

Clouodian HyperStore Administration Guide

Version 8.2

This page left intentionally blank.

Confidentiality Notice

The information contained in this document is confidential to, and is the intellectual property of, Cloudian, Inc. Neither this document nor any information contained herein may be (1) used in any manner other than to support the use of Cloudian software in accordance with a valid license obtained from Cloudian, Inc, or (2) reproduced, disclosed or otherwise provided to others under any circumstances, without the prior written permission of Cloudian, Inc. Without limiting the foregoing, use of any information contained in this document in connection with the development of a product or service that may be competitive with Cloudian software is strictly prohibited. Any permitted reproduction of this document or any portion hereof must be accompanied by this legend.

This page left intentionally blank.

Contents

What's New in HyperStore Version 8.x	19
HyperStore 8.2	19
APIs	19
System Management and Functionality	21
HyperStore 8.1.1	26
APIs	26
System Management and Functionality	27
HyperStore 8.1	28
APIs	28
System Management and Functionality	28
HyperStore 8.0	31
APIs	31
System Management and Functionality	31
Chapter 1. Introduction to HyperStore	33
1.1. HyperStore Documentation	33
1.2. HyperStore Overview	35
1.3. Licensing and Auditing	36
1.3.1. License Expiration	37
1.3.2. Licensed Maximum On-Premise Storage Usage	38
1.3.3. Licensed Maximum Tiered Storage Usage	39
1.3.4. Object Lock (WORM) License	41
1.3.5. HyperIQ License	41
1.3.6. License Updating	42
1.3.7. Auditing	42
1.4. Nodes, Data Centers, and Regions	42
1.4.1. Storage Policies in Multi-DC or Multi-Region Systems	43
1.4.2. Service Architecture in Multi-DC or Multi-Region Systems	43
1.4.3. Deploying HyperStore to Multiple DCs or Regions	44
1.4.4. Using the CMC with Multiple DCs or Regions	45
1.4.5. Using the Admin API in a Multi-Region System	46

1.5. Hosts and Network	46
1.5.1. Host Hardware and Operating System	46
1.5.2. File System Requirements	49
1.5.3. DNS Set-Up	53
1.5.4. Load Balancing	56
1.5.5. HyperStore Listening Ports	57
1.5.6. Outbound Internet Access	60
1.6. HyperStore Services	61
1.6.1. HyperStore Services Overview	61
1.6.2. Cloudian Management Console (CMC) Service	63
1.6.3. API Services	64
1.6.4. Database Services	64
1.6.5. HyperStore Service and the HSFS	66
1.6.6. Supporting Services	69
1.6.7. Auxiliary Services	70
1.7. System Diagrams	70
1.7.1. System Levels	70
1.7.2. Service Interconnections	71
1.7.3. Services Distribution -- 3 Nodes, Single DC	72
1.7.4. Services Distribution -- Multi-DC, Single Region	73
1.7.5. Services Distribution -- Multi-Region	74
1.7.6. Specialized Services Availability	75
1.7.7. S3 PUT Processing Flow	76
1.7.8. S3 GET Processing Flow	78
1.7.9. Strong Read-After-Write Consistency	79
1.7.10. Dynamic Consistency Levels	80
1.7.11. How vNodes Work	83
Chapter 2. Starting, Stopping, and Checking Status of Services	91
2.1. Using hscctl to Start, Stop, or Check Status of Services (Scope: Node, Data Center, Region, or System)	91
2.2. Using the CMC to Start, Stop, or Check Status of Services (Scope: Node)	93

2.3. Using systemctl to Start, Stop, or Check Status of Services (Scope: Node)	94
2.4. Shutting Down or Rebooting a Node	95
2.5. Automatic Service Start on Node Boot-Up or After a Service Crash	95
Chapter 3. Upgrading Your HyperStore Software Version	97
3.1. Preparing to Upgrade Your System	97
3.1.1. Additional Upgrade Preparation If Your System Currently Has Failed Disks	98
3.2. Upgrading Your System	99
3.2.1. Upgrade Errors	100
3.3. Actions to Complete After Upgrade	101
3.4. Installing a HyperStore Patch	102
3.5. Upgrading the HyperStore File Service	102
Chapter 4. Setting Up Administration Features	105
4.1. HyperStore Shell (HSH)	105
4.1.1. HyperStore Shell (HSH) Feature Overview	105
4.1.2. Enabling the HSH and Managing HSH Users	106
4.1.3. Using the HSH	111
4.2. Smart Support	117
4.2.1. Smart Support and Diagnostics Feature Overview	117
4.2.2. Configuring Smart Support and Node Diagnostics	120
4.2.3. Maintaining Personal Data Privacy While Using Smart Support and Node Diagnostics	121
4.2.4. Executing Node Diagnostics Collection	123
4.2.5. Cloudian Support Tunnel	123
4.3. Automated Disk Usage Management	124
4.3.1. Automated Disk Usage Management Feature Overview	124
4.3.2. Configuring Disk Usage Balancing	127
4.3.3. Triggering a Disk Usage Balance Check	128
4.3.4. Checking Disk Usage and Health Status	128
4.4. Automated Disk Failure Management	128
4.4.1. Automated Disk Failure Management Feature Overview	128
4.4.2. Disk Error Alerts	129
4.4.3. Responding to a Disabled Disk	129
4.4.4. Checking Disk Usage and Health Status	130

4.5. Automated Data Repair	130
4.5.1. Automated Data Repair Feature Overview	130
4.5.2. Configuring Automatic Data Repair	133
4.5.3. Checking Data Repair Status	133
4.5.4. Disabling or Stopping Data Repairs	134
4.6. Setting Up Alerts and Notifications	137
4.7. Creating Additional System Admin Users	137
4.8. Forwarding Log Messages to a Central Log Server	138
Chapter 5. Setting Up Service Features	139
5.1. hsDNS	139
5.1.1. hsDNS Feature Overview	139
5.1.2. Setting Up hsDNS	142
5.2. GPUDirect Storage	143
5.2.1. GPUDirect Storage Feature Overview	143
5.2.2. GDS Setup	146
5.3. Storage Policies	148
5.3.1. Storage Policies Feature Overview	148
5.3.2. Consistency Levels	152
5.3.3. Metadata Replication	153
5.3.4. Creating and Managing Storage Policies	155
5.3.5. Finding an Object's Replicas or EC Fragments	156
5.3.6. Storage Policy Resilience to Downed Nodes	156
5.4. Security and Privacy Features	177
5.4.1. HTTPS (SSL/TLS)	177
5.4.2. Server-Side Encryption	186
5.4.3. Setting Up User Password and Credential Controls	199
5.4.4. Enabling Secure Delete	202
5.4.5. FIPS Support	203
5.5. Auto-Tiering	203
5.5.1. Auto-Tiering Feature Overview	203
5.5.2. Preparing the Auto-Tiering Feature	209

5.5.3. Setting Up Auto-Tiering for a Bucket	210
5.5.4. Accessing Auto-Tiered Objects	211
5.6. Cross-Region Replication	213
5.6.1. Cross-Region Replication Feature Overview	213
5.6.2. Preparing the Cross-Region Replication Feature	218
5.6.3. Setting Up Cross-Region Replication for a Bucket	219
5.7. Object Lock	221
5.7.1. Object Lock Feature Overview	221
5.7.2. Preparing the Object Lock Feature	224
5.7.3. Setting Up Object Lock for a Bucket	224
5.7.4. Protections for Locked Objects	229
5.8. Object Metadata	230
5.8.1. Object Metadata and Search Feature Overview	230
5.8.2. Creating and Retrieving User-Defined Object Metadata and Tags	234
5.8.3. Preparing the Metadata Search Feature	236
5.9. File Services (SMB/NFS)	238
5.9.1. File Services Feature Overview	238
5.9.2. Preparing the File Services Feature	241
5.10. Quality of Service Controls	246
5.10.1. Quality of Service (QoS) Feature Overview	246
5.10.2. Preparing the QoS Feature	248
5.10.3. Applying QoS Controls to Users	250
5.11. Usage Reporting	252
5.11.1. Usage Reporting Feature Overview	252
5.11.2. Preparing the Usage Reporting Feature	255
5.11.3. Generating a Usage Report	256
5.11.4. Validating Storage Usage Data	257
5.12. Billing	258
5.12.1. Billing Feature Overview	258
5.12.2. Preparing the Billing Feature	259

5.12.3. Applying Billing to Users	262
5.13. Customizing the CMC	264
5.13.1. Showing/Hiding CMC UI Functions	264
5.13.2. Rebranding the CMC UI	266
5.13.3. Configuring a Login Page Acknowledgment Gate	270
5.13.4. Implementing Single Sign-On for the CMC	271
5.14. Provisioning Groups and Users	275
5.14.1. Group and User Provisioning Feature Overview	275
5.14.2. Provisioning Groups	276
5.14.3. Provisioning Users	276
5.14.4. LDAP Integration	278
Chapter 6. Cloudbian Management Console (CMC)	283
6.1. Cloudbian Management Console (CMC)	283
6.2. Dashboard	285
6.3. Analytics	294
6.3.1. Cluster Usage	294
6.3.2. Capacity Explorer	297
6.3.3. Usage By Users & Groups	299
6.3.4. Object Locator	305
6.4. Buckets	306
6.4.1. Add a Bucket	307
6.4.2. Set Bucket Properties	309
6.4.3. Delete a Bucket	343
6.5. Objects	344
6.5.1. Create or Delete a "Folder"	345
6.5.2. Upload an Object	346
6.5.3. Set Object Properties	348
6.5.4. List or Search for Objects	357
6.5.5. Download an Object	359
6.5.6. Retrieve a Tiered Object	360
6.5.7. Delete an Object	363

6.5.8. Object Naming Restrictions	364
6.6. File Share	365
6.6.1. Set Global Properties for File Shares	365
6.6.2. Add a File Share	368
6.6.3. Set File Share Custom Properties	373
6.6.4. Open a File Share and Provide Its Endpoint To Users	378
6.6.5. View File Share Usage Statistics	382
6.6.6. Edit or Delete a File Share	383
6.6.7. File Services Configuration Setting Matrix	384
6.6.8. FileServiceSettings	387
6.7. Search	396
6.8. Users & Groups	400
6.8.1. Manage Users	400
6.8.2. Manage Groups	415
6.8.3. Rating Plan	423
6.8.4. Account Activity	427
6.8.5. Allowlist	428
6.8.6. Set Quality of Service (QoS) Limits	430
6.9. IAM	435
6.9.1. Manage IAM User	435
6.9.2. Manage IAM Group	442
6.9.3. Manage IAM Role	448
6.9.4. Manage IAM Policy	454
6.9.5. Manage Identity Providers	462
6.9.6. Manage MFA Devices	464
6.10. Cluster	466
6.10.1. Data Centers	466
6.10.2. Node Status	471
6.10.3. Node Activity	480
6.10.4. Node Advanced	483

6.10.5. Cluster Information	490
6.10.6. Configuration Settings	498
6.10.7. File Service Settings	510
6.10.8. Storage Policies	512
6.10.9. KMIP Profiles	545
6.10.10. Repair Status	549
6.10.11. Operation Status	554
6.11. Alerts	555
6.11.1. Alerts	555
6.11.2. Alert Rules	560
6.11.3. How HyperStore Implements Alerts	570
6.12. My Account	571
6.12.1. Profile	571
6.12.2. Security Credentials	572
Chapter 7. Cluster and Node Operations	579
7.1. Adding Nodes	579
7.1.1. Special Requirements if an Existing Node is Down	579
7.1.2. Preparing to Add Nodes	579
7.1.3. Adding Nodes	582
7.2. Adding a Data Center	590
7.2.1. Special Requirements if an Existing Node is Down or Unreachable	590
7.2.2. Preparing to Add a Data Center	590
7.2.3. Adding a Data Center	592
7.3. Adding a Region	597
7.3.1. Preparing to Add a Region	597
7.3.2. Adding a Region	599
7.4. Removing a Node	603
7.4.1. Preparing to Remove a Node	603
7.4.2. Removing a Node	607
7.5. Replacing a Node	612
7.6. Changing a Node's IP Address	612

7.7. Restoring a Node That Has Been Offline	612
7.7.1. Restoring a Node That Has Been Down for Less Than Four Hours	612
7.7.2. Restoring a Node That Has Been Down for More Than Four Hours	614
7.7.3. Restoring a Node That Has Been Down for More Than a Few Days	614
7.8. Putting a Node or DC into Maintenance	614
7.8.1. Putting a Node into Maintenance	614
7.8.2. Putting a Data Center into Maintenance	614
7.9. Changing Node Role Assignments	616
7.9.1. Change Node Role Assignments	616
7.9.2. Move or Add a Credentials DB Slave or QoS DB Slave	617
7.9.3. Move the Cassandra Seeds Role	618
7.9.4. Reduce or Change the List of CMC Hosts	620
7.9.5. Move the Cron Job Primary or Backup Role	621
7.9.6. Change Internal NTP Servers or External NTP Servers	622
7.9.7. Move the Config Controller Primary or Backup Role	624
7.9.8. Move the Credentials DB Master Role or QoS DB Master Role	627
7.9.9. Move the Redis Monitor Primary or Backup Role	631
7.10. Cron Jobs and Automated System Maintenance	632
7.10.1. System cron Jobs	632
7.10.2. Cassandra Data Compaction	636
Chapter 8. Disk Operations	639
8.1. Disabling a HyperStore Data Disk	639
8.1.1. The Impact of Disabling a Disk	639
8.1.2. Disabling a Disk	639
8.2. Enabling a HyperStore Data Disk	640
8.2.1. The Impact of Enabling a Disk	641
8.2.2. Enabling a Disabled Disk	641
8.3. Replacing a HyperStore Data Disk	642
8.3.1. The Impact of Replacing a HyperStore Data Disk	643
8.3.2. Replacing a HyperStore Data Disk	643

8.4. Replacing a Cassandra Disk	645
8.5. Responding to Data Disks Nearing Capacity	645
8.6. Responding to Cassandra Disks Nearing Capacity	646
8.7. Adding More Disks To a Node is Not Supported	646
Chapter 9. File Service Operations	647
9.1. Replacing a Node in a Three-Node File Service Pool	647
9.2. Backing Up and Restoring a Single-Node File Service Pool	649
9.2.1. Creating an On-Demand Backup	649
9.2.2. Listing Available Backups	649
9.2.3. Rolling Back an Existing Node to a Previous State	650
9.2.4. Restoring a Replacement Node	650
Chapter 10. Search Service Operations	653
10.1. Check Search Service Status	653
10.2. Manage Search Indexes	653
10.3. Manage Search Snapshots	653
Chapter 11. Monitoring	655
11.1. Clouddian HyperIQ	655
11.2. Using the CMC to Monitor HyperStore	655
11.3. Capacity Monitoring and Expansion	656
11.3.1. Monitoring Cluster Storage Capacity Utilization	657
11.3.2. Adding Nodes to Your System	658
11.4. Using the Admin API to Monitor HyperStore	659
11.5. Using Linux Utilities to Monitor System Resource Usage	659
11.6. Using JMX to Monitor HyperStore Services	659
11.7. Checking HTTP(S) Responsiveness	664
11.7.1. Clouddian Management Console (CMC)	665
Chapter 12. Logging	667
12.1. HyperStore Logs	667
12.1.1. Admin Service Logs	667
12.1.2. Cassandra (Metadata DB) Logs	668
12.1.3. CMC Logs	669
12.1.4. HyperStore Firewall Log	671
12.1.5. HyperStore Service Logs	671
12.1.6. HyperStore Shell (HSH) Log	676

12.1.7. IAM Service Logs	677
12.1.8. Monitoring Agent and Collector Logs	679
12.1.9. Phone Home (Smart Support) Log	680
12.1.10. Redis (Credentials DB and QoS DB) and Redis Monitor Logs	680
12.1.11. S3 Service Logs (including Auto-Tiering, CRR, and WORM)	682
12.1.12. Search Service Log	687
12.1.13. SQS Service Logs	687
12.2. Log Rolling Settings	689
12.3. Forwarding Log Messages to a Central Log Server	691
12.4. Using the HSH to View Logs	691
Chapter 13. Configuration	693
13.1. Using hscctl to Control HyperStore Configuration	693
13.1.1. Checking Current Configuration Values	694
13.1.2. Changing the Value of a Setting and Applying Your Change	694
13.1.3. Additional Options When Using hscctl for Configuration Control	696
13.2. HyperStore Configuration Settings	700
13.2.1. Admin Service Settings	700
13.2.2. Metadata DB Service Settings	708
13.2.3. CMC Service Settings	715
13.2.4. Common Settings	741
13.2.5. HyperStore Shell Settings	760
13.2.6. HyperStore Service Settings	761
13.2.7. IAM Service Settings	789
13.2.8. Java Settings	797
13.2.9. Jobs Settings	797
13.2.10. LB (hsDNS) Settings	798
13.2.11. Logging Settings	803
13.2.12. Monitoring and Notification Settings	804
13.2.13. Redis Monitor Settings	810
13.2.14. S3 Service Settings	811
13.2.15. SQS Service Settings	841

13.3. CMC's Configuration Settings Page	846
13.4. Installer Advanced Configuration Options	847
13.5. Configuration Special Topics	850
13.5.1. HyperStore Firewall	850
13.5.2. Anti-Virus Software	856
13.5.3. NTP Automatic Set-Up	857
13.5.4. Changing S3, Admin, or CMC Listening Ports	858
13.5.5. Changing S3, Admin, CMC, or IAM/STS Service Endpoints	859
13.5.6. Tuning HyperStore Performance Parameters	860
Chapter 14. Command Line Tools	863
14.1. hsstool	863
14.1.1. hsstool cleanup	864
14.1.2. hsstool cleanupec	869
14.1.3. hsstool info	873
14.1.4. hsstool ls	876
14.1.5. hsstool metadata	878
14.1.6. hsstool opctl	881
14.1.7. hsstool opstatus	882
14.1.8. hsstool proactiverepairq	899
14.1.9. hsstool rebalance	904
14.1.10. hsstool repair	907
14.1.11. hsstool repaircassandra	915
14.1.12. hsstool repairec	918
14.1.13. hsstool repairqueue	925
14.1.14. hsstool ring	930
14.1.15. hsstool status	932
14.1.16. hsstool trmap	933
14.1.17. hsstool whereis	938
14.2. hsctl	943
14.3. cloudianInstall.sh	944
14.3.1. Command Line Options When Using cloudianInstall.sh for HyperStore Cluster Installation	944

14.4. system_setup.sh	947
14.5. Redis Monitor Commands	947
14.5.1. get cluster	948
14.5.2. get master	949
14.5.3. get nodes	949
14.5.4. get clients	950
14.5.5. enable monitoring	950
14.5.6. disable monitoring	951
14.5.7. enable notifications	951
14.5.8. disable notifications	951
14.5.9. set master	952
14.5.10. add node	953
14.5.11. add client	953
14.5.12. test dc partition	954
14.5.13. test split brain	954
14.5.14. disable dc partition monitoring	955
14.5.15. enable dc partition monitoring	956
14.5.16. disable split brain monitoring	957
14.5.17. enable split brain monitoring	957
14.5.18. resolve split brain	958

This page left intentionally blank.

