiceFailure
- UnmodifiableEntity

2.2.27. DetachUserPolicy

Removes the specified managed policy from the specified user.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **DetachUserPolicy**

2.2.27.1. Request Parameters

- PolicyArn
- UserName

2.2.27.2. Errors

- InvalidInput
- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.28. EnableMFADevice

Enables the specified MFA device and associates it with the specified IAM user.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **EnableMFADevice**

2.2.28.1. Request Parameters

- AuthenticationCode1
- AuthenticationCode2
- SerialNumber
- UserName

2.2.28.2. Errors

- EntityAlreadyExists
- EntityTemparilyUnmodifiable
- InvalidAuthenticationCode
- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.29. GetGroup

Returns a list of IAM users that are in the specified IAM group.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: GetGroup

2.2.29.1. Request Parameters

- GroupName

Note The "Marker" and "MaxItems" request parameters, if submitted, are ignored.

2.2.29.2. Response Elements

- Group

  Note For HyperStore, within the "Group" object the system-generated "GroupId" attribute value will be in this format: *<Canonical-UID-of-HyperStore-User>|<IAM-groupname>*

  For example: *e97eb4557aea18781f53eb2b8f7e282e|iamgroup2*

  The canonical user ID is that of the HyperStore user account under which the IAM group was created. The IAM group name will be preceded by the path if any was specified when the group was created.

  Similarly, the "Arn" attribute value will be in this format: *arn:aws:iam::<Canonical-UID-of-HyperStore-User>:group/<IAM-groupname>*

- IsTruncated

  Note "IsTruncated" will always be "false".

- Users.member.N

2.2.29.3. Errors

- NoSuchEntity
- ServiceFailure

2.2.30. GetGroupPolicy

Retrieves the specified inline policy document that is embedded in the specified IAM group.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **GetGroupPolicy**

2.2.30.1. Request Parameters

- GroupName
- PolicyName

2.2.30.2. Response Elements

- GroupName
- PolicyDocument
- PolicyName

2.2.30.3. Errors

- NoSuchEntity
- ServiceFailure

2.2.31. GetPolicy

Retrieves information about the specified managed policy, including the policy's default version and the total number of IAM users, groups, and roles to which the policy is attached.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **GetPolicy**

2.2.31.1. Request Parameters

- PolicyArn

2.2.31.2. Response Elements

- Policy

2.2.31.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

2.2.32. GetPolicyVersion

Retrieves information about the specified version of the specified managed policy, including the policy document.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **GetPolicyVersion**

2.2.32.1. Request Parameters

- PolicyArn
- VersionId

2.2.32.2. Response Elements

- PolicyVersion

2.2.32.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

2.2.33. GetRole

Retrieves information about the specified role, including the role's path, GUID, ARN, and the role's trust policy that grants permission to assume the role.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **GetRole**

2.2.33.1. Request Parameters

- RoleName

2.2.33.2. Response Elements

- Role

2.2.33.3. Errors

- NoSuchEntity
- ServiceFailure

2.2.34. GetRolePolicy

Retrieves the specified inline policy document that is embedded with the specified IAM role.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **GetRolePolicy**

2.2.34.1. Request Parameters

- PolicyName
- RoleName

2.2.34.2. Response Elements

- PolicyDocument
- PolicyName
- RoleName

2.2.34.3. Errors

- NoSuchEntity
- ServiceFailure

2.2.35. GetSAMLProvider

Returns the SAML provider metadata document that was uploaded when the IAM SAML provider resource object was created or updated.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **GetSAMLProvider**

2.2.35.1. Request Parameters

- SAMLProviderArn

2.2.35.2. Response Elements

- CreateDate
- SAMLMetadataDocument
- ValidUntil

2.2.35.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

2.2.36. GetUser

Retrieves information about the specified IAM user, including the user's creation date, path, unique ID, and ARN.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **GetUser**

2.2.36.1. Request Parameters

- UserName

2.2.36.2. Response Elements

- User

Note For HyperStore, within the "User" object the system-generated "UserId" attribute value will be in this format: *<Canonical-UID-of-HyperStore-User>|<IAM-username>*

For example: *e97eb4557aea18781f53eb2b8f7e282e|iamuser2*

The canonical user ID is that of the HyperStore user account under which the IAM user was created.The IAM user name will be preceded by the path if any was specified when the user was created.

Similarly, the "Arn" attribute value will be in this format: *arn:aws:iam::<Canonical-UID-of-HyperStore-User>:user/<IAM-username>*

2.2.36.3. Errors

- NoSuchEntity
- ServiceFailure

2.2.37. GetUserPolicy

Retrieves the specified inline policy document that is embedded in the specified IAM user.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **GetUserPolicy**

2.2.37.1. Request Parameters

- PolicyName
- UserName

2.2.37.2. Response Elements

- PolicyDocument
- PolicyName
- UserName

2.2.37.3. Errors

- NoSuchEntity
- ServiceFailure

2.2.38. ListAccessKeys

Returns information about the access key IDs associated with the specified IAM user.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **ListAccessKeys**

2.2.38.1. Request Parameters

- UserName

2.2.38.2. Response Elements

- AccessKeyMetadata.member.N
- IsTruncated

2.2.38.3. Errors

- NoSuchEntity
- ServiceFailure

2.2.39. ListAttachedGroupPolicies

Lists all managed policies that are attached to the specified IAM group.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **ListAttachedGroupPolicies**