What's New in HyperStore Version 8.x

This section introduces the **main new features and enhancements** in the Clodian HyperStore 8.x releases. The current release is HyperStore 8.2.

Note For more granular release details including bug fixes and configuration setting changes please see the release notes.

- ["HyperStore 8.2"](#) (page 19)
- ["HyperStore 8.1.1"](#) (page 26)
- ["HyperStore 8.1"](#) (page 28)
- ["HyperStore 8.0"](#) (page 31)

HyperStore 8.2

APIs

Support for S3 Select

HyperStore's S3 Service now supports "S3 Select", whereby a SQL SELECT command can be used within an S3 call to filter the content of a stored S3 object that is in CSV, JSON, or Parquet format.

Note The CMC does not currently support S3 Select. To use this HyperStore S3 Service feature you must use a third party S3 client that supports the "SelectObjectContent" call.

More information:

- "SelectObjectContent" in the ***Clodian HyperStore AWS APIs Support Reference***.

S3 GetBucketEncryption call can now reflect bucket's storage policy

Optionally, you can now have the S3 *GetBucketEncryption* call reflect not only whether server-side encryption has been explicitly set for a bucket (by use of the S3 *PutBucketEncryption* call), but also whether the bucket has inherited server-side encryption from its storage policy. By default the S3 *GetBucketEncryption* call reflects only whether server-side encryption has been explicitly set for a bucket.

More information:

- ["s3.getBucketEncryption.allowPolicyResponse"](#) (page 819)
- "GetBucketEncryption" in the ***Clodian HyperStore AWS APIs Support Reference***.

Support for "s3:x-amz-server-side-encryption-customer-algorithm" as condition key in bucket policies

In the HyperStore S3 Service's implementation of the S3 *PutBucketPolicy* call, the list of supported Condition Keys has been expanded to include "s3:x-amz-server-side-encryption-customer-algorithm".

More information:

- "PutBucketPolicy" in the **Cloudian HyperStore AWS APIs Support Reference**.

Support for new checksum algorithm CRC-64NVME in applicable S3 API calls

HyperStore's implementation of the S3 API calls that have supported *x-amz-checksum-** request headers -- such as the *x-amz-checksum-crc32* and *x-amz-checksum-crc32c* headers -- now also support the new algorithm that AWS S3 recently started supporting, CRC-64NVME. The header name when using this checksum algorithm is *x-amz-checksum-crc64nvme*. Correspondingly, the request headers *x-amz-sdk-checksum-algorithm* and *x-amz-checksum-algorithm* now support an additional value, *CRC64NVME*.

More information:

- In the HyperStore S3 API documentation, the PutObject, CreateMultipartUpload, CompleteMultipartUpload, RestoreObject, DeleteObject, PutBucketLifecycleConfiguration, PutObjectLockConfiguration, PutBucketReplication, PutBucketTagging, and PutBucketCors calls.

Support for new checksum option "full object" for S3 multipart uploads

HyperStore now supports AWS S3's recently introduced option for clients to specify that the S3 system perform a checksum on the full object content of a completed multipart upload, rather than a composite of the object part checksums. The new request header is called *x-amz-checksum-type* and the options are *COMPOSITE* or *FULL_OBJECT*.

More information:

- "CreateMultipartUpload" and "CompleteMultipartUpload" in the **Cloudian HyperStore AWS APIs Support Reference**.

S3 calls that previously required Content-MD5 header now also accept checksum headers instead

The HyperStore S3 API calls that used to require the presence of a *Content-MD5* request header will now accept the presence of either a *Content-MD5* request header or an *x-amz-checksum-** request header (such as *x-amz-checksum-crc32c* or *x-amz-checksum-crc64nvme*). If neither a *Content-MD5* header nor an *x-amz-checksum-** header is present, the request will fail. This change was made because the most recent versions of the AWS SDK and AWS CLI no longer send *Content-MD5* headers with these requests and now send only an *x-amz-checksum-** header instead.

The affected S3 calls are: DeleteObjects, PutObject (if *x-amz-object-lock-** headers are used), PutBucketLifecycleConfiguration, PutBucketReplication, PutBucketTagging, PutBucketCors, and PutObjectLockConfiguration.

S3 ListObjectVersions call now supports HyperStore extension headers for filtering by date-time range

As a HyperStore extension to the S3 call *ListObjectVersions*, the call now supports request headers to limit the return to object versions created within a specified date-time range, in the specified bucket. With this capability you can determine what object versions were created during a particular time period of interest (such as during a suspected cyber-attack).

More information:

- "ListObjectVersions" in the **Cloudian HyperStore AWS APIs Support Reference**.

Admin API now supports an operation for deleting all object versions created in a bucket within a specified date-time range

The HyperStore Admin API's *bucketops* resource now supports a call that deletes all object versions created in a bucket within a specified date-time range, in a single operation. There are also API calls for checking the operation's status, and for stopping an in-progress operation.

This capability can be used, for example, to recover from a ransomware attack that creates encrypted versions of objects as the most recent object versions.

More information:

- "bucketops" in the *Cloudian HyperStore Admin API Reference*.

System Management and Functionality

CentOS no longer supported as HyperStore host OS

To perform a fresh installation of HyperStore version 8.2, the host machines on which you are installing HyperStore must be running Rocky Linux 8.9 or 8.10, or RHEL 8.9 or 8.10.

To upgrade to HyperStore version 8.2, your existing HyperStore version must be 8.1.x, and your HyperStore host machines must be running Rocky Linux 8.9 or 8.10, or RHEL 8.9 or 8.10. If you have an existing HyperStore deployment and you have not yet worked with Cloudian to migrate the hosts from CentOS to Rocky Linux, you must do so before you can upgrade to HyperStore 8.2.

Running HyperStore 8.2 on CentOS is not supported.

Support for NVIDIA GPUDirect Storage

HyperStore now supports NVIDIA® GPUDirect® Storage, which uses Remote Direct Memory Access (RDMA) for direct transfer of object data between HyperStore and GPU memory on client hosts. This GPUDirect Storage (GDS) feature substantially increases S3 data transfer throughput, reduces latency, and reduces CPU burden. This performance profile makes HyperStore with GDS ideally suited to serve as a scalable data lake for AI and other GPU-centric workloads.

This feature is currently supported **only for new installations**. The feature has special requirements in regard to hardware and drivers.

More information:

- ["GPUDirect Storage Feature Overview" \(page 143\)](#)

Support for Quad-Level Cell (QLC) SSDs for object data storage

HyperStore now supports using Quad-Level Cell (QLC) SSDs as the object data capacity drives on your HyperStore host machines. Consult with your Cloudian sales representative for details about the supported QLC SSD types and sizes. QLC SSDs are **not** recommended for using as the HyperStore metadata drives, since the metadata drives will be subject to very frequent writes of small files. By contrast, QLC SSDs are best suited for read-heavy workloads and large objects.

More information:

- "Host Hardware and Operating System" and "Installing a New HyperStore System" in the *Cloudian HyperStore Installation Guide*.

Bridge Mode auto-tiering now supports filtering by object size

Bridge Mode auto-tiering -- whereby objects uploaded to a bucket are immediately transitioned to a specified destination system rather than being stored in the local system -- now supports filtering by object size. This enables bucket owners to create a lifecycle policy where, for example, incoming large objects are immediately tiered to a destination system while small objects are stored locally. This can be useful if you have a tiering destination system that uses object storage media best suited to large objects and not small objects (QLC SSDs for example).

More information:

- ["Auto-Tiering Feature Overview" \(page 203\)](#)
- ["Configure a Bucket Lifecycle Policy" \(page 315\)](#)

Installer method for restarting services has been deprecated

The HyperStore installer script's "Manage Services" menu has been deprecated as a method of stopping, starting, or restarting services, or checking service instead. Instead, use the `hscctl` command line tool for these tasks.

More information:

- ["Starting, Stopping, and Checking Status of Services" \(page 91\)](#)

Support for multiple KMSs for KMIP based server-side encryption

HyperStore now supports using multiple Key Management Systems (KMSs) for KMIP based server-side encryption. Previously HyperStore only supported using one KMS.

Also, HyperStore now provides a setting that controls who is allowed to configure KMS connections: system admins, group admins, and/or regular users. This is useful in MSP settings where different user groups may use different KMSs.

More information:

- ["Server-Side Encryption" \(page 186\)](#)
- ["Using SSE-KMIP" \(page 191\)](#)

Search Service for search based on object metadata

HyperStore now supports an optional HyperStore Search Service, which enables users to search for their objects based on object metadata.

More information:

- ["Object Metadata and Search Feature Overview" \(page 230\)](#)
- ["Preparing the Metadata Search Feature" \(page 236\)](#)

HyperStore now supports SAML-based access to the CMC

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. Now such users can access the CMC to use functionality permitted to their role. Previously users authenticated via SAML had to use third party S3 clients to access HyperStore's S3 Service.

More information:

- "SAML Support" in the IAM section of the ***Cloudian HyperStore AWS APIs Support Reference***.
- ["Manage IAM Role" \(page 448\)](#)

hsDNS is now production-ready

HyperStore's hsDNS feature -- for service discovery, domain resolution, and load balancing for HyperStore services -- has been enhanced to make it suitable for production environments. In particular:

- hsDNS now implements HTTP level health checks for HyperStore services, rather than simply checking `systemd` for system process status.
- HyperStore now supports automatic failover of the Config Controller primary role, which is important for hsDNS since hsDNS relies on HyperStore's Salt-based configuration management system to trigger updates to DNS records when hsDNS detects changes in service up/down status.

More information:

- ["hsDNS Feature Overview" \(page 139\)](#)
- ["Setting Up hsDNS" \(page 142\)](#)
- [common.configController.failover](#)

HyperStore Firewall enhancements

The HyperStore Firewall now allows more granular configuration control, including the ability to create custom ACLs and the ability to add firewall controls for additional, custom services that you may be running on your HyperStore nodes (such as an SNMP agent).

More information:

- ["HyperStore Firewall" \(page 850\)](#)

Improvements to the per-bucket rate limiting feature

HyperStore's per-bucket rate limiting feature, first introduced in HyperStore 8.1 and designed to prevent any single bucket from negatively impacting service performance for your user population as a whole, has been improved in several ways in HyperStore 8.2:

- The feature now limits incoming S3 requests (of all kinds, in total) to each bucket, as well as Metadata DB (Cassandra) requests generated by each bucket. Previously only the Metadata DB requests generated per bucket were limited.
- The per-bucket thresholds are now calculated dynamically, taking into account a variety of factors regarding your particular HyperStore deployment. Previously the thresholds were statically set, and by default were the same for all customers.
- Certain aspects of the Metadata DB rate limiting feature can now be checked on and controlled through the CMC.

More information:

- ["Quality of Service \(QoS\) Feature Overview" \(page 246\)](#)
- ["Cassandra QoS Commands" \(page 485\)](#)

Within a bucket, the CMC now supports listing object versions created within a date range

In the CMC, an owner of an object that uses versioning can now create a list of object versions created within a specified date-time range.

More information:

- ["List or Search for Objects" \(page 357\)](#)

HyperStore S3 Service now supports blocking suspicious PutObjectCopy operations

HyperStore now supports an option to block S3 *PutObjectCopy* operations that are attempting to copy an object over itself, with the same content except encrypted by SSE-C encryption. This is a potential attack vector for ransomware attacks, once an S3 credential has been stolen. The targeted objects would be encrypted in place, and with an SSE-C passphrase known only to the attackers.

The option to block these operations can be configured on a system-wide basis, and on a per-bucket basis.

More information:

- ["s3.cyberstorage.putObjectCopy.blockOverwriteEncryption" \(page 818\)](#)
- ["Configure Bucket Permissions" \(page 310\)](#)

Users with IAM resources cannot be deleted, by default

Now, through the CMC or through the Admin API, by default you cannot delete a user who still owns IAM resources (IAM groups, users, roles, and/or policies).

If you prefer, you can configure the system to allow deletion of such users. This was the behavior in previous HyperStore versions.

More information:

- ["common.user.allowDeleteWithIAMResources"](#) (page 754)

hsstool repairec command now repairs EC Hybrid storage policy data including replica data

If you are using EC Hybrid storage policies in your system (storage policies whereby large objects are erasure coded and small objects are replicated), one run of the *hsstool repairec* command will now repair -- across a whole data center -- all the replica data associated with such storage policies as well as all of the erasure coded data.

Previously for EC Hybrid storage policies you had to run *hsstool repairec* to repair the erasure coded data and then run *hsstool repair* on each node to repair the replicated data.

More information:

- ["Hybrid Storage Policies"](#) (page 151)
- ["hsstool repairec"](#) (page 918)
- ["Auto-Repair Schedule Settings"](#) (page 507)

hsstool repairec command now supports repairing the data in a specific bucket

The *hsstool repairec* command now supports a *-b <bucket>* option that you can use if you want to repair just the data associated with a particular bucket that is using an EC storage policy (or EC Hybrid storage policy).

More information:

- ["hsstool repairec"](#) (page 918)

hsstool repairec command now supports rebuilding a disk during a cluster rebalance

The command *hsstool repairec -d <mountpoint>* -- for repairing/rebuilding the erasure coded data on a particular disk (and also replica data associated with EC Hybrid storage policies) -- now supports a *-reb* option by which you can repair/rebuild a disk while a cluster rebalance is in progress in your system, after you've added nodes to the cluster. Previously repairing or rebuilding a disk was not allowed while a cluster rebalance was in progress.

More information:

- ["hsstool repairec"](#) (page 918)

hsstool whereis command enhanced to show data to be rebalanced

In the event that you have added nodes to your HyperStore system but the rebalance operations have not yet completed, the *hsstool whereis* command now supports an option to show object replicas or fragments that will be relocated from an existing node to a new node during the rebalance operations.

More information:

- ["hsstool whereis"](#) (page 938)

hsstool cleanup and hsstool cleanupec commands now prioritize the fullest disks

The *hsstool cleanup* and *hsstool cleanupec* operations have been optimized so that on the target host, the

data disks are cleaned in order from the fullest disks to the least full disks. Note that since by default the *hsstool cleanup* and *hsstool cleanupec* operations clean 10 data disks concurrently, this new behavior prioritizing the fullest disks comes into play only if the HyperStore host being cleaned has more than 10 data disks.

More information:

- ["hsstool cleanup" \(page 864\)](#)
- ["hsstool cleanupec" \(page 869\)](#)

hsstool repair command now supports concurrently repairing specified disks on different nodes

You can now use *hsstool -repair* command options to concurrently repair replicated data on specified disks on different nodes. Previously this capability to repair multiple disks in the cluster concurrently was only supported for erasure coded data, using *hsstool repairec* command options.

Note that you cannot repair multiple disks on the same node concurrently. You can only repair multiple disks concurrently if the disks are on different nodes, so that on any individual node only one targeted disk is being repaired at a time.

More information:

- ["hsstool repair" \(page 907\)](#)
- ["hsstool repairec" \(page 918\)](#)

hsstool repair command now supports repairing replicated object data specified in an input file

The *hsstool repair* command now supports an *hsstool repair -f <input-file>* option by which you can repair replicated object chunks specified in an input file. The input file can be a *cloudian-hyperstore-repair-failure.log* file or a file that you create.

Previously only the *hsstool repairec* command supported this option.

More information:

- ["hsstool repair" \(page 907\)](#)
- ["hsstool repairec" \(page 918\)](#)

HyperStore's auto-tiering feature now supports AWS Glacier Deep Archive as a destination

HyperStore's auto-tiering feature has been extended to include AWS Glacier Deep Archive as a supported destination.

Note that in the CMC interface and in HyperStore configuration settings:

- "AWS Glacier" denotes what AWS now calls "Glacier Flexible Retrieval"
- "AWS Deep Archive" denotes AWS Glacier Deep Archive

More information:

- ["Auto-Tiering Feature Overview" \(page 203\)](#)
- ["Preparing the Auto-Tiering Feature" \(page 209\)](#)
- ["Configure a Bucket Lifecycle Policy" \(page 315\)](#)
- "PutBucketLifecycleConfiguration" in the ***Cloudian HyperStore AWS APIs Support Reference***.

More granular configuration controls for using "X-Forwarded-For" headers in incoming S3 requests

HyperStore now supports two new configuration settings relating to the use of "X-Forwarded-For" headers as a means of identifying the source IP address of an incoming S3 request:

- If `common.xForwardedHeaders.enabled` is set to `true`, the system uses "X-Forwarded-For" headers for request logging.
- If `common.xForwardedHeaders.enabled` is set to `true` and also `common.xForwardedHeaders.trusted` is set to `true`, the system uses "X-Forwarded-For" headers for request logging and also for bucket policy and IAM policy enforcement (when policies include conditions based on source IP address of the request).

Both of these settings default to `false`.

In previous versions of HyperStore there was just one setting `logging.xForwardedFor` (which defaulted to `false`), and if you set this to `true` then "X-Forwarded-For" headers were used both for request logging and for policy enforcement. The new settings replace the old `logging.xForwardedFor` setting and allow for more granular control -- in particular, they allow you to use "X-Forwarded-For" headers for request logging but not for policy enforcement if you wish.

If in HyperStore 8.0.x or 8.1.x you configured `logging.xForwardedFor` to `true`, when you upgrade to HyperStore 8.2 your settings will be:

- `common.xForwardedHeaders.enabled=true`
- `common.xForwardedHeaders.trusted=false`

Consequently, if you want HyperStore to use "X-Forwarded-For" headers for bucket policy and IAM policy enforcement you must set `common.xForwardedHeaders.trusted` to `true` after executing the upgrade to HyperStore 8.2

More information:

- ["common.xForwardedHeaders.enabled"](#) (page 759)
- ["common.xForwardedHeaders.trusted"](#) (page 760)

HyperStore 8.1.1

APIs

Partial support for S3 'Accelerate', 'Analytics', 'IntelligentTiering', and 'Metrics' configuration calls

The HyperStore S3 Service now provides a greater degree of compliance with these AWS S3 calls:

- `Put{Accelerate,Analytics,IntelligentTiering,Metrics}Configuration` -- Not supported, will return 405 'MethodNotAllowed'
- `Get{Accelerate,Analytics,IntelligentTiering,Metrics}Configuration` -- Supported, but there will be no configuration to return since the corresponding Put operations are not supported
- `List{Analytics,IntelligentTiering,Metrics}Configuration` -- Supported, but there will be no configurations to list since the corresponding Put operations are not supported
- `Delete{Analytics,IntelligentTiering,Metrics}Configuration` -- Supported, but there will be no configuration to delete since the corresponding Put operations are not supported

Previously all of these calls returned 501 'NotImplemented'.