2.2.39.1. Request Parameters

- GroupName
- PathPrefix

2.2.39.2. Response Elements

- AttachedPolicies.member.N
- IsTruncated

2.2.39.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

2.2.40. ListAttachedRolePolicies

Lists all managed policies that are attached to the specified IAM role.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **ListAttachedRolePolicies**

2.2.40.1. Request Parameters

- PathPrefix
- UserName

2.2.40.2. Response Elements

- AttachedPolicies.member.N
- IsTruncated

2.2.40.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

2.2.41. ListAttachedUserPolicies

Lists all managed policies that are attached to the specified IAM user.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **ListAttachedUserPolicies**

2.2.41.1. Request Parameters

- PathPrefix
- UserName

2.2.41.2. Response Elements

- AttachedPolicies.member.N
- IsTruncated

2.2.41.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

2.2.42. ListEntitiesForPolicy

Lists all IAM users, groups, and roles that the specified managed policy is attached to.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **ListEntitiesForPolicy**

2.2.42.1. Request Parameters

- EntityFilter
- PathPrefix
- PolicyArn

2.2.42.2. Response Elements

- IsTruncated
- PolicyGroups.member.N
- PolicyUsers.member.N

2.2.42.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

2.2.43. ListGroupPolicies

Lists the names of the inline policies that are embedded in the specified IAM group.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **ListGroupPolicies**

2.2.43.1. Request Parameters

- GroupName

2.2.43.2. Response Elements

- IsTruncated
- PolicyNames.member.N

2.2.43.3. Errors

- NoSuchEntity
- ServiceFailure

2.2.44. ListGroups

Lists the IAM groups that have the specified path prefix.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **ListGroups**

2.2.44.1. Request Parameters

- PathPrefix

2.2.44.2. Response Elements

- Groups.member.N
- IsTruncated

2.2.44.3. Errors

- ServiceFailure

2.2.45. ListGroupsForUser

Lists the IAM groups that the specified IAM user belongs to.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **ListGroupsForUser**

Request Parameters

- UserName

Response Elements

- Groups.member.N
- IsTruncated

Errors

- NoSuchEntity
- ServiceFailure

2.2.46. ListMFADevices

Lists the MFA devices for an IAM user.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **ListMFADevices**

2.2.46.1. Request Parameters

- UserName

2.2.46.2. Response Elements

- IsTruncated
- Marker
- MFADevices.member.N
  - MFADevice
    - EnableDate
    - SerialNumber
    - UserName

2.2.46.3. Errors

- NoSuchEntity
- ServiceFailure

2.2.47. ListPolicies

Lists all the managed policies that are available under your account.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **ListPolicies**

2.2.47.1. Request Parameters

- OnlyAttached
- PathPrefix

Note The "Scope" request parameter, if submitted, is ignored and defaults to All. Note however that only Local policies are currently supported in HyperStore, so the policies returned by this command will all be Local policies.

2.2.47.2. Response Elements

- IsTruncated
- Policies.member.N

2.2.47.3. Errors

- ServiceFailure

2.2.48. ListPolicyVersions

Lists information about the versions of the specified managed policy, including the version that is currently set as the policy's default version.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **ListPolicyVersions**

2.2.48.1. Request Parameters

- PolicyArn

2.2.48.2. Response Elements

- IsTruncated
- Versions.member.N

2.2.48.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

2.2.49. ListRolePolicies

Lists the names of the inline policies that are embedded in the specified IAM role.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **ListRolePolicies**

2.2.49.1. Request Parameters

- RoleName

2.2.49.2. Response Elements

- IsTruncated
- PolicyNames.member.N

2.2.49.3. Errors

- NoSuchEntity
- ServiceFailure

2.2.50. ListRoles

Lists the IAM roles that have the specified path prefix.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **ListRoles**

2.2.50.1. Request Parameters

- PathPrefix

2.2.50.2. Response Elements

- IsTruncated
- Roles.member.N

2.2.50.3. Errors

- ServiceFailure

2.2.51. ListSAMLProviders

Lists the SAML provider resource objects defined in IAM in the account.

HyperStore supports the elements and errors listed below.

For action details and examples see the AWS documentation: **ListSAMLProviders**

2.2.51.1. Response Elements

- SAMLProviderList.member.N

2.2.51.2. Errors

- ServiceFailure

2.2.52. ListUserPolicies

Lists the names of the inline policies embedded in the specified IAM user.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **ListUserPolicies**

2.2.52.1. Request Parameters

- UserName

2.2.52.2. Response Elements

- IsTruncated
- PolicyNames.member.N

2.2.52.3. Errors

- NoSuchEntity
- ServiceFailure

2.2.53. ListUsers

Lists the IAM users that have the specified path prefix.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **ListUsers**

2.2.53.1. Request Parameters

- PathPrefix

2.2.53.2. Response Elements

- IsTruncated

  Note "IsTruncated" will always be "false".