More information:

- ***Clouidian HyperStore AWS APIs Support Reference***

System Management and Functionality

HyperStore Rocky Linux ISO now uses Rocky version 8.10

The HyperStore Rocky Linux ISO, which you will use if you want to install one or more HyperStore File Service pools, now uses Rocky Linux version 8.10. In HyperStore version 8.1, that ISO used Rocky 8.9.

Note HyperStore supports mixed environments where some File Service pools (installed with HyperStore 8.1) run on Rocky 8.9 while other File Service pools (installed with HyperStore 8.1.1 or later) run on Rocky 8.10.

More information:

- "Installing the HyperStore File Service" in the ***Cloudian HyperStore Installation Guide***.

hsstool repairqueue command is enhanced to provide repair-on-read visibility and control

The `hsstool repairqueue` command can now report on/off status for the HyperStore repair-on-read feature, and includes granular support for disabling or re-enabling the repair-on-read feature. Previously this command was oriented exclusively toward the scheduled auto-repair feature, and did not provide visibility or control for the repair-on-read feature.

More information:

- ["hsstool repairqueue" \(page 925\)](#)
- ["Automated Data Repair Feature Overview" \(page 130\)](#)

Increase in Metadata DB request rate limit per bucket

The "Rate Limiting for Metadata DB Requests Per Bucket" feature introduced in HyperStore 8.1 applied a limit of 100,000 Metadata DB queries per 5 minute interval for each bucket. This limit proved to be too low in some customer environments, and resulted too frequently in buckets being temporarily put into restriction. Consequently in HyperStore 8.1.1 the limit has been raised to 800,000 Metadata DB queries per 5 minute interval for each bucket. This feature will continue to be improved in future releases as more field data is gathered.

More information:

- ["Automatic Rate Limiting Per Bucket" \(page 248\)](#)

Log message forwarding feature now includes Firewall Service and HyperStore Shell logging

The existing feature that enables you to have HyperStore forward its log messages to a central `rsyslog` server has been extended to include Firewall Service logging and HyperStore Shell logging. This log message forwarding feature is disabled by default configuration.

More information:

- ["logging.syslog.remote.*" \(page 803\)](#)

HyperStore 8.1

APIs

S3 API support for Additional Checksum Algorithms

HyperStore's S3 API implementation now supports the AWS S3 "Additional Checksum Algorithms" functionality. This functionality supplements existing S3 object integrity safeguards by enabling S3 application users to select from one of several supported checksum algorithms when uploading objects, and to have the calculated checksum value stored as part of the object metadata.

More information:

- For example, "PutObject" in the ***Cloudian HyperStore AWS APIs Support Reference***.

Note This new functionality is not currently supported by the CMC. It is only supported by HyperStore's S3 API.

Admin API support for new element in 'UserInfo' object

For the Admin API's *PUT /user* and *GET /user* calls, the *UserInfo* object includes a new element "fileEndpoints" that indicates the File Service pools to which the user is allowed to publish file shares. Applicable only to HyperStore users for whom you enable File Services functionality.

More information

- "user" in the ***Cloudian HyperStore Admin API Reference***
- **"File Services Feature Overview"** (page 238)

System Management and Functionality

HyperStore File Service: Installation improvements

The HyperStore File Service is now easier to install and can be installed without internet connectivity.

More information:

- "Installing the HyperStore File Service" in the ***Cloudian HyperStore Installation Guide***
- **"File Services Feature Overview"** (page 238)

HyperStore File Service: Support for multiple File Service pools and Geo-Distributed File Shares

You can now install multiple File Service "pools" in your HyperStore system. Each pool resides in a single data center and consists of either one File Service node or three File Service nodes. You can install different File Service pools in different locations. For example if your HyperStore system spans multiple data centers you can install a File Service pool in each data center. HyperStore users who own file shares can optionally have the file shares be geo-distributed across two File Service pools in different locations with different sets of end users, and the system will keep the content in sync between the two locations.

More information:

- "Installing the HyperStore File Service" in the ***Cloudian HyperStore Installation Guide***
- **"File Services Feature Overview"** (page 238)

- ["File Service Settings"](#) (page 510)
- ["Add a File Share"](#) (page 368)

HyperStore File Service: Support for `hscctl`-driven node replacement operations

For a single-node File Service pool you can now revert the node to a prior backup, or replace a failed node with a new node, using the `hscctl` command line tool to drive most aspects of the operations.

For a three-node File Service pool you can now replace a failed node with a new node, using the `hscctl` command line tool to drive most aspects of the operation.

More information:

- ["Backing Up and Restoring a Single-Node File Service Pool"](#) (page 649)
- ["Replacing a Node in a Three-Node File Service Pool"](#) (page 647)

HyperStore File Service: Support for WORM mode

HyperStore users for whom File Services are enabled can now create file shares that use an Object Lock enabled bucket for back-end storage, and the front-end file share will operate in WORM (Write Once Read Many) mode.

More information:

- ["WORM \(Object Lock\)"](#) (page 240)
- ["Add a File Share"](#) (page 368)

IMPORTANT ! While object data storage in the share's back-end bucket will comply fully with U.S. SEC-17a and comparable regulatory regimes (if your system has a license for Certified Object Lock), the share's file data stored in the File Services front-end cache will not. The cached file data will be managed according to general WORM principles, but the cache implementation does not meet all of the stringent requirements of regulations such as U.S. SEC-17a.

HyperStore File Service: Support for automated uninstall

The HyperStore File Service now supports a simple, automated means of uninstalling a File Service pool.

More information:

- "Uninstalling a File Service Pool" in the ***Cloudian HyperStore Installation Guide***

Change in HyperStore license expiration behavior

Now, if your license period and grace period expire, the HyperStore system will still support S3 GET and HEAD operations -- and so, existing can be listed and read -- but will not support other S3 operations (such as the PUTs associated with creating buckets or uploading objects). Previously, if your license period and grace period expired system will not support any types of S3 operations.

More information:

- ["Licensing and Auditing"](#) (page 36)

Metadata DB (Cassandra) rate limiting

HyperStore now has a feature that automatically ensures that no single bucket places excessive load on the Metadata DB (Cassandra). This prevents any single bucket from negatively impacting service performance for your user population as a whole.

More information:

- ["Automatic Rate Limiting Per Bucket" \(page 248\)](#)

MFA is now supported for LDAP-authenticated CMC users

CMC users whose logins are authenticated by way of your organization's LDAP system now have the ability to set up Multi-Factor Authentication (MFA) for their CMC logins. Previously MFA was available for CMC users whose login passwords were maintained within the CMC but not for LDAP-authenticated users.

More information:

- ["Security Credentials" \(page 572\)](#)

Additional configuration settings supported by hscctl

The *hscctl* commands *hscctl config set* and *hscctl config get* now support additional configuration settings including:

- ["admin.logging.level" \(page 704\)](#)
- ["cmc.logging.level" \(page 720\)](#)
- ["cmc.sso.tolerance" \(page 727\)](#)
- ["cmc.ui.page.defaultSize" \(page 740\)](#)
- ["cmc.ui.page.maxSize" \(page 740\)](#)
- ["cmc.ui.security.serviceInfo" \(page 740\)](#)
- ["common.user.password.maxLength" \(page 757\)](#)
- ["hyperstore.cleanup.objectDelete.queue.maxLength" \(page 765\)](#)
- ["hyperstore.logging.level" \(page 770\)](#)
- ["hyperstore.messagebus.connections.timeout" \(page 772\)](#)
- ["hyperstore.repair.EC.queue.size" \(page 772\)](#)
- ["hyperstore.threadPool.cleanup.jobs" \(page 780\)](#)
- ["iam.logging.level" \(page 791\)](#)
- ["Log Rolling Settings" \(page 689\)](#)
- ["monitoring.event.acknowledgedTTI" \(page 807\)](#)
- ["s3.batchOperations.delete.maxPerDC" \(page 813\)](#)
- ["s3.batchOperations.purge.threads.max" \(page 814\)](#)
- ["s3.client.connectionTimeout" \(page 815\)](#)
- ["s3.client.maxRetries" \(page 816\)](#)
- ["s3.client.socketTimeout" \(page 816\)](#)
- ["s3.client.upload.multipart.partSize" \(page 816\)](#)
- ["s3.logging.level" \(page 823\)](#)
- ["s3.sse.kms.region" \(page 831\)](#)
- ["s3.user.maxCredentials" \(page 841\)](#)
- ["sqs.logging.level" \(page 843\)](#)

Additional commands supported by HyperStore Shell

The HyperStore Shell (HSH) now supports additional commands, specific to particular hardware platforms: *sg_*

ses, ledctl, storcli64, MegaCli64, and ssacli.

More information:

- ["Using the HSH" \(page 111\)](#)

HyperStore 8.0

APIs

None

System Management and Functionality

New configuration management infrastructure fronted by a new command line tool, hscctl

Salt has replaced Puppet as the cluster configuration management technology used within HyperStore. In the context of HyperStore, Salt provides more versatility as well as better scalability and performance.

Your primary way of working with HyperStore system configuration is now a command line tool called *hscctl*. You will no longer edit configuration files. Instead you can make a configuration change, push your change to the cluster, and restart the affected service(s), all with simple *hscctl* commands.

As in past versions of HyperStore, you can still manage a limited number of configuration settings through the CMC's **Configuration Settings** page and through the installer's "Advanced Configuration Options" section. But these same settings are now also manageable via *hscctl*, and beyond that, *hscctl* also provides access to a wide range of settings not available in the CMC or the installer -- settings that required manual editing of configuration files in previous HyperStore versions.

More information:

- ["Using hscctl to Control HyperStore Configuration" \(page 693\)](#)
- [HyperStore Configuration Settings](#)
- ["HyperStore Configuration Settings" \(page 700\)](#)
- ["Starting, Stopping, and Checking Status of Services" \(page 91\)](#)
- ["Security and Privacy Features" \(page 177\)](#)

Rocky Linux 8.7 is required for new HyperStore installs

To do a **fresh install** of HyperStore 8.0 your host machines **must be running Rocky Linux 8.7 (or RHEL 8.7)**.

Fresh installs of HyperStore 8.0 on CentOS 7 or RHEL are not supported.

Upgrades from HyperStore 7.5.x to HyperStore 8.0 on CentOS/RHEL 7 are supported.

More information:

- ["Host Hardware and Operating System" \(page 46\)](#)

Improved built-in DNS component for non-production systems

In a non-production system, you can enable the new HyperStore service **hsDNS** to act as an authoritative nameserver on all nodes and provide DNS based Service Discovery. Built on CoreDNS technology, hsDNS automatically manages DNS resolving of all service endpoints within your HyperStore cluster, and takes into account the up/down status of service instances on each node as well as the up/down status of nodes.

Optionally hsDNS supports local affinity, and exposure on an external network to hook into your DNS infrastructure.

The hsDNS service is disabled by default configuration.

Note that hsDNS replaces *dnsmasq* from previous versions of HyperStore.

More information:

- ["lb.hsDNS.enabled"](#) (page 799)

Support for multipath I/O (MPIO) for storage devices

HyperStore now supports multipath I/O (MPIO) for storage devices. The following HyperStore disk management functionalities now accommodate MPIO:

- *system_setup.sh* (for configuring hosts before HyperStore installation)
- CMC functions for displaying disk status, generating alerts, and bringing disabled disks back online

More information:

- ["View a Node's Disk Detail"](#) (page 474)
- ["Alert Rules"](#) (page 560)
- ["Enabling a HyperStore Data Disk"](#) (page 640)

HyperStore File Services feature now supports file share snapshots

Users for whom you have enabled File Services functionality can now use the CMC to set up a regular schedule for the system to automatically create snapshots of a file share. A file share "snapshot" is a point-in-time image of a file share, which preserves pointers to the object data in a bucket as it exists at a particular moment in time. Leveraging the bucket versioning feature, a file share snapshot gives SMB and NFS client application users the ability to browse the file share as it existed in the past, and download objects as they existed in the past.

More information:

- ["Set File Share Custom Properties"](#) (page 373)

HyperStore File Services feature now supports bi-modal access

Users for whom you have enabled File Services functionality can now use the CMC to enable bi-modal access to a file share and its corresponding bucket. When bi-modal access is enabled, read/write access is supported not only for SMB/NFS clients of the file share but also for S3 clients reading/writing directly to the corresponding bucket via the S3 protocol. In particular, S3 client applications can add or make changes to the contents of a bucket and SMB/NFS client applications will be able to see and access that changed content in the corresponding file share.

More information:

- ["Set File Share Custom Properties"](#) (page 373)

HyperStore Search feature is temporarily unsupported

The HyperStore Search feature first introduced in HyperStore 7.5 -- for searching for objects by reference to their object metadata -- is not supported in HyperStore 8.0. It will be supported again in a future HyperStore 8.x version.

Chapter 1. Introduction to HyperStore

1.1. HyperStore Documentation

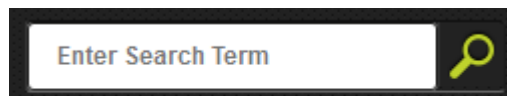
The HyperStore user documentation consists of:

- Cloudian HyperStore Help (HTML5)
- Cloudian HyperStore Administrator's Guide (PDF)
- Cloudian HyperStore Installation Guide (PDF)
- Cloudian HyperStore Admin API Reference (PDF)
- Cloudian HyperStore AWS APIs Support Reference (S3, IAM, STS, & SQS APIs) (PDF)

The HyperStore Help is available through the CMC (by clicking the Help button) and is also available in the directory `/opt/cloudian-staging/8.2/doc/HyperStoreHelp` on your Config Controller node (in that directory you can open the `HyperStoreHelp.html` file). The PDF guides are available in the directory `/opt/cloudian-staging/8.2/doc/HyperStorePDFManuals` on the Config Controller node.

The Help has all of the content from the Administrator's guide and the API guides.

The Help features a built-in search engine. The search box is in the upper right of the interface. As with any search engine, enclose your search phrase in quotes if you want to limit the results to exact match only.



In the Help, in most cases screen shots are presented initially as small thumbnail images. This allows for a more compact initial view of the content on a page and makes it easier for you to skim through the text on the page. If you want to see the full size image simply hold your cursor over the thumbnail image. When you're done looking at the full size image just move your cursor off of it and it will disappear, leaving only the thumbnail image.

Also in the interest of presenting a compact initial view of the content on a page, the Help often makes use of expandable/collapsible text. To expand (or subsequently collapse) such text you can click on the triangle icon to the left of the text or on the text itself.

Example of collapsed text in initial view of a Help page:

group

[Admin API Methods]

The Admin API methods built around the **group** resource are for managing HyperStore service user groups. This includes support for creating, changing, and deleting user groups, and also for assigning rating plans to groups.

- ▶ **DELETE /group** Delete a group
- ▶ **GET /group** Get a group's profile
- ▶ **GET /group/list** Get a list of group profiles
- ▶ **GET /group/ratingPlanId** Get a group's rating plan ID
- ▶ **POST /group** Change a group's profile
- ▶ **POST /group/ratingPlanId** Assign a rating plan to a group
- ▶ **PUT /group** Create a new group

Example of that same page with the first expandable text item expanded:

group

[Admin API Methods]

The Admin API methods built around the **group** resource are for managing HyperStore service user groups. This includes support for creating, changing, and deleting user groups, and also for assigning rating plans to groups.

▼ **DELETE /group** Delete a group

Syntax

```
DELETE /group?groupId=string
```

There is no request payload.

Parameter Descriptions

▼ groupId

(Mandatory, string) Unique identifier of the group.

Usage Notes

Before you can delete a group you must first delete all users associated with the group, using the [DELETE /user](#) method.

Example Using cURL

The [example](#) below deletes the "QA" group.

```
curl -X DELETE -k -u sysadmin:password https://localhost:19443/group?groupId=QA
```

Response Codes

This method will return one of the [Common Response Status Codes](#) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameters : {groupId}
400	Group does not exist

To expand or collapse all of the expandable/collapsible text on a page, click this button in the upper left of the Help interface:



The Help is also responsive to the type of user logged in to the CMC: a system administrator logged in to the CMC sees the full Help content, but a group administrator or regular user logged in to the CMC sees only the content applicable to their role. Also, in the Help that group admins and regular users see, there is no Clouidian branding and no references to HyperStore or the Clouidian Management Console.

Note A system administrator logged into the CMC can download any of the HyperStore PDF documents listed above. A group administrator or regular user logged into the CMC can only download the *Clouidian HyperStore AWS APIs Support Reference* PDF document.

1.2. HyperStore Overview

Clouidian® HyperStore® is a multi-tenant object storage system that fully supports the Amazon Simple Storage System (S3) API. The HyperStore system enables any service provider or enterprise to deploy an S3-compliant multi-tenant storage cloud.

The HyperStore system is designed specifically to meet the demands of high volume, multi-tenant data storage:

- **Amazon S3 API compliance.** The HyperStore system is fully compatible with Amazon S3's HTTP REST API. Customers' existing S3 applications will work with the HyperStore service, and existing S3 development tools and libraries can be used for building HyperStore client applications.
- **Secure multi-tenancy.** The HyperStore system provides the capability to securely have multiple user groups reside on a single, shared infrastructure. Data for each user is logically separated from other users' data and cannot be accessed by any other user unless access permission is explicitly granted.
- **Quality of service controls.** HyperStore system administrators can set storage quotas and usage rate limits on a per-group and per-user basis. Group administrators can set quotas and rate controls for individual members of the group.
- **Access control rights.** Read and write access controls are supported at per-bucket and per-object granularity. Objects can also be exposed via public URLs for regular web access, subject to configurable expiration periods.
- **Reporting and billing.** The HyperStore system supports usage reporting on a system-wide, group-wide, or individual user basis. Billing of groups or users can be based on storage quotas and usage rates (such as bytes in and bytes out).
- **Tiering and replication to external systems.** The HyperStore system supports having objects be tiered (moved) to an external S3-compatible destination system on a defined schedule. The system also supports having objects be replicated to an external S3-compatible destination system.
- **Object Lock (Write Once Read Many).** The HyperStore system supports the S3 APIs for applying Object Lock to buckets and objects, so that object versions cannot be deleted or altered for a defined retention period.
- **Horizontal scalability.** Running on commodity off-the-shelf hardware, a HyperStore system can scale out to hundreds of nodes across multiple data centers, supporting millions of users and hundreds of petabytes of data. New nodes can be added without service interruption.
- **High availability.** The HyperStore system has a fully distributed, peer-to-peer architecture, with no single point of failure. The system is resilient to network and node failures with no data loss due to the automatic replication and recovery processes inherent to the architecture. A HyperStore cluster can be deployed across multiple data centers to provide redundancy and resilience in the event of a data center scale disaster.

1.3. Licensing and Auditing

Subjects covered in this section:

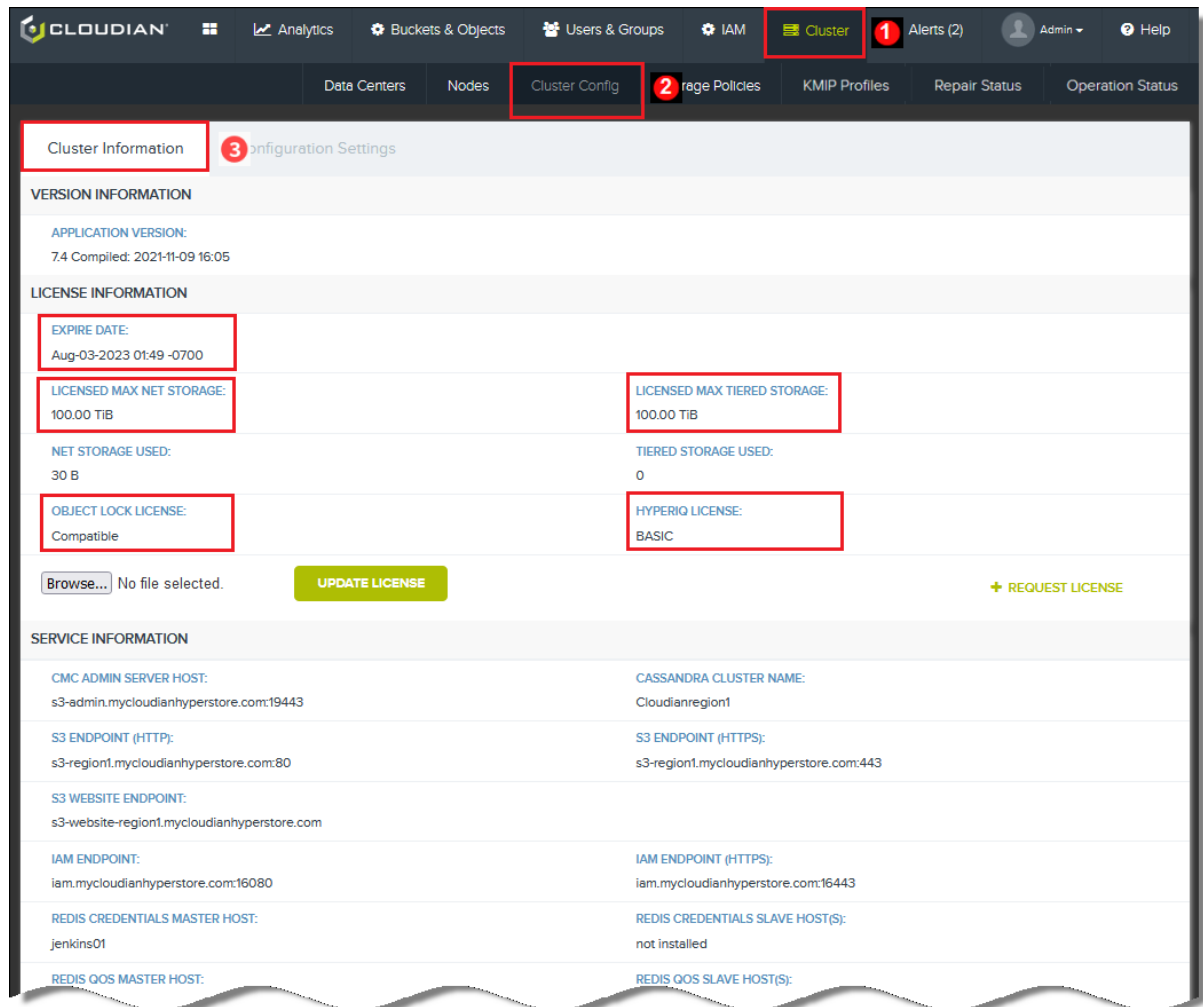
- *Introduction (immediately below)*
- ["License Expiration" \(page 37\)](#)
- ["Licensed Maximum On-Premise Storage Usage" \(page 38\)](#)
- ["Licensed Maximum Tiered Storage Usage" \(page 39\)](#)
- ["Object Lock \(WORM\) License" \(page 41\)](#)
- ["HyperIQ License" \(page 41\)](#)
- ["License Updating" \(page 42\)](#)
- ["Auditing" \(page 42\)](#)

A valid Cloudian® software license is required to run HyperStore® software. Evaluation licenses are available as well as production licenses. Before using HyperStore software, you must obtain a license from Cloudian.

A Cloudian HyperStore license has these key attributes:

- Expiration date
- Maximum allowed on-premise storage volume
- Maximum allowed tiered storage volume
- Object lock functionality enabled or disabled
- HyperIQ license level

You can see the attributes of your particular HyperStore license by accessing the CMC's **Cluster Information** page (**Cluster -> Cluster Config -> Cluster Information**).



The sections that follow describe these attributes and their enforcement in more detail.

1.3.1. License Expiration

Each Cloudian HyperStore license has an expiration date. Also as part of your license configuration there is a warning period (which commences prior to the expiration date) and a grace period (which extends beyond the expiration date).

If you reach the **warning period** preceding your license expiration, then when you use any part of the CMC, the top of the interface displays a warning that your license expiration date is approaching.

If you reach your license **expiration date** you enter a grace period, per the terms of your contract. During the grace period:

- In the CMC, the top of the screen displays a warning indicating that your license has expired and that your HyperStore system will be disabled in a certain number of days (the number of days remaining in your grace period).
- The system still accepts and processes all types of incoming S3 requests, but every S3 response returned by the S3 Service includes an extension header indicating that the system license has expired (header name: *x-gemini-license*; value: *Expired: <expiry_time>*).

If you reach the **end of your grace period** after the license expiration date:

- No S3 service is available for end users except for GET and HEAD operations (so existing can still be listed and read). All other types of incoming S3 requests -- such as PUT operations associated with creating buckets or uploading objects -- will be rejected with a "403 Forbidden" error response. The response also includes the expiration header described above.
- You can still log into the CMC to perform system administration functions (including applying an updated license), but your access to users' stored S3 objects will be read-only (GET and HEAD operations).
- The top of the CMC screen will display an error message indicating that your license has expired. Also, in the CMC's [Dashboard](#) page the Cluster Health panel will indicate that the system status is "Read Only".

It's best to update your license well in advance of your license expiration date. See "[License Updating](#)" (page 42) below.

1.3.2. Licensed Maximum On-Premise Storage Usage

Depending on your particular license terms, your HyperStore system will have either a Net storage limit or a Raw storage limit, for on-premise data storage.

With a license based on **Net** storage volume -- also known as "Usable" volume -- the limit is on total object storage bytes minus overhead from [storage policies](#) (object replication or erasure coding). For example if a 1GiB object is replicated three times in your system it counts as only 1GiB toward a Net storage limit. A Net storage license is typically used if your cluster consists entirely of software-only nodes, with no HyperStore Appliance nodes.

With a license based on **Raw** storage volume the limit is on the total raw storage capacity used in your system. All HyperStore object data and metadata counts toward this limit, including storage overhead from replication or erasure coding. For example if a 1GiB object is replicated three times in your system it counts as 3GiB toward a Raw storage limit. Likewise, all object metadata and system metadata count toward a Raw storage limit.

A Raw storage license is typically used in either of two types of environments:

- Appliance-only environment. Each HyperStore Appliance has its own amount of licensed Raw storage capacity. If your system consists entirely of HyperStore Appliances, then the Raw licensed storage capacity for your whole system is the simply sum of the individual Appliance licensed capacities.
- Mixed environment of Appliances and software-only nodes. In a mixed environment, the Raw licensed storage capacity for your whole system is the sum of the individual Appliance licensed capacities plus an additional raw capacity allowance that Clouidian builds into your license to accommodate the software-only nodes.

If your license is based on Raw storage, then your total licensed Raw storage limit will be automatically increased if you add a new HyperStore Appliance node to the system. The amount of Raw storage added to your licensed system maximum depends on the particular HyperStore Appliance that you've added to your system. Conversely, if you remove an Appliance from your system this will reduce your total licensed system Raw storage maximum; and the Raw storage allowance associated with a particular Appliance machine cannot be transferred to other nodes in your system.

Note If your HyperStore licensed maximum storage is in terms of Net bytes, adding a HyperStore Appliance to your cluster will not change your Net storage limit. If you are interested in increasing your Net storage limit or converting to a Raw storage limit, consult with Clouidian Support.

In the CMC's **Cluster Information** page you can view the Net or Raw licensed usage maximum for your whole system and also your current system-wide Net or Raw bytes usage count. If your current usage level exceeds 70% of your licensed maximum usage the CMC displays a warning message in both the **Cluster Information** page and the **Dashboard** page. If your current usage level exceeds 90% of your licensed maximum usage the CMC displays a critical message in both of those pages.

Your total system storage usage maximum will be **automatically enforced** by the system no longer allowing S3 clients to upload data to the system. This enforcement will kick in when your system stored byte count reaches **110%** of your licensed maximum usage. At such point the system will reject S3 PUT and POST requests and return an error to the S3 clients. This will continue until one of the following occurs:

- You delete object data so that the system byte count falls below **100%** of your licensed maximum. HyperStore checks every five minutes to see if your storage usage has fallen below the licensed maximum, and if it does fall below the maximum then S3 PUTs and POSTs will again be allowed.

Note In the case of Raw usage, your deletions will not impact your system's raw usage count until the next run of the automatic purging process that occurs hourly on each node. By contrast, a Net usage count is decremented immediately when you delete objects.

- You acquire and install a new license with a larger storage maximum (see ["License Updating"](#) (page 42)). Upon new license installation, S3 PUTs and POSTs will again be allowed.

If the system stored byte count reaches 110% of your licensed maximum usage the CMC will display a pop-up warning message to the system administrator whenever he or she logs in. This will recur on each login event until the system byte count falls below 100% of usage, or a new license with larger storage maximum has been installed.

IMPORTANT ! Regardless of whether your system has a Net storage license or a Raw storage license, **if the data disks on a node become 90% full then that node will stop accepting new S3 writes**. This is not a license enforcement mechanism but rather a system safety feature. For more information see ["Automated Disk Usage Management Feature Overview"](#) (page 124).

1.3.2.1. The Effect of Versioning on On-Premise Storage Volume Measurement

If some users have versioning enabled on their buckets -- so that the system retains rather than overwriting older versions of an object when the user uploads a new version of the object -- then each stored object version (the older versions as well as the current version) counts toward your system stored bytes count.

1.3.2.2. The Effect of Cross-Region Replication on On-Premise Storage Volume Measurement

If some users use the cross-region replication feature to replicate objects from one HyperStore bucket to another HyperStore bucket within the same HyperStore system, then the original source objects and the object replicas in the destination bucket both count toward your system stored bytes count. For more information on cross-region replication see ["Cross-Region Replication Versus Auto-Tiering"](#) (page 214).

1.3.3. Licensed Maximum Tiered Storage Usage

HyperStore supports an auto-tiering feature that users can enable on a per-bucket basis. With auto-tiering, objects can be automatically moved on a configurable schedule to an external destination system such as

Amazon S3 or Glacier, Microsoft Azure, or Google Cloud Storage. For more information on auto-tiering see ["Auto-Tiering Feature Overview" \(page 203\)](#).

In regard to HyperStore licensing, the system treats auto-tiered data as a separate category than on-premise data. Data that's been auto-tiered out of your HyperStore system and is now stored in an external, third party system counts toward a Maximum Tiered Storage limit -- not toward your on-premise storage limit. The Maximum Tiered Storage limit is based on Net (also known as Usable) bytes, not Raw bytes.

The enforcement of this separate tiered storage limit works in largely the same way as the enforcement of the on-premise storage limit.

In the CMC's **Cluster Information** page you can view your tiered storage limit and also your current tiered storage usage level. If your tiered usage level exceeds 70% of your licensed tiered storage maximum the CMC displays a warning message in the **Cluster Information** page. This becomes a critical message if the usage exceeds 90% of licensed maximum.

The tiered usage maximum will be **automatically enforced** by the system no longer allowing auto-tiering to any third party destination system. This enforcement will kick in when your tiered storage byte count reaches **110%** of your licensed maximum. At this point auto-tiering to third party destinations will no longer work until one of the following occurs:

- Through a HyperStore interface -- i.e. the CMC or the HyperStore S3 API -- you delete auto-tiered data so that the total tiered byte count falls below 100% of your licensed maximum. HyperStore checks every five minutes to see if your tiered storage usage has fallen below the licensed maximum, and if it does fall below the maximum then auto-tiering to non-HyperStore destinations is allowed again and automatically resumes.

IMPORTANT ! If you're trying to reduce your tiered storage volume to below your licensed maximum, be sure to delete auto-tiered objects **through a HyperStore interface** and not directly through one of the tiering destination system's interfaces. If you do the latter, HyperStore will not detect that you've reduced your tiered storage volume. For more information see ["Accessing Auto-Tiered Objects" \(page 211\)](#).

- You acquire and install a new license with a larger tiered storage maximum (see ["License Updating" \(page 42\)](#)). Upon new license installation, auto-tiering to third party destinations will be allowed again and will automatically resume.

If the tiered byte count reaches 110% of your licensed maximum usage the CMC will display a pop-up warning message to the system administrator whenever he or she logs in. This will recur on each login event until the tiered byte count falls below the licensed maximum, or a new license with larger tiered storage maximum has been installed.

If auto-tiering to third party destination systems stops because you've exceeded 110% of your licensed maximum, and then later resumes when you come back into compliance with your license, the system will auto-tier any objects that were flagged for auto-tiering during the time period when auto-tiering was halted for license non-compliance. So as long as you come back into compliance, the period of non-compliance will not result in any permanent failures to auto-tier objects that are supposed to have been auto-tiered based on users' bucket lifecycle configurations.

1.3.3.1. Tiered Storage Licensing Exceptions and Qualifiers

- Auto-tiered objects count toward your tiered storage licensed maximum, not your on-premise storage licensed maximum. However, for each auto-tiered object there is a small bit of **object metadata** (8KB per object) that is retained on-premise and counts toward your on-premise storage limit.
- The auto-tiering feature supports an option (configurable on a per-bucket basis) to **retain a local copy** of auto-tiered objects for a specified period of time. If this option is used, then the retained local copy counts toward your on-premise storage limit (while the tiered object copy counts toward your tiered storage limit), until the local copy reaches the end of its retention period and is automatically deleted from local storage.
- The auto-tiering feature supports an option to **temporarily restore** tiered objects into local storage. This works by downloading a copy of the object. During the time period that the object is locally restored, the object counts toward your on-licensed on-premise storage limit as well as toward your licensed tiered storage limit. The same is true when the "cache" (stream and restore) method is used for GETs of tiered objects -- tiered objects that have been cached count toward your on-licensed on-premise storage limit as well as toward your licensed tiered storage limit.
- The auto-tiering feature supports an option (configurable on a per-bucket basis) to tier to a HyperStore destination -- either a different service region within the same HyperStore, or an entirely separate HyperStore system. **Tiering to a HyperStore destination does not count toward the licensed tiering limit.** Instead it counts toward the target HyperStore system's licensed on-premise storage limit.

1.3.4. Object Lock (WORM) License

Your license may or may not include support for the HyperStore Object Lock (WORM) feature. To check whether your license supports this feature, see the CMC's **Information** page: in the "Object Lock License" field it will indicate either "Disabled" or "Compatible" or "Certified". If Disabled, you cannot use the Object Lock feature -- the system will return a 403 Forbidden response if an S3 client application attempts to create a new bucket with object lock enabled.

For more information about this feature, including descriptions of the two types of licensed Object Lock support -- "Compatible Object Lock" and "Certified Object Lock" -- see ["Object Lock Feature Overview" \(page 221\)](#).

If your current license does not support Object Lock and you want to use this feature, or if you want to switch your licensed Object Lock type, contact Cloudian Support.

1.3.5. HyperIQ License

Cloudian HyperIQ is a solution for dynamic visualization and analysis of HyperStore system monitoring data and S3 service usage data. HyperIQ is a separate product available from Cloudian that deploys as virtual appliance on VMware or VirtualBox and integrates with your existing HyperStore system. For more information about HyperIQ contact your Cloudian representative.

Your HyperStore license has a HyperIQ attribute that determines the level of HyperIQ functionality available to you if you acquire and set up the HyperIQ virtual appliance:

- Basic -- HyperIQ dashboards for OS and service status monitoring are supported indefinitely. This is the default.
- Enterprise -- HyperIQ dashboards for OS and service status monitoring are supported indefinitely, and also an S3 analytics dashboard is supported until a defined expiration date. The presence of the S3

analytics dashboard is what distinguishes Enterprise level HyperIQ support from Basic HyperIQ support.

1.3.6. License Updating

You may need to update your license periodically, depending on the specific terms of your Cloudian license agreement. Updating your license requires obtaining a new license file from Cloudian and applying that file on all of your HyperStore nodes. Existing customers can obtain a new license file by emailing a request to cloudian-license@cloudian.com.

Once you've obtained a new license file you can use the CMC's **Information** page to dynamically apply the new license file to your HyperStore system. For instructions see ["Install a New License File" \(page 497\)](#).

1.3.7. Auditing

If you have a production license for HyperStore software, the system will regularly transmit auditing data to Cloudian, Inc., using the system's [Smart Support](#) functionality.

1.4. Nodes, Data Centers, and Regions

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Storage Policies in Multi-DC or Multi-Region Systems" \(page 43\)](#)
- ["Service Architecture in Multi-DC or Multi-Region Systems" \(page 43\)](#)
- ["Deploying HyperStore to Multiple DCs or Regions" \(page 44\)](#)
- ["Using the CMC with Multiple DCs or Regions" \(page 45\)](#)
- ["Using the Admin API in a Multi-Region System" \(page 46\)](#)

The basic units of a HyperStore system are:

- **Node** -- HyperStore software running on a host machine.
- **Data center** -- A physical data center (DC) in which multiple HyperStore nodes are running.
- **Service region** -- A service region is a unified set of HyperStore nodes deployed in one or multiple data centers across a particular geographic area. A region has its own unique S3 service endpoint, its own unique inventory of stored objects, and a unified Cassandra database for storing metadata (the Metadata DB).

The common deployment topologies for a HyperStore system are:

- **Single data center constituting a single region.** This is the simplest deployment topology, where the whole HyperStore system consists of a single data center in which multiple HyperStore nodes are running. The one DC constitutes its own region. The number of nodes can scale from a minimum of three -- the smallest viable size of a HyperStore system -- up to dozens of nodes, all running in one DC. Adding more nodes within a DC is the most common way to add capacity to the system. For more information see ["Capacity Monitoring and Expansion" \(page 656\)](#).
- **Multiple data centers in a single region.** With this topology, HyperStore nodes running in multiple data centers comprise one unified storage system. Typically the motivation for this type of deployment is replication of data across DCs, for the purposes of data protection, service resilience, and/or disaster recovery. This cross-DC replication is configurable through the use of HyperStore storage policies.

- **Multiple service regions.** With this topology the HyperStore system spans multiple service regions, each of which consists of one or more data centers. Each region has its own separate S3 service end-point (to which S3 clients submit requests), its own independent token space, and its own separate inventory of stored objects. In a multi-region HyperStore system, the regions are in most respects separate S3-compatible object storage systems -- with the significant exceptions that the same population of authorized end users has access to all the service regions, and that HyperStore supports unified administration across the multiple regions. Typically the motivation for having multiple service regions is to allow users to choose one geographic region or another for storing their data, for reasons of proximity or regulatory compliance.