- Users.member.N

2.2.53.3. Errors

- ServiceFailure

2.2.54. ListVirtualMFADevices

Lists the virtual MFA devices defined in the HyperStore account root by assignment status.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **ListVirtualMFADevices**

2.2.54.1. Request Parameters

- AssignmentStatus

2.2.54.2. Response Elements

- IsTruncated
- Marker
- VirtualMFADevices.member.N
    - VirtualMFADevice
        - Base32StringSeed
        - EnableDate
        - QRCodePNG
        - SerialNumber
        - Tags.member.N
            - Tag
                - Key
                - Value
        - User
            - Arn
            - CreateDate
            - PasswordLastUsed
            - Path
            - PermissionsBoundary
                - PermissionsBoundaryArn
                - PermissionsBoundaryType
            - Tags.member.N
                - Tag
                    - Key
                    - Value
            - UserId
            - UserName

2.2.54.3. Errors

Only **common errors** are returned.

2.2.55. PutGroupPolicy

Adds or updates an inline policy document that is embedded in the specified IAM group.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **PutGroupPolicy**

2.2.55.1. Request Parameters

- GroupName
- PolicyDocument

  Note For information about HyperStore's IAM policy document support see **"Supported IAM Policy Elements"** (page 121).

- PolicyName

2.2.55.2. Errors

- LimitExceeded
- MalformedPolicyDocument
- NoSuchEntity
- ServiceFailure

2.2.56. PutRolePolicy

Adds or updates an inline policy document that is embedded in the specified IAM role.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **PutRolePolicy**

2.2.56.1. Request Parameters

- PolicyDocument
- PolicyName
- RoleName

2.2.56.2. Errors

- LimitExceeded
- MalformedPolicyDocument
- NoSuchEntity
- ServiceFailure
- UnmodifiableEntity

2.2.57. PutUserPolicy

Adds or updates an inline policy document that is embedded in the specified IAM user.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **PutUserPolicy**

2.2.57.1. Request Parameters

- PolicyDocument

> **Note** For information about HyperStore's IAM policy document support see **"Supported IAM Policy Elements"** (page 121).

- PolicyName
- UserName

2.2.57.2. Errors

- LimitExceeded
- MalformedPolicyDocument
- NoSuchEntity
- ServiceFailure

2.2.58. RemoveUserFromGroup

Removes the specified user from the specified group.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **RemoveUserFromGroup**

2.2.58.1. Request Parameters

- GroupName
- UserName

2.2.58.2. Errors

- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.59. ResyncMFADevice

Synchronizes the specified MFA device with its IAM resource object on the HyperStore servers.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **ResyncMFADevice**

2.2.59.1. Request Parameters

- AuthenticationCode1
- AuthenticationCode2

- SerialNumber
- UserName

2.2.59.2. Errors

- InvalidAuthenticationCode
- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.60. SetDefaultPolicyVersion

Sets the specified version of the specified policy as the policy's default (operative) version.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **SetDefaultPolicyVersion**

2.2.60.1. Request Parameters

- PolicyArn
- VersionId

2.2.60.2. Errors

- InvalidInput
- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.61. SimulatePrincipalPolicy

Simulate how a set of IAM policies attached to an IAM entity works with a list of API operations and AWS resources to determine the policies' effective permissions.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **SimulatePrincipalPolicy**

2.2.61.1. Request Parameters

- ActionNames.member.N

  Note Requests with invalid or unknown actions (such as *s3:PetObject*) will be rejected with a 400 InvalidInput error. Requests with a wildcard action (such as *s3:**) will also be rejected with a 400 InvalidInput error.

- MaxItems

> **Note** The HyperStore IAM Service does not support pagination of results. Therefore if the request specifies a "MaxItems" value and the simulation produces more than that many results, the request will be rejected with a 400 InvalidInput error and the error message will indicate that the "MaxItems" value must be increased or the number of action names must be decreased.

- PolicySourceArn
- ResourceArns.member.N

2.2.61.2. Response Elements

- EvaluationResults.member.N
    - EvalActionName
    - EvalDecision
    - EvalResourceName
    - ResourceSpecificResults.member.N
        - EvalResourceDecision
        - EvalResourceName
- IsTruncated

> **Note** "IsTruncated" will always be "false".

2.2.61.3. Errors

- InvalidInput
- NoSuchEntity

2.2.62. UpdateAccessKey

Changes the status of the specified access key from Active to Inactive, or vice versa.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **UpdateAccessKey**

2.2.62.1. Request Parameters

- AccessKeyId
- Status
- UserName

2.2.62.2. Errors

- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.63. UpdateAssumeRolePolicy

Updates the policy that grants an IAM entity permission to assume a role.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **UpdateAssumeRolePolicy**

2.2.63.1. Request Parameters

- PolicyDocument
- RoleName

Note For Conditions in a trust policy, HyperStore supports only the *StringEquals* condition operator and only the following condition keys:
aws:TokenIssueTime
sts:ExternalId
saml:aud
saml:doc
saml:iss
saml:namequalifier
saml:sub
saml:sub_type

2.2.63.2. Errors

- LimitExceeded
- MalformedPolicyDocument
- NoSuchEntity
- ServiceFailure
- UnmodifiableEntity

2.2.64. UpdateGroup

Updates the name and/or the path of the specified IAM group.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **UpdateGroup**

2.2.64.1. Request Parameters

- GroupName
- NewGroupName
- NewPath

2.2.64.2. Errors

- EntityAlreadyExists
- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.2.65. UpdateRole

Updates the description or maximum session duration setting of a role.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **UpdateRole**

2.2.65.1. Request Parameters

- Description
- MaxSessionDuration
- RoleName

2.2.65.2. Errors

- NoSuchEntity
- ServiceFailure
- UnmodifiableEntity

2.2.66. UpdateRoleDescription

Although HyperStore supports this Action, it is recommended to use **UpdateRole** instead.