For a diagram showing the relation between nodes, data centers, and service regions see ["System Levels" \(page 70\)](#).

1.4.1. Storage Policies in Multi-DC or Multi-Region Systems

In a **multi-DC, single-region** HyperStore system, configurable storage policies determine whether and how data is replicated across DCs. You can have multiple storage policies -- for example you could have one storage policy that stores data only in one particular DC, and another storage policy that replicates data so that copies of each object are stored in each of your DCs. When users create a bucket they choose which storage policy will apply to data uploaded to that bucket.

In a **multi-region** system, each region has its own set of storage policies, and each region's storage policies operate only within that region. As noted previously, each region is essentially an independent storage system with its own inventory of objects -- and its own storage policies for distributing data within the region.

For more information about storage policies, including details about multi-DC storage policies, see ["Storage Policies Feature Overview" \(page 148\)](#).

For information about a supported option for asynchronously replicating data from a bucket in one region to a different bucket in a different region, see ["Cross-Region Replication Feature Overview" \(page 213\)](#).

1.4.2. Service Architecture in Multi-DC or Multi-Region Systems

1.4.2.1. Multi-DC, Single Region System

Along with common services that run on every node the system (such as the S3 Service, the HyperStore Service, and the Metadata DB), HyperStore also includes specialized services that run on only one or a few nodes. If you deploy HyperStore across multiple DCs in a service region, the system automatically allocates the specialized services appropriately. For example, each DC will have two nodes acting as Credentials DB slave nodes and each DC will have one node acting as a QoS DB slave node.

For a summary of services and how they are allocated, see ["HyperStore Services Overview" \(page 61\)](#).

For a diagram showing typical services distribution in a multi-DC region, see ["Services Distribution -- Multi-DC, Single Region" \(page 73\)](#).

Also, in a multi-DC region each DC has its own sub-set of HyperStore nodes that are automatically configured to act as internal NTP servers. For more information see ["NTP Automatic Set-Up" \(page 857\)](#).

1.4.2.2. Multi-Region System

A multi-region HyperStore system has the following characteristics:

- The **first service region that you installed** acts as the **default region**. The default region plays several roles in a multi-region system. For example:
 - If service users do not specify a region when they create a new S3 storage bucket, the system will create the bucket in the default region.
 - Only the Admin Service instances in the default service region support the full Admin API.
- Each region has its own S3 service endpoint (URI used by client applications for HTTP access).
- Each region has its own independent object storage inventory and by default there is no object replication across regions (although there is an option for [cross-region replication](#) on a bucket-to-bucket basis).
- When users create a new bucket they choose which region the bucket will be created in.
- User access credentials are valid across the system as a whole. In support of user authentication, a single, uniform Credentials DB serves the entire multi-region system. There is just one Credentials DB master node for the whole system, and that node is located in your default region. Within each region, there are two Credentials DB slave nodes per data center.
- Quality of service (QoS) controls are implemented separately in each region. The QoS limits that you establish for a service region will be applied only to user activity in that particular region. In support of QoS implementation, each region has its own independent QoS DB. Each regional QoS DB has its own master node. In each region there is also one QoS DB slave node per data center.
- The Redis Monitor application will monitor the Credentials DB and QoS DBs in all the regions (and if necessary trigger failover of the master role within each database). One primary Redis Monitor application instance serves the whole multi-region system, and if the primary Redis Monitor instance goes down the backup instance takes over. The primary Redis Monitor instance and backup Redis Monitor instance are on separate nodes in your default region.
- Group and user profile information is stored only in the default region, and is accessed there by services in the other regions. Group and user information is stored only in the Metadata DB in the default region. HyperStore services in non-default regions access the Metadata DB in the default region to retrieve this group and user information as needed.
- Just one Config Controller node is used to propagate system configuration settings throughout the whole multi-region system, during system installation and for ongoing system configuration management.

For a diagram showing typical services distribution in a multi-region system, see ["Services Distribution -- Multi-Region"](#) (page 74).

1.4.3. Deploying HyperStore to Multiple DCs or Regions

The recommended way to arrive at a multi-DC or multi-region HyperStore system is to initially install and use HyperStore in a single DC constituting a single region; and then later in the life of the system, to expand the system by adding a DC or a region. The CMC supports adding a new DC to an existing region (installing HyperStore software on host machines in a new data center and updating the system configuration to reflect the addition of the new DC) or adding a new region to the system (installing HyperStore software on host machines in one or more new data centers and updating the system configuration to reflect the addition of the new service region and data center[s]). For instructions see:

- ["Adding a Data Center"](#) (page 590)
- ["Adding a Region"](#) (page 597)

1.4.4. Using the CMC with Multiple DCs or Regions

The Cloudian Management Console (CMC) facilitates unified administration of a multi-DC or multi-region HyperStore system.

1.4.4.1. Multi-DC, Single Region System

In the CMC, in most respects a multi-DC deployment within a service region is presented as one integrated storage system. For example, performance statistics such as S3 transactions per second and S3 bytes throughput per second are reported for the service region as a whole. However the CMC also is data center aware and provides visibility into the individual DCs within a service region. For example:

- The [Capacity Explorer](#) page lets you see how much free storage capacity remains in each DC (as well as in each node, and also in the service region as a whole).
- The [Data Centers](#) page shows you your node inventory in each DC and provides summary status information for each node. From this page you can also add nodes to your system on a per-DC basis.
- The [Object Locator](#) page lets you see exactly where all of a specified object's replicas or erasure coded fragments are located (on which nodes, in which DC)

1.4.4.2. Multi-Region System

If your HyperStore deployment is set up as a multi-region system, that will be reflected in these aspects of the CMC interface:

- User Provisioning and Administration
 - When you **"Add a User"** (page 401), you will assign the user a rating plan (for billing purposes) for each region. You can assign the same rating plan in each region, or different rating plans in each region. The same is true when you **"Add a Group"** (page 416): when you set a group-level rating plan (which serves as the default rating plan for users within the group), you choose a rating plan for each region.
 - When you **"Set Quality of Service (QoS) Limits"** (page 430), you will assign a different set of limits for each region. QoS limits are applied on a per-region basis.
- S3 Data Storage and Access
 - When service users **"Add a Bucket"** (page 307), they choose which region to create the bucket in.
- Usage Reporting and Billing
 - When you **"Create a Usage Report"** (page 300) for a user, a user group, or the system, you can choose the region for which to report usage. You can also generate reports that includes usage from all regions.
 - When you use the [Account Activity](#) page to generate a statement for billing a user, you generate a separate bill for each region.
- HyperStore System Monitoring
 - When you use the [Data Centers](#) page to get a high level view of each data center's status, you choose which region's data centers you want to check on before choosing the data center.
 - When you use the [Node Status](#) page to check the status detail for individual nodes, you start by selecting a region, and then select a node within that region.

- HyperStore System Management
 - When you [perform management operations on individual nodes](#) (such as cleanup or repair), you first select a region, then select a node from within that region, then select the operation to perform on that node.
 - When you are ["Adding Nodes"](#) (page 579) to the system, you specify the region and data center in which to add it.

1.4.5. Using the Admin API in a Multi-Region System

If your HyperStore system has multiple regions, then:

- For some Admin API calls you can optionally use a "region" URI parameter to indicate that you want the operation applied to a particular region. For example, the syntax for retrieving a user's rating plan is:

```
GET /user/ratingPlan?userId=string&groupId=string[&region=string] HTTP/1.1
```

For such API calls, if you do not specify a region then the default region is presumed.

- Certain Admin API calls are only supported by the Admin Service in the default region. If you submit these calls to the Admin Service in a non-default region, you will receive a 403: Forbidden response. For more information see [Admin API Behavior in Multi-Region Systems](#). For more information see the Introduction section in the *Cloudian HyperStore Admin API Reference*.

1.5. Hosts and Network

1.5.1. Host Hardware and Operating System

Subjects covered in this section:

- ["Hardware Recommendations"](#) (page 46)
- ["Operating System Requirements"](#) (page 47)

These host recommendations and requirements apply if you are doing a fresh installation of a HyperStore system or if you are adding new nodes to an existing HyperStore system.

1.5.1.1. Hardware Recommendations

The table below shows the recommended specifications for individual host machines in a HyperStore system.

Component	Recommended Server Spec		Recommended VM Spec	
	Description	Qty	Description	Qty
Processors	Intel Xeon 5218R or better	1-2	20 core CPU or greater	1-2
RAM	128GB or greater	4-12 DIMM slots	128GB or greater	1
OS Drives	NVMe Storage 480GB or greater	2	NVMe Storage 480GB or greater	2
Flash Metadata Drives	NVMe Storage 1920GB or greater	2	SAS SSD / NVMe Storage 1920GB or greater	2

Component	Recommended Server Spec		Recommended VM Spec	
	Description	Qty	Description	Qty
Object Data Capacity Drives	7200RPM SAS 512e HDD or PCIe4 NVMe Flash/SSD	4-45	7200RPM SAS 512e HDD or PCIe4 NVMe Flash/SSD	4-12
Networking	Dual-port 10/25GbE	1-2	Dual-port 10/25GbE	1-2

Note that:

- For production environments the recommended specifications in the table are a starting point. For guidance on scaling a HyperStore deployment to meet your workload requirements, consult with your Cloudian sales representative.
- HyperStore supports using **Quad-Level Cell (QLC) SSDs** as the object data capacity drives, so long as those drives are large capacity drives (15TB or higher). Consult with your Cloudian sales representative for details about the supported QLC SSD types and sizes. QLC SSDs are **not** recommended for using as the metadata drives, since these will be subject to very frequent writes of small files. By contrast, QLC SSDs are best suited for read-heavy workloads and large objects. Note that if you are using QLC SSDs as your object data drives:
 - It is recommended that you have two non-QLC NVMe drives dedicated to HyperStore metadata storage, as indicated in the table above.
 - When you launch the HyperStore installer (as described in [Installing a New HyperStore System](#)) use the `configure-qlc-rdb` command line option.
- For virtualization environments, running HyperStore on VMware ESXi and vSphere is supported so long as the VMs have specs meeting or exceeding those in the table. However, avoid KVM or Xen as there are known problems with running HyperStore in those virtualization environments. For more guidance on deploying HyperStore on VMware, ask your Cloudian representative for the "Best Practices Guide: Virtualized Cloudian HyperStore on VMware vSphere and ESXi".

1.5.1.2. Operating System Requirements

To perform a fresh installation of HyperStore 8.2, on each of your host machines the operating system must be **Rocky Linux 8.9 or 8.10** (8.10 is recommended). Alternatively your hosts can be running Red Hat Enterprise Linux 8.9 or 8.10. HyperStore 8.2 does **not** support fresh installation on:

- Other versions of Rocky Linux or RHEL
- CentOS
- Other types of Linux distribution
- Non-Linux operating systems

If you have not already done so, for each host machine on which you intend to install HyperStore, install Rocky Linux 8.9 or 8.10 (or RHEL 8.9 or 8.10) in accordance with your hardware manufacturer's recommendations.

Below, see these additional requirements related to host systems on which you intend to install HyperStore:

- ["Partitioning of Disks Used for the OS and Metadata Storage"](#) (page 48)
- ["Host Firewall Services Must Be Disabled"](#) (page 48)
- ["Do Not Mount /tmp Directory with 'noexec'"](#) (page 49)

- ["root User umask Must Be 0022"](#) (page 49)
- ["Note: Automatic Exclusions to OS Package Updates"](#) (page 49)

Partitioning of Disks Used for the OS and Metadata Storage

For the disks used for the OS and metadata storage -- typically two mirrored SSDs as noted in the hardware requirements table above -- **do not accept the default partition schemes offered by Rocky Linux / RHEL:**

- By default Rocky Linux / RHEL allocates a large portion of disk space to a */home* partition. This will leave inadequate space for HyperStore metadata storage.
- By default Rocky Linux / RHEL proposes using LVM. Cloudbian recommends using standard partitions instead.

Cloudbian recommends that you manually create a partition scheme like this:

For Software RAID

- 1x 1G as */boot*, Device Type RAID1, label *boot*, fs: *ext4*
- 1x 8G as SWAP, Device Type RAID1, label *swap*
- 1x remaining space as */*, Device Type RAID 1, label *root*, fs: *ext4*

For Hardware SUDO RAID with UEFI

- 1x 1G as */boot/efi*, label *efi*
- 1x 1G as */boot*, label *boot*, fs: *ext4*
- 1x 8G as SWAP, label *swap*
- 1x remaining space as */*, label *root*, fs: *ext4*

Host Firewall Services Must Be Disabled

To install HyperStore the following services **must be disabled on each HyperStore host machine:**

- *firewalld*
- *iptables*
- *SELinux*

To disable *firewalld*:

```
# systemctl stop firewalld
# systemctl disable firewalld
```

Rocky Linux / RHEL uses *firewalld* by default rather than the *iptables* service (*firewalld* uses *iptables* commands but the *iptables* service itself is not installed on Rocky Linux / RHEL by default). So you do not need to take action in regard to *iptables* unless you installed and enabled the *iptables* service on your hosts. If that's the case, then disable the *iptables* service.

To disable **SELinux**, edit the configuration file */etc/selinux/config* so that *SELINUX=disabled*. Save your change and then restart the host.

HyperStore includes a built-in firewall service (a HyperStore-custom version of the *firewalld* service) that is configured to protect HyperStore internal services while keeping HyperStore public services open. For more information see the "HyperStore Firewall" section in the *Cloudbian HyperStore Administrator's Guide*.

Note For information about HyperStore port usage see ["HyperStore Listening Ports"](#) (page 57).

Do Not Mount `/tmp` Directory with 'noexec'

The `/tmp` directory on your host machines must not be mounted with the 'noexec' option. If the `/tmp` directory is mounted with 'noexec', you will not be able to extract the HyperStore product package file and the HyperStore installer (installation script) will not function properly.

root User umask Must Be 0022

On hosts on which you will install HyperStore, the `root` user umask value must be '0022' (which is the default on Linux hosts). If the `root` user umask is other than '0022' the HyperStore installation will abort.

Note: Automatic Exclusions to OS Package Updates

As part of HyperStore installation, the HyperStore installation script will install prerequisites including Java and Salt on your HyperStore host machines. If you subsequently use `dnf` to update your OS packages, HyperStore automatically excludes Java and Salt related packages from the update. This is to ensure that only the correct, tested versions of these packages are used together with HyperStore. After HyperStore installation, this auto-exclusion is configured in the `/etc/dnf/plugins/versionlock.list` file on your host machines. You can review that file if you wish to see specifically which packages are "locked" at which versions, but do not remove any entries from the lock list.

1.5.2. File System Requirements

Subjects covered in this section:

- *Introduction (immediately below)*
- ["OS/Metadata Drives and Data Drives" \(page 49\)](#)
- ["Mount Point Naming Guidelines" \(page 50\)](#)
- ["Option for Putting the Metadata DB on Dedicated Drives Rather Than the OS Drives" \(page 50\)](#)
- ["You Must Use UUIDs in fstab" \(page 51\)](#)
- ["A Data Directory Mount Point List \(fstab.txt\) Is Required" \(page 52\)](#)
- ["Reducing Reserved Space to 0% for HyperStore Data Disks" \(page 52\)](#)

Cloudian recommends that you use the HyperStore `system_setup.sh` tool to configure the disks and mount points on your HyperStore nodes, as described in [Configuring Network Interfaces, Time Zone, and Data Disks](#). The tool is part of the HyperStore product package (when you extract the `.bin` file).

If you do not use the system setup tool for disk setup, use the information below to make sure that your hosts meet HyperStore file system requirements.

1.5.2.1. OS/Metadata Drives and Data Drives

Although it's possible to install HyperStore on a host with just a single hard drive, for a rigorous evaluation or for production environments each host should have multiple drives (see ["Host Hardware and Operating System" \(page 46\)](#)). On host machines with multiple hard drives:

- HyperStore will by default use the drive that the OS is on for storing system metadata (in the Metadata DB, the Credentials DB, and the QoS DB). Cloudian recommends that you dedicate two drives to the OS and system metadata in a RAID-1 mirroring configuration. Preferably the OS/metadata drives should be SSDs. Also preferably, the metadata should be on dedicated drives as described in ["Option for Putting the Metadata DB on Dedicated Drives Rather Than the OS Drives" \(page 50\)](#)

- You must format all other available hard drives with `ext4` file systems mounted on raw disks. These drives will be used for storing S3 object data. RAID is not necessary on the S3 object data drives.

For example, on a machine with 2 SSDs and 12 HDDs:

- Mirror the OS on the two SSDs. For more detailed recommendations for partitioning these disks see ["Partitioning of Disks Used for the OS and Metadata Storage"](#) (page 48).
- Format each of the 12 HDDs with `ext4` file systems and configure mount points such as `/cloudian1`, `/cloudian2`, `/cloudian3` and so on.

Note On the HDDs for storing object data, HyperStore **does not support** XFS file systems; VirtIO disks; or Logical Volume Manager (LVM). For questions regarding these unsupported technologies, contact Cloudian Support.

1.5.2.2. Mount Point Naming Guidelines

If you are installing HyperStore on multiple hosts that each have multiple disks for object data storage, use the same mount point naming scheme on each of your hosts. If all your hosts have the same number of disks, then they should all have the identical set of mount points for HyperStore object storage. For example, if each host has 12 disks for object storage, then on all your hosts you could name the mount points `/cloudian1`, `/cloudian2`, `/cloudian3`, and so on up through `/cloudian12`.

If in your installation cluster some hosts have more disks than others, use as much overlap in mount point naming as possible. For example, suppose that most of your hosts have 10 disks for storing object data while one host has 12 disks. In this scenario, all of the hosts can have mount points `/cloudian1`, `/cloudian2`, `/cloudian3`, and so on up through `/cloudian10`, while the one larger host has those same mount points plus also `/cloudian11` and `/cloudian12`.

Note Although uniformity of mount point naming across nodes (to the extent possible) is desirable for simplicity's sake, the HyperStore installation does support a way to accommodate differences in the number or names mount points across nodes -- this is described in ["A Data Directory Mount Point List \(fslist.txt\) Is Required"](#) (page 52)..

1.5.2.3. Option for Putting the Metadata DB on Dedicated Drives Rather Than the OS Drives

Regarding the Metadata DB (built on Cassandra), the recommended configuration is to put your Cassandra data on dedicated drives, rather than on the OS drives. In this case you would have:

- OS drives in RAID-1 configuration. The Credentials DB and QoS DB will also be written to these drives.
- Cassandra drives in RAID-1 configuration. On these drives will be written Cassandra data and also the Cassandra commit log. (And also the Digest DB commit log, in HyperStore 8.2 and later.)

Note You must create a Cassandra data directory named as `<mountpoint>/cassandra` (for example `cassandradb/cassandra`) and a Cassandra commit log directory named as `<mountpoint>/cassandra_commit` (for example `cassandradb/cassandra_commit`).

- Multiple drives for S3 object data (with mount points for example `/cloudian1`, `/cloudian2`, `/cloudian3` and so on), with no need for RAID protection.

For more information see ["A Data Directory Mount Point List \(fslist.txt\) Is Required"](#) (page 52).

Note If you are using **Quad-Level Cell (QLC) SSDs**, these should be used as the object data drives -- **not** as the metadata drives. The metadata drives will be subject to very frequent writes of small files. By contrast, QLC SSDs are best suited for read-heavy workloads and large objects.

1.5.2.4. You Must Use UUIDs in `fstab`

In your `fstab` file, **you must use UUIDs** to identify the devices to which you will mount HyperStore S3 object data directories. Do not use device names or LABELs.

If you are not using UUIDs in `fstab` currently, follow the instructions below to modify your `fstab` so that it uses UUIDs for the devices to which you will mount S3 object data directories (you do not need to do this for the OS/-metadata mount points).

As `root`, do the following:

1. Check whether your `fstab` is currently using UUIDs for your S3 object data drives. In the example below, there are two S3 object data drives and they are currently identified by device name, not by UUID.

```
# cat /etc/fstab
...
...
/dev/sdb1 /cloudian1 ext4 rw,noatime,barrier=0,data=ordered,errors=remount-ro 0 1
/dev/sdc1 /cloudian2 ext4 rw,noatime,barrier=0,data=ordered,errors=remount-ro 0 1
```

2. Back up your existing `fstab` file:

```
# cp /etc/fstab /etc/fstab.backup.<today's date>
```

3. Retrieve the UUIDs for your devices by using the `blkid` command.

```
# blkid
...
...
/dev/sdb1: UUID="a6fed29c-97a0-4636-afa9-9ba23e1319b4" TYPE="ext4"
/dev/sdc1: UUID="rP38Ux-3wzO-sP3Y-2CoD-2Tdu-fjpO-ffPFZV" TYPE="ext4"
```

4. Open `fstab` in an editor.
5. For each device that you are using for S3 object storage, replace the device name with `UUID=<UUID>`, copying the device's UUID from the `blkid` response in the previous step. For example:

```
# Original line

/dev/sdb1 /cloudian1 ext4 rw,noatime,barrier=0,data=ordered,errors=remount-ro 0 1

# Revised line

UUID="a6fed29c-97a0-4636-afa9-9ba23e1319b4" /cloudian1 ext4 rw,noatime,barrier=0,
data=ordered,errors=remount-ro 0 1
```

6. After editing `fstab` so that each device on which you will store S3 data is identified by a UUID, save your changes and close the `fstab` file.

7. Remount the host's file systems:

```
# mount -a
```

Repeat this process for **each host on which you will install HyperStore**.

1.5.2.5. A Data Directory Mount Point List (fslist.txt) Is Required

If you do not use the HyperStore *system_setup.sh* script to configure the data disks and mount points on your nodes, you must manually create a data directory mount point list file and place it in your installation staging directory on the Config Controller node, as described below.

Note If you use the *system_setup.sh* script to configure the disks and mount points on your nodes, the script creates the needed mount point list files automatically and you can ignore the instructions below.

If all your nodes have the same data mount points -- for example if all nodes have as their data mount points */cloudian1*, */cloudian2*, and so on through */cloudian12* -- you only need to create one mount point list file. If some nodes have a different set of mount points than do other nodes -- for example if some nodes have more data disks than other nodes -- you will need to create a default mount point list file and also a node-specific mount point list file for each node that differs from the default.

In your installation staging directory create a file named *fslist.txt* and in the file enter one line for each of your S3 data directory mount points, with each line using the format below.

```
<deviceName> <mountPoint>
```

Example of a properly formatted file (truncated):

```
/dev/sdc1 /cloudian1
/dev/sdd1 /cloudian2
...
```

Note Use device names in your *fslist.txt* file, not UUIDs.

Optionally, you can also include an entry for the Cassandra data directory and an entry for the Cassandra commit log directory, if you do not want this data to be put on the same device as the operating system (see ["Option for Putting the Metadata DB on Dedicated Drives Rather Than the OS Drives"](#) (page 50)). If you do not specify these Cassandra directory paths in *fslist.txt*, then by default the system automatically puts Cassandra data and commit log directories on the same device on which the operating system resides.

Do not use symbolic links when specifying your mount points. The HyperStore system does not support symbolic links for data directories.

If some of your hosts have data directory mount point lists that differ from the cluster default, in the installation staging directory create a *<hostname>_fslist.txt* file for each such host. For example, along with the default *fslist.txt* file that specifies the mount points that most of your hosts use, you could also have a *cloudian-node11_fslist.txt* file and a *cloudian-node12_fslist.txt* file that specify mount points for two non-standard nodes that have hostnames *cloudian-node11* and *cloudian-node12*.

1.5.2.6. Reducing Reserved Space to 0% for HyperStore Data Disks

By default Linux systems reserve 5% of file system space for root user and system services. On modern large-capacity disks this can be a waste of a considerable amount of storage space. Cloudian recommends that you

set the reserved space to 0% for each drive on which you will store HyperStore object data (S3 object data).

For each HyperStore data drive do the following.

```
### Check current "Reserved block count":

# tune2fs -l <device>

### Set Reserved block count to 0%:

# tune2fs -m 0 <device>

### For example:

# tune2fs -m 0 /dev/sdc1
```

1.5.3. DNS Set-Up

Note As an alternative to setting up your DNS system to directly resolve HyperStore service endpoints (as described in this section), you have the option to use HyperStore's integrated nameserver **hsDNS** in combination with DNS delegation. For more information, in the "Service Features" section of your Cloudian HyperStore Administrator's Guide, see "hsDNS".

Subjects covered in this section:

- *Introduction (immediately below)*
- ["HyperStore Service Endpoints" \(page 53\)](#)
- ["Configuring Resolution of Service Endpoints" \(page 55\)](#)
- ["Using Customized Service Endpoints" \(page 55\)](#)

For your HyperStore system to be accessible to external clients, you must configure your DNS name servers with entries for the HyperStore service endpoints. **Cloudian recommends that you complete your DNS configuration prior to installing the HyperStore system.** This section describes the required DNS entries.

1.5.3.1. HyperStore Service Endpoints

HyperStore includes a variety of services each of which is accessible to clients by way of a web service endpoint. On your name servers you will need to configure a DNS entry for each of these service endpoints.

By default the HyperStore system uses a standard format for each service endpoint, building on two values that are specific to your environment:

- Your organization's domain (for example *enterprise.com*)
- The name or names of your HyperStore service region or regions (for example *boston* for a single-region system, or *boston* and *chicago* for a multi-region system). Only lower case alphanumeric characters and dashes are allowed in region names, the minimum allowed region name length is 3 characters and the maximum length is 52 characters.

During HyperStore installation you will supply your domain and your service region name(s), and the interactive installer will show you the default service endpoints derived from the domain and region name(s). During installation you can accept the default endpoints or specify custom endpoints instead. The table that follows below is based on the default endpoint formats.

Note

* Do not include the string "s3" in your domain or in your region name(s). By default HyperStore generates S3 service endpoints by prepending an "s3-" prefix to your *<regionname>.<domain>* combination. If you include "s3" within either your domain or your region name, this will result in two instances of "s3" in the system-generated S3 service endpoints, and this may cause S3 service requests to fail for some S3 clients.

* If you specify custom endpoints during installation, do not use IP addresses in your endpoints.

* HyperStore by default derives the S3 service endpoint(s) as *s3-<regionname>.<domain>*. However HyperStore also supports the format *s3.<regionname>.<domain>* (with a dot after the "s3" rather than a dash) if you specify custom endpoints with this format during installation.

The table below shows the default format of each service endpoint. The examples show the service endpoints that the system would automatically generate if the domain is *enterprise.com* and the region name is *boston*.

Service Endpoint	Default Format and Example	Description
S3 service endpoint (one per service region)	<i>s3-<regionname>.<domain></i> <i>s3-boston.enterprise.com</i>	This is the service endpoint to which S3 client applications will submit requests. If you are installing a HyperStore system across multiple service regions, each region will have its own S3 service endpoint, and therefore you must create a DNS entry for each of those region-specific endpoints — for example <i>s3-boston.enterprise.com</i> and <i>s3-chicago.enterprise.com</i> .
S3 service endpoint wildcard (one per service region)	<i>*.s3-<regionname>.<domain></i> <i>*.s3-boston.enterprise.com</i>	This S3 service endpoint wildcard entry is necessary to resolve virtual-hosted-style S3 requests, wherein the bucket name is specified as a sub-domain -- for example <i>bucket1.s3-boston.enterprise.com</i> and <i>bucket2.s3-boston.enterprise.com</i> and so on.
S3 static website endpoint (one per service region)	<i>s3-website-<regionname>.<domain></i> <i>s3-website-boston.enterprise.com</i>	This S3 service endpoint is used for buckets configured as static websites. Note Your S3 static website endpoint cannot be the same as your S3 service endpoint. They must be different, or else the website endpoint will not work properly.
S3 static website endpoint wildcard (one per service region)	<i>*.s3-website-<regionname>.<domain></i> <i>*.s3-website-boston.enterprise.com</i>	This S3 static website endpoint wildcard entry is necessary to resolve virtual-hosted-style S3 requests, wherein the bucket name is specified as a sub-domain, for buckets configured as static websites.
Admin Service endpoint	<i>s3-admin.<domain></i> <i>s3-admin.enterprise.com</i>	This is the service endpoint for HyperStore's Admin API. The Cloudbian Management Console accesses this API, and you can also

Service Endpoint	Default Format and Example	Description
(one per entire system)		access this API directly with a third party client (such as a command line tool like <i>cURL</i>).
IAM Service endpoint (one per entire system)	<i>iam.<domain></i> <i>iam.enterprise.com</i>	This is the service endpoint for accessing HyperStore's implementation of the Identity and Access Management API. Note The IAM service endpoint can also be used for making Security Token Service (STS) API calls.
SQS Service endpoint (one per entire system)	<i>sqs.<domain></i> <i>sqs.enterprise.com</i>	This is the service endpoint for accessing HyperStore's implementation of the Simple Queue Service (SQS) API. Note The SQS Service is disabled by default. For information about enabling this service, see the SQS section of the <i>Cloudian HyperStore AWS APIs Support Reference</i> .
Cloudian Management Console (CMC) endpoint (one per entire system)	<i>cmc.<domain></i> <i>cmc.enterprise.com</i>	The CMC is HyperStore's web-based console for performing system administrative tasks. The CMC also supports actions such as creating storage buckets or uploading objects into buckets.

1.5.3.2. Configuring Resolution of Service Endpoints

IMPORTANT ! Cloudian Best Practices suggest that a highly available load balancer be used in production environments where consistent performance behavior is desirable. For environments where a load balancer is unavailable, other options are possible. Please consult with your Cloudian Sales Engineer for alternatives.

For a production environment, in your DNS configuration each HyperStore service endpoint should resolve to the virtual IP address(es) of two or more load balancers that are configured for high availability. For more detail see "[Load Balancing](#)" (page 56).

1.5.3.3. Using Customized Service Endpoints

If you do not want to use the default service endpoint formats, the HyperStore system allows you to specify custom endpoint values during the installation process. If you intend to create custom endpoints, configure DNS entries to resolve the custom endpoint values that you intend to use, rather than the default-formatted endpoint values shown in the "[HyperStore Service Endpoints](#)" (page 53) table. Make a note of the custom endpoints

for which you configure DNS entries, so that later you can correctly specify those custom endpoints when you perform the HyperStore installation.

If you want to use a **custom S3 endpoint** that does not include a region string, the installer allows you to do so. Note however that if your S3 endpoints lack region strings the system will not be able to support the region name validation aspect of AWS Signature Version 4 authentication for S3 requests (but requests can still succeed without the validation).

If you want to use **multiple S3 endpoints per service region** -- for example, having different S3 endpoints resolve to different data centers within one service region -- the installer allows you to do this. For this approach, the recommended syntax is `s3-<regionname>.<dcname>.<domain>` -- for example `s3-boston.dc1.enterprise.com` and `s3-boston.dc2.enterprise.com`.

Note If you want to change HyperStore service endpoints after the system has already been installed, you can do so as described in the "Changing S3, Admin, CMC, or IAM Service Endpoints" section of the *Cloudian HyperStore Administrator's Guide*. If you change any endpoints, be sure to update your DNS configuration.

1.5.4. Load Balancing

IMPORTANT ! Cloudian recommends that a highly available load balancer be used in production environments where consistent performance behavior is desirable. For environments where a load balancer is unavailable, other options are possible. Please consult with your Cloudian Sales Engineer for alternatives. The discussion below assumes that you are using a load balancer.

HyperStore uses a peer-to-peer architecture in which each node in the cluster can service requests to the S3, Admin, CMC, IAM, STS, and SQS service endpoints. In a production environment you should use load balancers to distribute S3, Admin, CMC, IAM, STS, and SQS service endpoint requests evenly across all the nodes in your cluster. In your DNS configuration the S3, Admin, CMC, IAM, STS, and SQS service endpoints should resolve to the virtual IP address(es) of your load balancers; and the load balancers should in turn distribute request traffic across all your nodes. Cloudian recommends that you **set up your load balancers prior to installing the HyperStore system**.

For high availability it is preferable to use two or more load balancers configured for failover between them (as versus having just one load balancer which would then constitute a single point of failure). The load balancers could be commercial products or you can use open source technologies such as [HAProxy](#) (load balancer software for TCP/HTTP applications) and [Keepalived](#) (for failover between two or more load balancer nodes). If you use software-defined solutions such as these open source products, for best performance you should install them on dedicated load balancing nodes -- not on any of your HyperStore nodes.

For a **single-region HyperStore system**, for each service configure the load balancers to distribute request traffic across all the nodes in the system.

For a **multi-region HyperStore system**:

- Configure each region's S3 service endpoint to resolve to load balancers in that region, which distribute traffic across all the nodes within that region.
- Configure the Admin, IAM, STS, SQS, and CMC service endpoints to resolve to load balancers in the **default service region**, which distribute traffic to all the nodes in the default service region. (You will specify a default service region during the HyperStore installation process. For example, you might

have service regions *boston* and *chicago*, and during installation you can specify that *boston* is the default service region.)

For detailed guidance on load balancing set-up, request a copy of the ***HyperStore Load Balancing Best Practice Guide*** from your Clouidian Sales Engineering representative.

Note The HyperStore S3 Service supports **PROXY Protocol** for incoming connections from a load balancer. This is disabled by default, but after HyperStore installation is complete you can enable it by configuration if you wish. For more information search on "s3.enable.proxyProtocol" in the ***Clouidian HyperStore Administrator's Guide***.

Note For information about how to perform health checks of HyperStore's HTTP(S) services such as the S3 Service and the CMC, see the "Checking HTTP(S) Responsiveness" section in the ***Clouidian HyperStore Administrator's Guide***.

1.5.5. HyperStore Listening Ports

The HyperStore system uses the listening ports specified in the table below. Only the service ports for the CMC, S3, IAM, SQS, and Admin services -- the port numbers in italics in the "Listening Port" column -- should be open to traffic originating from outside the HyperStore system. All other ports must be closed to traffic from outside the system, for system security.

Each HyperStore node includes a built-in HyperStore Firewall that implements port restrictions appropriate to a HyperStore cluster. The HyperStore Firewall is disabled by default in HyperStore systems that were originally installed as a version older than 7.2; and enabled by default in HyperStore systems that originally installed as version 7.2 or newer. You can enable/disable the firewall on all HyperStore nodes by using the installer's Advanced Configuration Options. For instructions see "HyperStore Firewall" in the ***Clouidian HyperStore Administrator's Guide***.

Note If you are installing HyperStore across **multiple data centers and/or multiple service regions**, the HyperStore nodes in each data center and region will need to be able to communicate with the HyperStore nodes in the other data centers and regions. This includes services that listen on the internal interface (such as Cassandra, the HyperStore Service, and Redis). Therefore you will need to configure your networking so that the internal networks in each data center and region are connected to each other (for example, by using a VPN).

Service	Listening Port	Interface(s) Bound To	Purpose
Clouidian Management Console (CMC)	<i>8888</i>	All	Requests from administrators' or end users' browsers over HTTP
	<i>8443</i>	All	Requests from administrators' or end users' browsers over HTTPS
S3 Service	<i>80</i>	All	Requests from the CMC or other S3 client applications over HTTP
	<i>443</i>	All	Requests from the CMC or

Service	Listening Port	Interface(s) Bound To	Purpose
			other S3 client applications over HTTPS
	81	All	Requests relayed by an HAProxy load balancer using the PROXY Protocol
	4431	All	Requests relayed by an HAProxy load balancer using the PROXY Protocol with SSL (if enabled by configuration)
	19080	Internal	JMX access
IAM Service and STS Service	16080	All	Requests from the CMC or other Identity and Access Management (IAM) or Security Token Service (STS) clients over HTTP Note In the current HyperStore release, the STS Service uses the same listening ports as the IAM Service.
	16443	All	Requests from the CMC or other IAM or STS clients over HTTPS
	19084	Internal	JMX access
SQS Service	18090	All	Requests from Simple Queue Service (SQS) clients over HTTP
	18443	All	Requests from SQS clients over HTTPS
	19085	Internal	JMX access
Admin Service	18081	All	Requests from the CMC or other Admin API clients over HTTP
	19443	All	Requests from the CMC or other Admin API clients over HTTPS (Note: The CMC by default uses HTTPS to access the Admin Service) IMPORTANT ! The Admin Service is inten-

Service	Listening Port	Interface(s) Bound To	Purpose
			ded to be accessed only by the CMC and by system administrators using other types of clients (such as cURL). Do not expose the Admin Service to a public network.
	19081	Internal	JMX access
Redis Monitor	9078	Internal	Communication between primary and backup Redis Monitor instances
	19083	Internal	JMX access
HyperStore Service	19090	Internal	Data operation requests from the S3 Service
	19050	Internal	Communication between HyperStore Service instances
	19082	Internal	JMX access
Credentials DB and QoS DB (Redis)	6379	Internal	Requests to the Credentials DB from the S3 Service, HyperStore Service, or Admin Service; and communication between Credentials DB nodes
	6380	Internal	Requests to the QoS DB from the S3 Service, HyperStore Service, or Admin Service; and communication between QoS DB nodes
Metadata DB (Cassandra)	9042	Internal	Data operations requests from the S3 Service, HyperStore Service, or Admin Service, using CQL protocol
	9160	Internal	Data operations requests from the S3 Service, HyperStore Service, or Admin Service, using Thrift protocol
	7000	Internal	Communication between Cassandra instances
	7199	Internal	JMX access
Cloudian Monitoring Agent	19070	Internal	Requests from the Cloudian Monitoring Data Collector
Config Controller	4505	Internal	On your Config Controller node

Service	Listening Port	Interface(s) Bound To	Purpose
			this is the port to which Salt agents ("minions") establish a persistent connection so that the Salt Master can publish to the minions.
	4506	Internal	Salt minions connect to this port on the Config Controller as needed to send results to the Salt Master, and to request files and minion-specific data values.
SSH	22	All	The HyperStore installer accesses this SSH port on each node on which you are installing HyperStore software (during initial cluster install or if you subsequently expand your cluster)
NTP	123	All	NTP port for time synchronization between nodes
Echo	7	Internal	The Clouddian Monitoring Data Collector uses Echo (port 7) to check whether each node is reachable.
HyperIQ	9999	All	If you use Clouddian HyperIQ to monitor your HyperStore system, HyperIQ accesses port 9999 on each HyperStore node

1.5.6. Outbound Internet Access

The HyperStore installation process does not require outbound internet access. However, the following HyperStore features do access the internet once the system is in operation; and HyperStore does need access to NTP server(s) during the installation (see the "Pre-Configured chronyd" bullet point below). If you use forward proxying in your environment, after HyperStore installation you may want to set up forward proxying to support these HyperStore features:

- **Smart Support** — The Smart Support feature (also known as "Phone Home") securely transmits HyperStore daily diagnostic information to Clouddian Support over the internet. HyperStore supports configuring this feature to use an explicit forward proxy for its outbound internet access (after installation, the relevant settings are `common.phoneHome.proxy.host`, `common.phoneHome.proxy.port`, `common.phoneHome.proxy.auth.username`, and `common.phoneHome.proxy.auth.password`). To use a forward proxy with this feature you should configure your forward proxy to support access to `*.s3-support.clouddian.com` (that is, to any sub-domain of `s3-support.clouddian.com`).
- **Auto-Tiering and Cross-Region Replication** — If you want to use either the auto-tiering feature or the cross-region replication feature (CRR), the S3 Service running on each of your HyperStore nodes

requires outbound internet access. These features do not support configuring an explicit forward proxy, but you can use transparent forward proxying if you wish. (Setting up transparent forward proxying is outside the scope of this documentation.) For more information on these features see the "Auto-Tering Feature Overview" and "Cross-Region Replication Overview" sections in the *Cloudian HyperStore Administrator's Guide*.