AWS documentation: **UpdateRoleDescription**

2.2.67. UpdateSAMLProvider

Updates the metadata document for an existing SAML provider resource object.

HyperStore supports the parameters, response elements, and errors listed below.

For action details and examples see the AWS documentation: **UpdateSAMLProvider**

2.2.67.1. Request Parameters

- SAMLMetadataDocument
- SAMLProviderArn

2.2.67.2. Response Elements

- SAMLProviderArn

2.2.67.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

2.2.68. UpdateUser

Updates the name and/or the path of the specified IAM user.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **UpdateUser**

2.2.68.1. Request Parameters

- NewPath
- NewUserName
- UserName

2.2.68.2. Errors

- EntityAlreadyExists
- EntityTemporarilyUnmodifiable
- LimitExceeded
- NoSuchEntity
- ServiceFailure

2.3. Supported IAM Policy Elements

IAM policies grant permissions to IAM groups and users. You can create IAM policy documents through the CMC or by using a third party or custom IAM client. If you use the CMC, you have the choice of using a visual policy editor or using a JSON editor.

HyperStore supports AWS standard IAM policy formatting and most policy elements for granting S3 or IAM service permissions.

For guidance on how to construct IAM policies for S3 service permissions or IAM service permissions, see the AWS documentation on this topic. For example:

- *Policies and Permissions*

  http://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html

- *IAM JSON Policy Elements Reference*

  https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements.html

- *Actions, Resources, and Condition Keys for Amazon S3*

  https://docs.aws.amazon.com/IAM/latest/UserGuide/list_amazons3.html

- *Actions, Resources, and Condition Keys for Identity And Access Management*

  https://docs.aws.amazon.com/service-authorization/latest/reference/list_awsid-entityandaccessmanagement.html

Below is an example of a simple IAM policy document granting permission to list the contents of a bucket named "bucket1":

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::bucket1"
  }]
}
```

Note that:

- For "Version", you **must** use "2012-10-17"
- HyperStore supports **most but not all** of the S3 Actions and IAM Actions, and most but not all of the condition keys, cited in the AWS documentation for IAM policy formation. In general, when constructing IAM policies you can use all the Actions that correspond to operations supported by the HyperStore S3 Service and the HyperStore IAM Service. You can check the HyperStore S3 API documentation and HyperStore IAM API documentation if you are unsure whether a particular operation is supported. Alternatively you can check the CMC interface for creating an IAM policy -- the interface lists all the supported S3 actions and IAM actions.
- HyperStore does not allow IAM users to use the **SQS Service**. The SQS Service will reject requests from IAM users even if those users' IAM policies are configured to allow SQS actions.
- Actions in IAM policies are **case sensitive**, so be sure to exactly match the desired Action name as it appears in the AWS documentation.

2.4. SAML Support

The HyperStore object storage system supports Security Assertion Markup Language (SAML 2.0) based access to S3 storage resources. It does so by supporting the standard AWS IAM Service calls and Security Token Service calls that are needed to set up and execute SAML based access. With SAML, federated users -- users who have been authenticated by a trusted identity provider system (IdP) external to HyperStore -- can be granted temporary access to S3 resources, subject to policy-based permission restrictions.

At a high level, the process of setting up and using SAML access for HyperStore works as described below.

- **"Downloading the HyperStore SAML Metadata Document for IdP Setup"** (page 122)
- **"Using the IAM Service to Create SAML Provider Resources"** (page 123)
- **"Using the IAM Service to Create Roles"** (page 123)
- **"Using the STS Service to Assume a Role"** (page 124)

2.4.1. Downloading the HyperStore SAML Metadata Document for IdP Setup

For each external identity provider system (IdP) that you expect to be a source of SAML assertions submitted to HyperStore , you must load the HyperStore SAML Service Provider Metadata document into the IdP. This metadata document is specific to your HyperStore system, and provides the IdP with information about how to submit SAML assertions to HyperStore. The procedure for applying the SAML Service Provider metadata document into the IdP will depend on the IdP that you are working with (refer to your IdP's documentation), but regardless of those particulars you can download the document from the Cloudian Management Console (CMC) at this URL:

```
https://<cmc FQDN or IP>:<cmc port>/static/saml-metadata.xml
```

For example:

```
https://cmc.enterprise.com:8443/static/saml-metadata.xml
```

2.4.2. Using the IAM Service to Create SAML Provider Resources

HyperStore's IAM Service supports all the calls that you need to create and manage SAML provider resources within the IAM Service. A SAML provider resource describes an IdP that will be a source of SAML assertions submitted to HyperStore on behalf of federated users who have been authenticated by the IdP.

You can create SAML provider resources either by using the CMC's **Manage Identity Providers** page, or by using a third party IAM client to access the HyperStore IAM Service. If you are using a third party IAM client, the relevant IAM calls are:

- CreateSAMLProvider
- ListSAMLProviders
- GetSAMLProvider
- UpdateSAMLProvider
- DeleteSAMLProvider

You should create a SAML provider resource for each IdP that will be a trusted source of incoming SAML assertions.

2.4.3. Using the IAM Service to Create Roles

HyperStore's IAM Service supports all the calls that you need to create and manage IAM roles. As part of creating an IAM role you define a "trust policy" that specifies who will be allowed to assume that role. To facilitate SAML based access to HyperStore, you specify one or more SAML providers as the principal within an IAM role's trust policy, as you create the role. Once an IAM role is created, you then specify the S3 permissions granted to that role, by either attaching a managed permissions policy to the role or creating an inline permission policy specific to that role.

You can create and manage IAM roles either by using the CMC's **Manage Roles** page, or by using a third party IAM client. If you are using a third party IAM client to access the HyperStore IAM Service, the relevant IAM calls are:

- CreateRole
- ListRoles
- GetRole
- UpdateRole
- UpdateAssumRolePolicy
- DeleteRole
- AttachRolePolicy

- ListAttachedRolePolicies
- DetachRolePolicy
- PutRolePolicy
- ListRolePolicies
- GetRolePolicy
- DeleteRolePolicy

2.4.3.1. Limitations

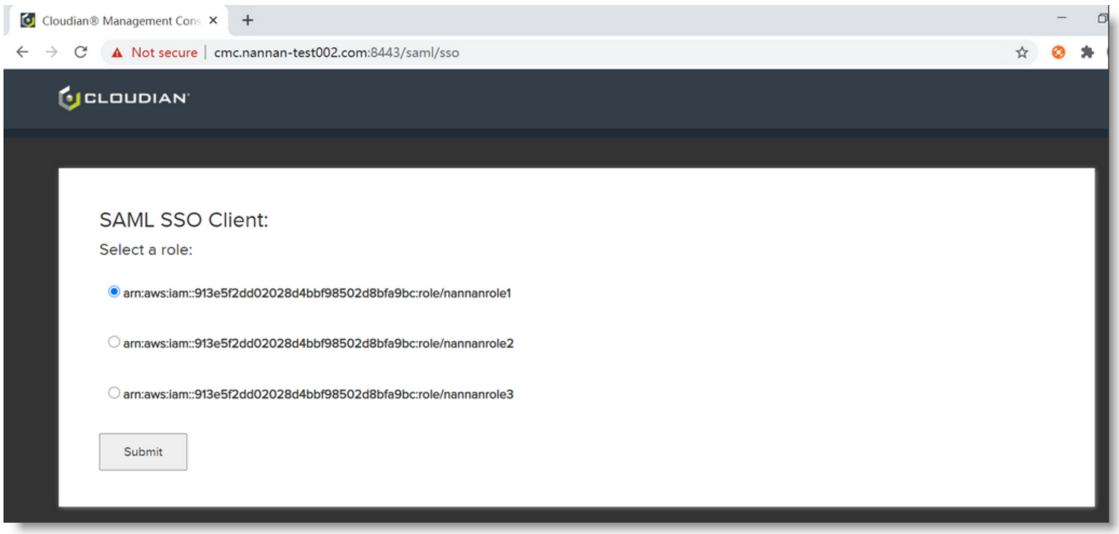
- Role session policies and role tags are not supported.
- Support for Conditions in trust policies is limited; see the *CreateRole* API call.

2.4.4. Using the STS Service to Assume a Role

Once an IdP has been configured with HyperStore's SAML metadata XML document, and you have created a SAML provider IAM resource for that IdP and specified that provider as part of an IAM role's trust policy, SAML assertions from that provider can be used to allow federated users to assume that role by calling the HyperStore Security Token Service's *AssumeRoleWithSAML* API call. This API call initiates a role session during which properly authenticated and authorized users are provided with temporary S3 security credentials.

There are two options for using this API call to assume a role:

- A third party STS client application can be used to submit an *AssumeRoleWithSAML* API call to HyperStore's STS Service.
- The CMC hosts a single-sign-on (SSO) page to which an IdP can submit a SAML assertion on behalf of a user who has successfully logged into the IdP. The IdP can submit an HTTP POST to the **Location** URL identified within the **AssertionConsumerService** attribute of the HyperStore SAML Metadata document. Based on the submitted SAML assertion contents, the CMC will display a list of Roles for which the user identified in the assertion is eligible. The user can select a Role from that list, and the CMC will then submit an *AssumeRoleWithSAML* request for that Role to the HyperStore STS Service. The CMC then makes the returned temporary security credentials available for the user to copy to their clipboard, and the user can then paste the temporary credentials into an S3 application and perform the S3 operations permitted by the Role.


Cloudian® Management Cons
Not secure
cmc.nannan-test002.com:8443/saml/sso
CLOUDIAN

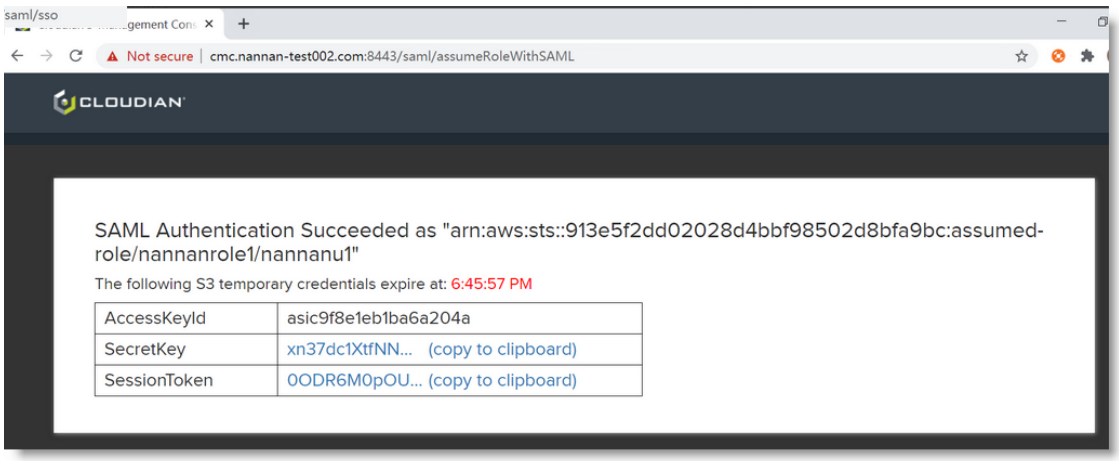
SAML SSO Client:

Select a role:

● arn:aws:iam::913e5f2dd02028d4bbf98502d8bfa9bc:role/nannanrole1

○ arn:aws:iam::913e5f2dd02028d4bbf98502d8bfa9bc:role/nannanrole2

○ arn:aws:iam::913e5f2dd02028d4bbf98502d8bfa9bc:role/nannanrole3



SAML Authentication Succeeded as "arn:aws:sts::913e5f2dd02028d4bbf98502d8bfa9bc:assumed-role/nannanrole1/nannanu1"

The following S3 temporary credentials expire at: 6:45:57 PM

AccessKeyId	asic9f8e1eb1ba6a204a
SecretKey	xn37dc1XtfNN... (copy to clipboard)
SessionToken	0ODR6M0pOU... (copy to clipboard)

2.4.4.1. Limitations

Users with temporary security credentials obtained from the HyperStore STS Service:

- Cannot log into the CMC
- Cannot access the IAM Service (even with a third party client application)

What users with temporary security credentials **can** do is **access the HyperStore S3 Service by using a third party S3 client application**. Their S3 permissions will be limited to those permissions ascribed to the role that they have assumed.

This page left intentionally blank

Chapter 3. STS API

3.1. Introduction