- **Pre-Configured *chronyd*** — Accurate, synchronized time across the cluster is vital to HyperStore service. When you install your HyperStore cluster, the installation script automatically configures a robust NTP set-up using *chronyd*. In each of your HyperStore data centers four of your HyperStore nodes are automatically configured to act as internal NTP servers. (If a HyperStore data center has only four or fewer nodes, then all the nodes in the data center are configured as internal NTP servers.) These internal NTP servers are configured to connect to external NTP servers — by default the public servers from the *pool.ntp.org* project. In order to connect to the external NTP servers, the internal NTP servers must be allowed outbound internet access. This feature does not support configuring an explicit forward proxy, but you can use transparent forward proxying if you wish. (Setting up transparent forward proxying is outside the scope of this documentation.)

IMPORTANT ! If you do not allow HyperStore hosts to have outbound connectivity to the internet, then during the interactive installation process -- when you are prompted to specify the NTP servers that HyperStore hosts should connect to -- you must specify NTP servers within your environment, rather than the public NTP servers that HyperStore connects to by default. If HyperStore hosts cannot connect to any NTP servers, the installation will fail.

After HyperStore installation, to see which of your HyperStore nodes are internal NTP servers, log into the CMC and go to **Cluster** → **Cluster Config** → **Cluster Information**. On that CMC page you can also see your configured list of NTP servers.

For more information on HyperStore's NTP set-up, see the "NTP Automatic Set-Up" section in the *Cloudian HyperStore Administrator's Guide*.

1.5.6.1. Multi-DC Considerations

If you are installing HyperStore across multiple data centers and/or multiple service regions, the HyperStore nodes in each data center and region will need to be able to communicate with the HyperStore nodes in the other data centers and regions. This includes services that listen on the internal interface (such as Cassandra, the HyperStore Service, and Redis). Therefore you will need to configure your networking so that the internal networks in each data center and region are connected to each other (for example, by using a VPN). See "[HyperStore Listening Ports](#)" ([page 57](#)) for HyperStore requirements regarding listening port access.

1.6. HyperStore Services

1.6.1. HyperStore Services Overview

The Cloudian HyperStoreTM system is composed of several types of services each of which plays a role in implementing the overall HyperStore object storage service. The table below shows the major HyperStore services and how these services are distributed in a multi-node cluster. There are common services that run on every node, and specialized support services that run on only one or a sub-set of nodes.

For services distribution diagrams, see ["Services Distribution -- 3 Nodes, Single DC" \(page 72\)](#). For information on starting and stopping services, see ["Starting, Stopping, and Checking Status of Services" \(page 91\)](#).

Service Category	Service	Where It Runs
Common Services	S3 Service	Every node.
	HyperStore Service	Every node.
	Metadata DBService	Every node.
	Admin Service	Every node.
	IAM Service	Every node in the default service region.
	STS Service	Every node in the default service region.
	SQS Service	Every node.
	Cloudian Management Console (CMC)	Every node.
Specialized Support Services	Credentials DB Master	One node per entire HyperStore system.
	Credentials DB Slaves	Two per data center. If you have a large cluster (25 nodes or more in a data center), consult with Cloudian Support about whether you should add more Credentials DB slaves. For instructions on adding slaves, see "Move or Add a Credentials DB Slave or QoS DB Slave" (page 617) . Slaves will not be on same node as the Credentials DB master
	QoS DB Master(s)	One node per service region.
	QoS DB Slave(s)	One node per data center. Slave will not be on same node as the QoS DB Master.
	Redis Monitor	One primary node and one backup node per entire HyperStore system.
	Crontab configuration and Monitoring Data Collector	One primary node and one backup node per service region.

Service Category	Service	Where It Runs
Auxiliary Services	Local NTP server	One local NTP server per service region.
	Config Controller	One primary node and one backup node per entire HyperStore system.
	HyperStore File Service	Optional service that can be installed on auxiliary nodes after the core HyperStore system is installed.
	HyperStore Search Service	Optional service that can be installed on auxiliary nodes after the core HyperStore system is installed.

Note Within a newly installed cluster, the HyperStore installer automatically chooses the hosts for services that are not intended to run on every node. These host assignments are recorded to an installation configuration file that the installer generates when it runs (*CloudianInstallConfiguration.txt* in your installation staging directory). After your installation is completed, these host assignments can also be viewed on the CMC's **Cluster Information** page (**Cluster -> Cluster Config -> Cluster Information**). If you want to modify these assignments after install, see ["Change Node Role Assignments"](#) (page 616).

1.6.2. Clodian Management Console (CMC) Service

The Clodian Management Console (CMC) is a web-based user interface for Clodian HyperStore system administrators, group administrators, and end users. The functionality available through the CMC depends on the user type associated with a user's login ID (system admin, group admin, or regular user).

As a HyperStore system administrator, you can use the CMC to perform tasks such as:

- Monitoring the system and performing system operations
- Provisioning groups and users
- Managing quality of service (QoS) controls
- Creating and managing rating plans
- Generating usage data reports
- Generating bills
- Viewing and managing users' stored data objects (if you've configured the system to allow this)

Group administrators can perform a more limited range of admin tasks pertaining to their own group, and can also perform S3 operations such as creating and configuring buckets and uploading and downloading objects. Regular users can only perform S3 operations such as creating and configuring buckets and uploading and downloading objects.

The CMC acts as a client to several of the [API Services](#) including the Admin Service and the S3 Service.

1.6.3. API Services

1.6.3.1. Admin Service

The HyperStore Admin Service implements a RESTful HTTP API through which you can perform administrative operations such as:

- Provisioning groups and users
- Managing quality of service (QoS) controls
- Creating and managing rating plans
- Generating usage data reports
- Generating bills
- Retrieving system metrics and monitoring data

The ["Cloudian Management Console \(CMC\) Service"](#) (page 63) is a client to the Admin Service. You also have the option of using a command line tool such as cURL to submit Admin API commands, or building your own Admin Service client.

For more information about the Admin API see the *Cloudian HyperStore Admin API Reference*.

1.6.3.2. AWS Compliant API Services

The HyperStore **S3 Service** provides comprehensive support for the AWS Simple Storage Service API. (A separate service, the **S3 Select Service**, provides dedicated support for the S3 Select feature. The main S3 Service proxies S3 *SelectObjectContent* requests through to the S3 Select Service for processing.)

The HyperStore **IAM, STS, and SQS Services** provide partial support for the AWS Identity and Access Management API, the AWS Security Token Service API, and the AWS Simple Queue Service API, respectively.

For the S3 Service and IAM Service you can use the ["Cloudian Management Console \(CMC\) Service"](#) (page 63) as a client application, or use a third party or custom client application. For the STS and SQS Services, the CMC does not provide client access and so you must use third party or custom client applications to access these services.

For more information about HyperStore's implementation of these services see the *Cloudian HyperStore AWS APIs Support Reference*.

1.6.4. Database Services

1.6.4.1. Metadata DB

The HyperStore system stores object metadata, user metadata, and system metadata in the Metadata DB. The Metadata DB is built on the Apache open source storage platform [Cassandra](#). The HyperStore system creates and uses several "keyspaces" within Cassandra:

- The **UserData_<policyid>** keyspaces store:
 - User bucket information
 - Object metadata. For an overview of the HyperStore system's support for object metadata, see ["Object Metadata and Search Feature Overview"](#) (page 230).

Note There is one *UserData_<policyid>* keyspace for each storage policy in the system. For information about storage policies see ["Storage Policies Feature Overview"](#) (page 148).

- The **AccountInfo** keyspace stores information about HyperStore S3 user accounts and group accounts (including IAM user and group accounts)
- The **Reports** keyspace stores system-wide, per-group, and per-user S3 usage data, in support of the HyperStore usage reporting functionality. It will also store per-bucket usage data if you [enable per-bucket usage tracking](#).
- The **Monitoring** keyspace stores system monitoring statistics in support of HyperStore's system monitoring functionality. It also stores status information for node operations such as repairs and cleanups; and stores token range maps that are used by the system when you add nodes to your cluster.
- The **ECKeyspace** keyspace does not actually store any erasure coded object data; rather, the HyperStore system creates this keyspace so that the HyperStore erasure coding feature can leverage Cassandra functions for token-based mapping of objects (erasure coded object fragments, in this case) to nodes within the storage cluster.
- The **Notification** keyspace stores bucket notification messages. For more information see the SQS section of the *Cloudian HyperStore AWS APIs Support Reference*.

S3 client applications do not access the Metadata DB directly. Instead, all S3 client access is to the [S3 Service](#), which in turn accesses the Metadata DB in support of S3 operations. The [HyperStore Service](#) and [Admin Service](#) also access the Metadata DB.

1.6.4.2. Credentials DB and QoS DB

The **Credentials DB** stores user credentials and additional S3 operation supporting metadata such as multi-part upload session information and public URL access counters.

The **QoS DB** stores user-level and group-level [Quality of Service](#) settings that have been established by system administrators. The QoS DB is also used to keep count of user requests, so that Quality of Service limits can be enforced by the system.

The Credentials DB and QoS DB are both built on [Redis](#), an open source, in-memory key-value data store optimized for fast performance.

The S3 Service, Admin Service, and HyperStore Service are the clients to the Credentials DB and QoS DB. Communication is through a protocol called Redis Serialization Protocol (RESP).

In a multi-region HyperStore deployment there will be:

- Just one, universal Credentials DB which serves the entire HyperStore deployment.
- A separate, independent QoS DB in each service region

Credentials DB and QoS DB Node Roles

Each Credentials DB and each QoS DB is implemented across two or more nodes, with the nodes playing different roles. These roles are:

- **master** — All write requests from DB clients are implemented on the master node. There is only one master node for each Credentials DB and each QoS DB. In a multi-region HyperStore deployment, the universal Credentials DB has one master node and each regional QoS DB has its own master node.

- **slave** — In each Credentials DB and each QoS DB, data from the DB master node is asynchronously replicated on to one or more slave nodes (at least one slave node per data center). The slave nodes support doing reads for DB clients but not writes. If a master node fails, the master role is automatically failed over to a slave node. This fail-over process is managed by the [Redis Monitor Service](#).

Credentials DB and QoS DB roles are assigned to your HyperStore nodes automatically during installation. For more information on the distribution of these services across the cluster see ["Services Distribution -- 3 Nodes, Single DC"](#) (page 72).

1.6.4.3. Checksum DB

The Checksum DB stores digests (MD5 hashes) of the object data files stored in the HyperStore File System (HSFS). For more information see ["HyperStore Service and the HSFS"](#) (page 66), particularly the section on "File Digests".

1.6.5. HyperStore Service and the HSFS

The Cassandra storage platform that underlies the [Metadata DB](#) provides valuable built-in data management functionality including data partitioning, automatic replication, easy cluster expansion, quorum calculation, and so on. For storing small data items, Cassandra also provides good performance. But as the data size increases, storing data in the Linux file system becomes more efficient than storing it in Cassandra.

The HyperStore system uses a hybrid storage solution where Cassandra is used for storing metadata while the Linux filesystem on Cassandra nodes is used for storing object data. The area of the Linux file system where S3 object data is stored is called the **HyperStore File System (HSFS)**.

The general strategy is that Cassandra capabilities are used to determine the distributed data management information such as the nodes that a specific object's metadata should be written to and the nodes that the object's data should be written to. Then at the storage layer, the metadata is stored in Cassandra and the object data is stored in the HSFS.

Within the HSFS, objects can be stored and protected in either of two ways:

- Replicated storage
- Erasure coded storage

For more information on data storage and protection options, see ["Storage Policies Feature Overview"](#) (page 148).

When the system stores S3 objects, the full path to the objects will be as indicated below:

- For S3 object replicas:

```
<mountpoint>/hsfs/<base62-encoded-vNode-token>/<policyid>/<000-255>/<000-255>/<filename>
```

- For S3 object erasure coded fragments:

```
<mountpoint>/ec/<base62-encoded-vNode-token>/<policyid>/<000-255>/<000-255>/<filename>
```

- The path segments are:
 - The <mountpoint> is one of your HyperStore data mount points as configured by the ["hyper-store.directory.data"](#) (page 766) setting.
 - The *hsfs* or *ec* segment distinguishes replicated data (designated here as "hsfs") from erasure-coded data (designated as "ec").

- The `<base62-encoded-vNode-token>` is a base-62 encoding of the token belonging to the [vNode](#) to which the object replica or erasure coded fragment is assigned.
- The `<policyid>` segment indicates the storage policy used by the S3 storage bucket with which the object is associated.
- The two `<000-255>` segments of the path are based on a hash of the `<filename>`, normalized to a 255*255 number.
- The `<filename>` is a dot-separated concatenation of the object's system-assigned token and a timestamp based on the object's Last Modified Time. The token is an MD5 hash (in decimal format) of the bucket name and object name. The timestamp is formatted as `<UnixTimeMillis><6digitAtomicCounter>-<nodeIPAddrHex>`. The last element of the timestamp is the IP address (in hexadecimal format) of the S3 Service node that processed the object upload request.

Note For objects last modified prior to HyperStore version 6.1, the timestamp is simply Unix time in milliseconds. This was the timestamp format used in HyperStore versions 6.0.x and older.

- Example, for a replicated object named "HyperStoreAdminGuide.pdf":

```
/hyperstore1/hsfs/1L1tEZZCCQwdQBdGel4yNk/c4a276180b0c99346e2285946f60e59c/109/154/55898779481268535726200574916609372181.1487608689783689800-0A320A15
```

In the above example:

- "hyperstore1" is one of the HyperStore data mount points configured for the system
- "hsfs" indicates that the object is a replicated object (not an erasure-coded object)
- "1L1tEZZCCQwdQBdGel4yNk" is the Base-62 encoding of the token belonging to the vNode to which the object instance is assigned
- "c4a276180b0c99346e2285946f60e59c" is the system-generated identifier of the storage policy used by the S3 storage bucket with which the object is associated.
- "109/154" is a hash of the file name, normalized to a 255*255 number.
- "55898779481268535726200574916609372181.1487608689783689800-0A320A15" is the file name. The "55898779481268535726200574916609372181" segment is the object's system-assigned token, in decimal format. The "1487608689783689800-0A320A15" segment is the object's Last Modified Time timestamp, in format `<UnixTimeMillis><6digitAtomicCounter>-<nodeIPAddrHex>`.

Note Presuming that [versioning](#) is disabled (as it is by default), when an S3 client uploads an updated version of an object the system will overwrite the existing replica file with the new version. The token segment of the file name will remain constant (the object keeps the same token) and the timestamp segment of the file name will change.

1.6.5.1. File Digests

Each replica file and each erasure coded fragment file has a corresponding digest containing the hexadecimal MD5 hash of the file as well as a small amount of metadata including the object name (the name of the object for which the file is a replica or an erasure coded fragment) and a last modified timestamp. These digests are used by the HyperStore Service when reading and writing objects and are also used by "[hsstool](#)" (page 863)

operations such as repair and cleanup. The digests are stored in a Checksum DB on the same mount point as the corresponding file. On each mount point, there is one Checksum DB for storing digests for replica data files, and one Checksum DB for storing digests for erasure coded data files. The Checksum DBs are built on [Rock-sDB](#), an open source, high-performance persistent key-value store.

For replica data files, the Checksum DB is stored under:

```
<mountpoint>/digest/hsfs/
```

For erasure coded data files, the Checksum DB is stored under:

```
<mountpoint>/digest/ec/
```

Within each Checksum DB, the *key* is a byte array consisting of the object's token and the file timestamp in binary format, and the *value* is the digest itself.

Note

- When an existing object is updated by an S3 client, the object's token (a decimal formatted MD5 hash of the object key) remains the same but the object's digest (including a hexadecimal formatted MD5 hash of the object data) changes.
- For an erasure coded S3 object, each fragment has the same token (based on object key) but a different digest (based on fragment content).
- For multipart S3 objects — uploaded to the system through the S3 API methods for Multipart Uploads — each part has a different token (since each part has a distinct object key incorporating a part number) and a different digest (based on part content).