3.1.1. HyperStore Support for the AWS STS API

To facilitate Security Assertion Markup Language (SAML) based access to HyperStore S3 services, Hyper-Store provides **limited support** for the Amazon Web Services Security Token Service (STS) API:

- Only a few STS Actions are supported -- for details see "Supported STS Actions".
- HyperStore does not allow users with temporary security credentials (obtained through the STS Service) to perform **IAM operations**. The HyperStore IAM Service will reject requests that contain temporary credentials. Users with temporary credentials can only access the S3 Service (within the permission restrictions of the roles that such users assume).
- Users with temporary security credentials are not allowed to log into the CMC.

The default STS Service endpoint URLs for HTTP and HTTPS are:

- *http://sts.<your-organization-domain>:16080*
- *https://sts.<your-organization-domain>:16443*

Note The STS Service uses the same listening ports as the IAM Service.

To access the STS Service, HyperStore users **need S3 access credentials**. When users are created in Hyper-Store, S3 access credentials are automatically created for the users.

If users are using a third party STS client to access the HyperStore STS Service, the users can obtain their S3 access credentials by logging in to the CMC and going to the **Security Credentials** page (via the drop-down menu under the user login name). They can then supply those credentials to the third party STS client application.

3.1.1.1. STS API Notes for HyperStore Administrators

DNS

Be sure to configure the STS endpoint domain in your DNS environment. For more information see the "DNS Set-Up" section of the ***Cloudian HyperStore Administrator's Guide***.

3.1.2. STS Common Request Parameters

From the **"Common Parameters" section** of the AWS STS API specification, HyperStore supports the parameters listed below. If a common parameter from that specification section is not listed below, HyperStore does not support it.

- Action
- Version
- X-Amz-Algorithm

- X-Amz-Credential
- X-Amz-Date
- X-Amz-Security-Token (only supported for **GetCallerIdentity** requests)
- X-Amz-Signature
- X-Amz-SignedHeaders

3.1.3. STS Common Errors

From the **"Common Errors" section** of the AWS STS API specification, HyperStore supports the parameters listed below. If a common parameter from that specification section is not listed below, HyperStore does not support it.

- AccessDeniedException
- InternalFailure
- InvalidAction
- InvalidClientTokenId
- InvalidParameterCombination
- InvalidParameterValue
- MissingAuthenticationToken
- MissingParameter
- ServiceUnavailable
- ValidationError

3.2. Supported STS Actions

The HyperStore implementation of the AWS STS API supports the Actions listed in this section. If an STS Action is not listed in this section, HyperStore does not support it. For each Action, the documentation here lists the request parameters and request or response elements that HyperStore supports. If a parameter or element is not listed in this documentation, HyperStore does not support it.

For detailed descriptions of each Action and its associated parameters and elements, see the AWS documentation links.

3.2.1. AssumeRole

Returns a set of temporary security credentials that you can use to access S3 resources that you might not normally have access to.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **AssumeRole**

Note HyperStore does not allow users with temporary security credentials to perform **IAM operations**.

3.2.1.1. Request Parameters

- DurationSeconds
- ExternalId
- RoleArn
- RoleSessionName

3.2.1.2. Response Elements

- AssumedRoleUser
- Credentials

3.2.1.3. Errors

- InvalidParameterValue
- NoSuchEntity

3.2.2. AssumeRoleWithSAML

Returns a set of temporary security credentials for users who have been authenticated via a SAML authentication response.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **AssumeRoleWithSAML**

Note For an overview of HyperStore support for SAML see **"SAML Support"** (page 122). For more information about using the STS *AssumeRoleWithSAML* call with HyperStore see **"Using the STS Service to Assume a Role"** (page 124).

3.2.2.1. Request Parameters

- DurationSeconds
- PrincipalArn
- RoleArn
- SAMLAssertion

3.2.2.2. Response Elements

- AssumedRoleUser
- Audience
- Credentials
- Issuer
- NameQualifier

- Subject
- SubjectType

3.2.2.3. Errors

- ExpiredToken
- InvalidClientTokenId
- InvalidParameterValue

3.2.3. GetCallerIdentity

Returns details about the IAM user or role whose credentials are used to call the operation.

HyperStore supports the elements listed below.

For action details and examples see the AWS documentation: **GetCallerIdentity**

3.2.3.1. Response Elements

- Account
- Arn
- UserId

3.2.3.2. Errors

- ExpiredToken
- InvalidClientTokenId

Chapter 4. SQS API

4.1. Introduction

4.1.1. HyperStore Support for the AWS SQS API

In support of the bucket notifications feature, HyperStore provides **limited support** for the Amazon Web Services Simple Queue Service (SQS) API. The queueing and processing of messages is implemented within the HyperStore system. Bucket owners can use the S3 API operation **"PutBucketNotificationConfiguration"** (page 64) to configure bucket notifications so that when specified S3 operations occur within the bucket -- such as objects being uploaded to the bucket or deleted from the bucket -- HyperStore publishes a notification message to a specified SQS queue.

In the current HyperStore release, there are these limitations to the bucket notifications feature and the SQS Service:

- A third party SQS client application must be used to interface with the HyperStore SQS Service to perform operations such as creating and configuring queues and receiving and deleting queued messages. The Cloudian Management Console (CMC) does not support SQS operations.
- A third party S3 client application must be used to execute the **"PutBucketNotificationConfiguration"** (page 64) operation. The CMC does not support this S3 operation.
- For bucket notifications to an SQS queue to work, the bucket owner must also be the owner of the SQS queue.
- The SQS Service does not support HTTPS access. Only regular HTTP access is supported.
- IAM users cannot use the SQS Service. Only HyperStore account root users can use the SQS Service.
- The HyperStore SQS Service supports many of the Actions from the Amazon SQS API, but not all of them. For more detail see **"SQS Supported Actions"** (page 132).

The SQS Service and the bucket notifications feature are **disabled by default** (HyperStore administrators see **"Enabling the SQS Service and the Bucket Notification Feature"** (page 132)). When the SQS Service and the bucket notifications feature are enabled then:

- A third party SQS client application can be used to submit requests to the HyperStore SQS Service, such as for creating and configuring a queue. For HyperStore support of SQS Actions see **"SQS Supported Actions"** (page 132). The default SQS Service endpoint URL including the port number is ***http://s3-sqs.<your-organization-domain>:18090***
- A third party S3 client application can be used to submit a *PutBucketNotificationConfiguration* request to the HyperStore S3 Service, to configure notifications for an existing bucket. For HyperStore support of this S3 API method see **"PutBucketNotificationConfiguration"** (page 64). As noted previously, the **bucket owner must also be the SQS queue owner**.

To access the SQS Service, HyperStore users **need S3 access credentials**. When users are created in HyperStore, S3 access credentials are automatically created for the users.

If users are using a third party SQS client to access the HyperStore SQS Service, the users can obtain their S3 access credentials by logging in to the CMC and going to the **Security Credentials** page (via the drop-down menu under the user login name). They can then supply those credentials to the third party SQS client application.

4.1.1.1. SQS API Notes for HyperStore Administrators

- **"Enabling the SQS Service and the Bucket Notification Feature"** (page 132)
- **"DNS"** (page 132)
- **"SQS Request Logging"** (page 132)

Enabling the SQS Service and the Bucket Notification Feature

HyperStore's SQS Service and its bucket notifications feature are **disabled by default**. To enable the SQS Service and the bucket notifications feature do the following:

1. In *common.csv*:
   - Set *sqs_enabled* to *true* (by default it is *false*)
   - Set *sqs_endpoint* to an SQS service endpoint for your domain (the recommended endpoint format is *s3-sqs.<organization-domain>*)
   - Optionally set *sqs_port*, if you want an SQS listening port other than the default which is 18090
2. In *mts.properties.erb*:
   - Edit the *cloudian.s3.unsupported* property to remove *notification* from the comma-separated list of unsupported S3 request types.
   - Below the *cloudian.s3.unsupported* property, **add this new property** to the file (it is not in the file by default):

     ```
     cloudian.s3.bucketnotification=true
     ```
3. In the installation staging directory (*/opt/cloudian-staging/7.5.2*) launch the installer:

   ```
   ./cloudianInstall.sh
   ```
4. Enter **2** for Cluster Management then **a** for Review Cluster Configuration, then enter **yes** when asked if you want to update the Puppet server with your changes.
5. Use the installer to push the configuration changes and restart the S3 Service and the SQS Service.

DNS

Be sure to configure the SQS endpoint domain in your DNS environment. For more information see the "DNS Set-Up" section of the ***Cloudian HyperStore Administrator's Guide***.

SQS Request Logging

Information about requests processed by the SQS Service are logged to *cloudian-sqs-request.log*, which exists on each node.For more information see the "Logging" section of the ***Cloudian HyperStore Administrator's Guide***.

4.2. SQS Supported Actions

The HyperStore implementation of the AWS SQS API supports the Actions listed in this section. If an SQS Action is not listed in this section, HyperStore does not support it. For each Action, the documentation here lists the request parameters and request or response elements that HyperStore supports. If a parameter or element is not listed in this documentation, HyperStore does not support it.

For detailed descriptions of each Action and its associated parameters and elements, see the AWS documentation links.

Note The HyperStore SQS Service is disabled by default. System administrators can enable the service.

4.2.1. ChangeMessageVisibility

Changes the visibility timeout of a specified message in a queue to a new value.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **ChangeMessageVisibility**

4.2.1.1. Request Parameters

- QueueUrl
- ReceiptHandle
- VisibilityTimeout

4.2.1.2. Errors

- AWS.SimpleQueueService.MessageNotInflight
- ReceiptHandleIsInvalid

4.2.2. CreateQueue

Creates a new standard queue.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **CreateQueue**

Note HyperStore currently only supports Standard queues. HyperStore does not support FIFO queues.

4.2.2.1. Request Parameters

- Attribute

  HyperStore currently only supports these queue attributes:

  - DelaySeconds
  - MaximumMessageSize
  - MessageRetentionPeriod
  - ReceiveMessageWaitTimeSeconds
  - VisibilityTimeout

  Any other attributes included in the CreateQueue request will be ignored.

- QueueName
- Tag

4.2.2.2. Response Elements

- QueueUrl

4.2.2.3. Errors

- AWS.SimpleQueueService.QueueDeletedRecently
- QueueAlreadyExists

4.2.3. DeleteMessage

Deletes the specified message from the specified queue.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **DeleteMessage**

4.2.3.1. Request Parameters

- QueueUrl
- ReceiptHandle

4.2.3.2. Errors

- InvalidIdFormat
- ReceiptHandleIsInvalid

4.2.4. DeleteQueue

Deletes the queue specified by the QueueUrl, regardless of the queue's contents.

HyperStore supports the parameters listed below.

For action details and examples see the AWS documentation: **DeleteQueue**

4.2.4.1. Request Parameters

- QueueUrl

4.2.5. GetQueueAttributes

Gets attributes for the specified queue.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **GetQueueAttributes**

4.2.5.1. Request Parameters

- AttributeName.N

  Note For the list of queue attributes that HyperStore supports, see **"CreateQueue"** (page 133).

- QueueUrl

4.2.5.2. Response Elements

- Attribute

4.2.5.3. Errors

- InvalidAttributeName

4.2.6. GetQueueUrl

Returns the URL of an existing Amazon SQS queue.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **GetQueueUrl**

4.2.6.1. Request Parameters

- QueueName
- QueueOwnerAWSAccountId

4.2.6.2. Response Elements

- QueueUrl

4.2.6.3. Errors

- AWS.SimpleQueueService.NonExistentQueue

4.2.7. ListQueues

Returns a list of your queues in the current region.

HyperStore supports the parameters and elements listed below.

For action details and examples see the AWS documentation: **ListQueues**

4.2.7.1. Request Parameters

- MaxResults
- NextToken

- QueueNamePrefix

4.2.7.2. Response Elements

- NextToken
- QueueUrl.N

4.2.8. PurgeQueue

Deletes the messages in a queue specified by the QueueURL parameter.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **PurgeQueue**

4.2.8.1. Request Parameters

- QueueUrl

4.2.8.2. Errors

- AWS.SimpleQueueService.NonExistentQueue
- AWS.SimpleQueueService.PurgeQueueInProgress

4.2.9. ReceiveMessage

Retrieves one or more messages (up to 10), from the specified queue.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **ReceiveMessage**

4.2.9.1. Request Parameters

- MaxNumberOfMessages
- QueueUrl
- ReceiveRequestAttemptId
- VisibilityTimeout
- WaitTimeSeconds

Note HyperStore does not currently support message attributes.

4.2.9.2. Response Elements

- Message.N

4.2.9.3. Errors

- OverLimit

4.2.10. SendMessage

Delivers a message to the specified queue.

HyperStore supports the parameters, elements, and errors listed below.

For action details and examples see the AWS documentation: **SendMessage**

Note The SendMessage action is not intended to be used by external SQS clients. The HyperStore S3 Service internally uses the SendMessage action to publish notification messages to a queue.

4.2.10.1. Request Parameters

- DelaySeconds
- MessageBody
- QueueUrl

Note HyperStore does not currently support message attributes.

4.2.10.2. Response Elements

- MD5OfMessageBody
- MessageId

4.2.10.3. Errors

- AWS.SimpleQueueService.UnsupportedOperation
- InvalidMessageContents

4.2.11. SetQueueAttributes

Sets the value of one or more queue attributes.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: **SetQueueAttributes**

4.2.11.1. Request Parameters

- Attribute

> **Note** For the list of queue attributes that HyperStore supports see **"CreateQueue"** (page 133).

- QueueUrl

4.2.11.2. Errors

- InvalidAttributeName