Retrieving a Digest

The HyperStore system supports a JMX command for retrieving a digest from a particular node:

- HyperStore Service listener port = 19082
- MBean = *com.gemini.cloudian.hybrid.server.digest:type=RocksDBDigestStore*
- Operation and arguments = *getDigestString=<bucketName>/<objectName>*

For example, using the command line JMX tool *cmdline-jmxclient* that comes bundled with your HyperStore system:

```
[root]# java -jar /opt/cloudian/tools/cmdline-jmxclient-*.jar -- localhost:19082
com.gemini.cloudian.hybrid.server.digest:type=RocksDBDigestStore
getDigestString=bucket1/LocalInstallProcedure.docx

10/25/2016 06:27:30 -0700 org.archive.jmx.Client
getDigestString=bucket1/LocalInstallProcedure.docx:
68855684469431950092982403183202182439.1477401010696032189-0A0A1608
9c741e3e7bbe03e05510071055151a6e
bucket1/LocalInstallProcedure.docx
2016-10-25T13:10:10.696Z
/var/lib/cloudian/hsfs/1mDFFH13tL1DI5YhNKDX3d/a7a28896654319cc7af4c39748a27e3d/243/187/
68855684469431950092982403183202182439.1477401010696032189-0A0A1608
13164
```

For clarity, in the example above an empty line has been inserted between the JMX command and the response. This example is for a replicated object named *LocalInstallProcedure.docx* from the bucket named *bucket1*. In the response, *68855684469431950092982403183202182439.1477401010696032189-0A0A1608* is the Checksum DB key for this entry (in format *<objectToken>.<timestamp>*). The subsequent

lines are the digest contents. `9c741e3e7bbe03e05510071055151a6e` is the replica's MD5 hash in hexadecimal; the `/var/lib/cloudian/hsfs/...` line is the replica file path and name; and `13164` is the replica's file size in bytes.

Note In the command line example above, `-:` is the `USER:PASS` value (indicating that the system is not configured to require a `userid` and password for JMX access). For *cmdline-jmxclient* usage information, enter the following command:

```
[root]# java -jar /opt/cloudian/tools/cmdline-jmxclient-*.jar
```

To check the current *cmdline-jmxclient* version number (replaced by the wildcard character in the command above), change to the `/opt/cloudian/tools` directory and list the directory contents. Look for the *cmdline-jmxclient-<version>.jar* file.

1.6.6. Supporting Services

Services that play a supporting role for the HyperStore system include:

- **Config Controller and Minions** — HyperStore's cluster configuration management system is built on the open source version of [Salt](#). For more information on HyperStore configuration management see ["Using hsctl to Control HyperStore Configuration" \(page 693\)](#). A Config Minion runs on every node. The Config Controller (Salt Master) runs on one node and is also configured on a backup node. [Manual failover to the backup](#) is supported if the primary Config Controller instance goes down.
- **Cloudian Monitoring Data Collector and Agents** — The Cloudian Monitoring Data Collector runs on one node in each of your service regions (the same node on which the system maintenance cron jobs run), and regularly collects data from the Monitoring Agents that run on every node. The Monitoring Collector writes its collected node health statistics to the Metadata DB's "Monitoring" keyspace. The Monitoring Collector and the system maintenance cron jobs are also configured on a backup node, and automatic failover to the backup occurs if the primary node goes offline or if *crond* goes down on the primary.
- **Redis Monitor** — The [Credentials DB and QoS DB](#) are both built on [Redis](#). The Redis Monitor monitors Credentials DB and QoS DB cluster health and implements automatic failover of the master node role within each of the DBs. If the Redis Monitor detects that a DB master node has gone down, it promotes an available slave node to the master node role; and informs the DB's clients (the S3 Service, IAM Service, Admin Service, and HyperStore Service) of the identity of the new master. For redundancy, the Redis Monitor runs on two HyperStore nodes, configured as primary on one node and as backup on the other node.
- **Pre-Configured *chronyd*** — Accurate, synchronized time across the cluster is vital to HyperStore service. When you install your HyperStore cluster, the installation script automatically configures a robust NTP set-up using *chronyd*. For more information see ["NTP Automatic Set-Up" \(page 857\)](#). To see which of your HyperStore nodes are internal NTP servers and which external NTP servers they are synchronizing with, log into the CMC and go to the **Cluster Information** page (**Cluster -> Cluster Config -> Cluster Information**).
- **hsDNS** — hsDNS is a HyperStore component that can act as an authoritative nameserver residing on all HyperStore nodes and providing DNS based load balancing and service discovery for all HyperStore services. By default hsDNS is disabled. For more information see ["hsDNS Feature Overview" \(page 139\)](#) and ["Setting Up hsDNS" \(page 142\)](#).

1.6.7. Auxiliary Services

HyperStore currently supports one type of "auxiliary" service. This is an optional HyperStore component that you can install separately on auxiliary nodes (which have lesser resource requirements than the core HyperStore nodes) after the core HyperStore system has been installed:

- **HyperStore File Service.** This component enables your HyperStore system to support Server Message Block (SMB) and Network File System (NFS) based file services. For more information see:
 - ["File Services Feature Overview" \(page 238\)](#)
 - ["Preparing the File Services Feature" \(page 241\)](#)
- **HyperStore Search Service.** With the HyperStore Search Service your HyperStore users can search for objects based on object metadata, including user-defined metadata. For more information see:
 - ["Object Metadata and Search Feature Overview" \(page 230\)](#)
 - ["Preparing the Metadata Search Feature" \(page 236\)](#)

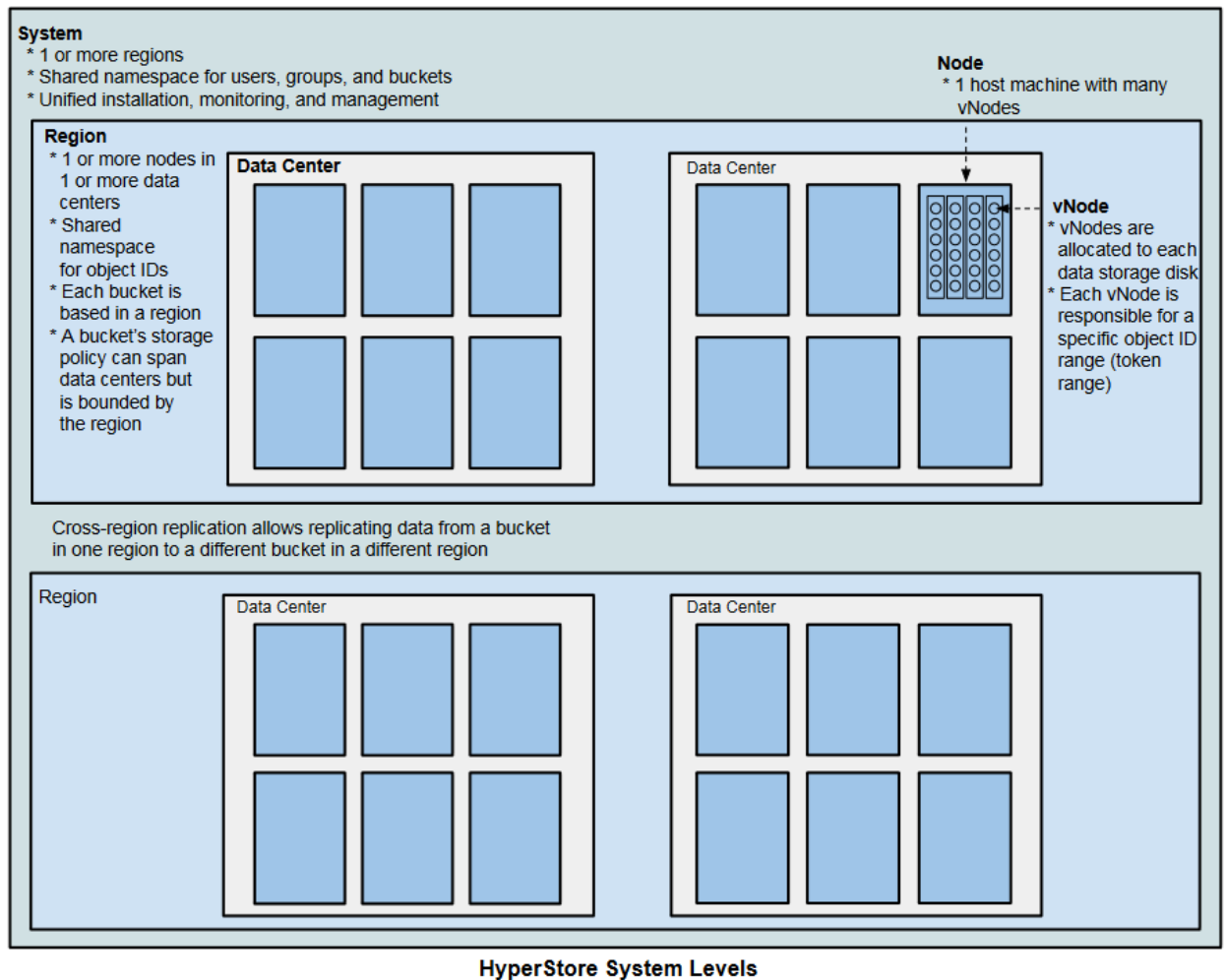
1.7. System Diagrams

- ["System Levels" \(page 70\)](#)
- ["Service Interconnections" \(page 71\)](#)
- ["Services Distribution -- 3 Nodes, Single DC" \(page 72\)](#)
- ["Services Distribution -- Multi-DC, Single Region" \(page 73\)](#)
- ["Services Distribution -- Multi-Region" \(page 74\)](#)
- ["Specialized Services Availability" \(page 75\)](#)
- ["S3 PUT Processing Flow" \(page 76\)](#)
- ["S3 GET Processing Flow" \(page 78\)](#)
- ["Strong Read-After-Write Consistency" \(page 79\)](#)
- ["Dynamic Consistency Levels" \(page 80\)](#)
- ["How vNodes Work" \(page 83\)](#)

1.7.1. System Levels

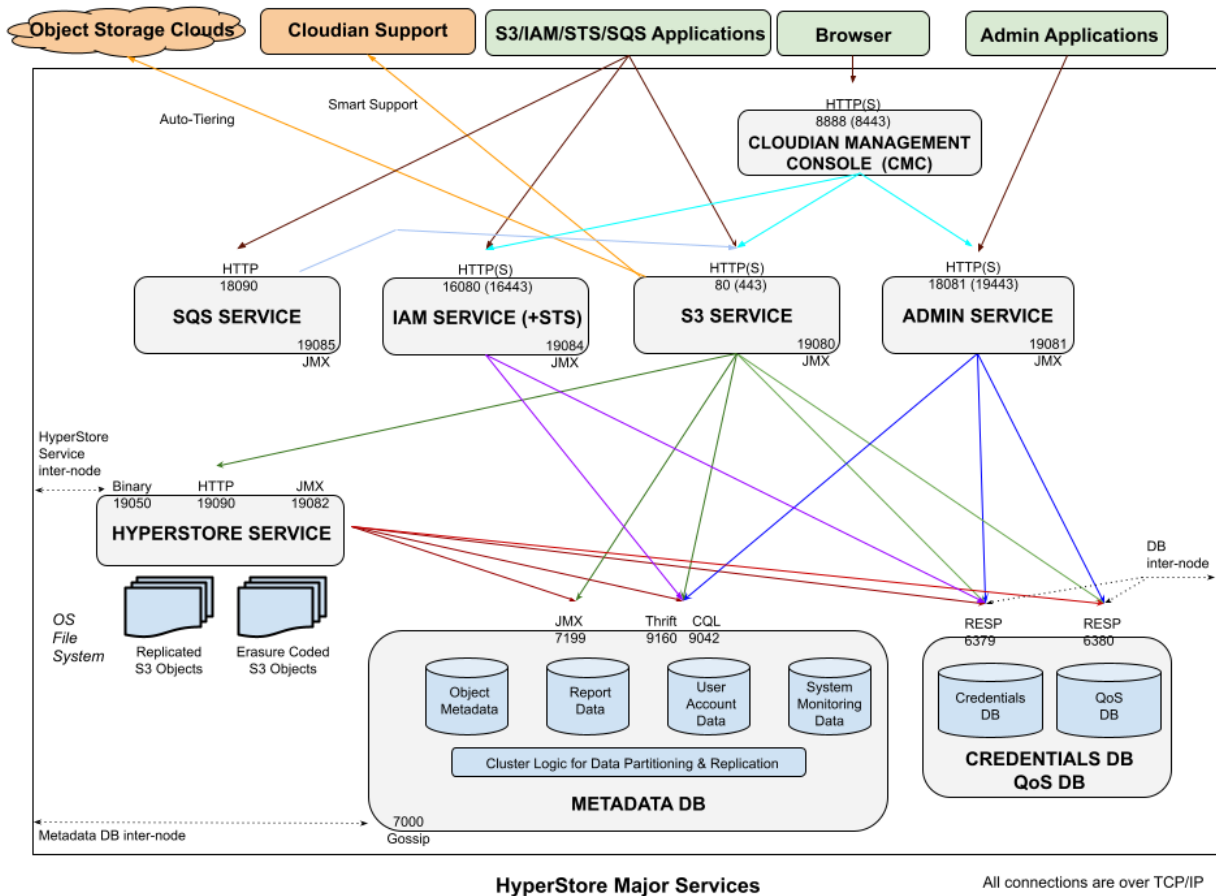
The diagram below shows the conceptual and functional distinctions between the "levels" within a HyperStore system. From broadest to most granular the levels are:

- **System** (also known as a **"Cluster"**)
- **Region**
- **Data Center**
- **Node**
- **vNode**



1.7.2. Service Interconnections

The diagram below shows the major service components that comprise the core HyperStore system, the connections between those services, and the default listening ports to which connections are made.



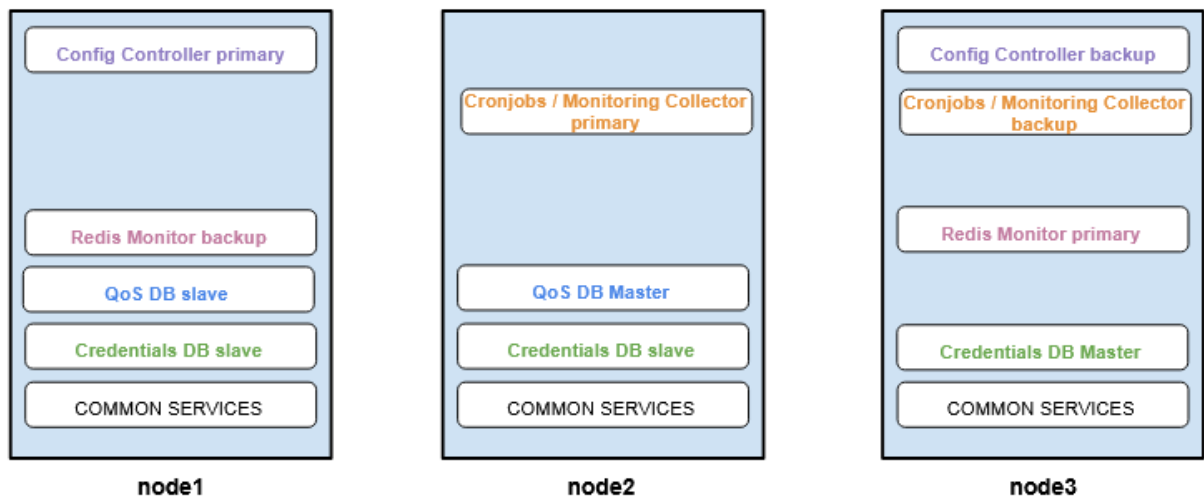
Note The diagram excludes certain supporting services such as the Redis Monitor, the Monitoring Data Collector, and the Config Controller and Minions. For a complete list of HyperStore services and the listening ports they use, see ["HyperStore Listening Ports"](#) (page 57).

1.7.3. Services Distribution -- 3 Nodes, Single DC

Proper distribution of HyperStore service components across multiple physical nodes is handled automatically by the HyperStore installer. The diagram below shows a typical HyperStore services distribution in a three-node cluster within a single data center (DC). Things to note:

- On every node in your cluster are the S3 Service, IAM Service, Admin Service, HyperStore Service, Metadata DB, CMC, Config Minion, and Monitoring Agent. These collectively are labeled as "COMMON SERVICES" in the diagram.
- For each specialized service that has a primary instance and a backup instance (such as Config Controller or Redis Monitor), the backup resides on a different node than the primary. Likewise the QoS DB slave will reside on a different node than the QoS DB Master, and the two Credentials DB slaves will reside on different nodes than the Credentials DB Master.
- If you have a **larger cluster** in a single DC, you will still have the same number of specialized service instances as shown in the diagram (for example, one primary Config Controller instance and one backup instance) — the only difference is that this set of specialized service instances will be spread across your cluster rather than concentrated among three nodes as shown in the diagram. For example

if you have five or more nodes in the DC, then the Credentials DB Master, the two Credentials DB slaves, the QoS DB Master, and the QoS DB slave will all be on different nodes.



HyperStore Services Distribution -- Three Node System

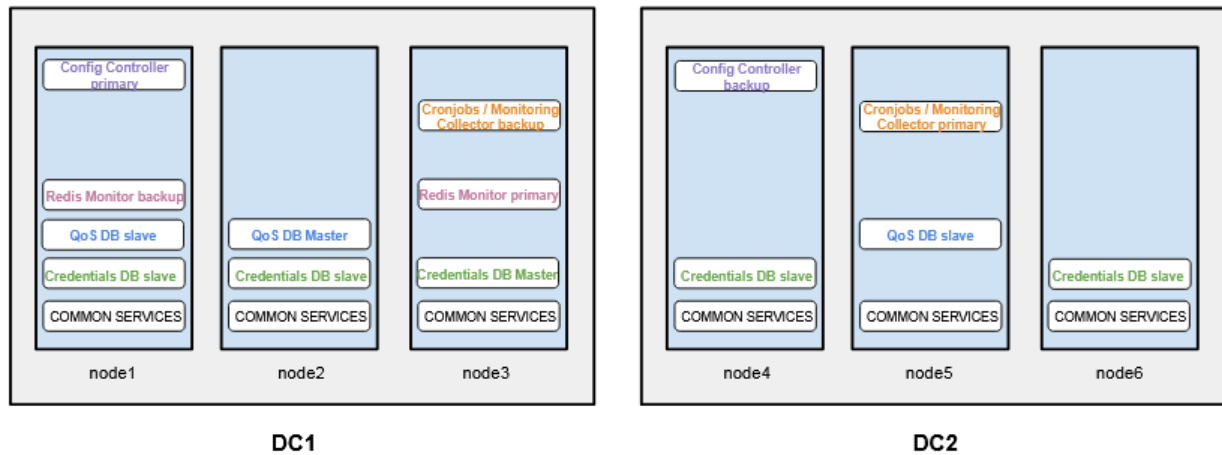
Note For information about the exact location of services in your HyperStore system, log into the CMC and go to the **Cluster Information** page (**Cluster -> Cluster Config -> Cluster Information**). The system allows you to move services from one host to another, if you wish to do so. For instructions see ["Change Node Role Assignments"](#) (page 616).

Note If you have a very large cluster (25 nodes or more in a data center), consult with Cloudbian Support about whether you should add more Credentials DB slaves. For instructions on adding Credentials DB slaves, see ["Move or Add a Credentials DB Slave or QoS DB Slave"](#) (page 617).

1.7.4. Services Distribution -- Multi-DC, Single Region

Proper distribution of HyperStore service components across multiple physical nodes is handled automatically by the HyperStore installer. The diagram below shows a typical HyperStore services distribution across a six-node system that spans two data centers. The system is configured as a single service region. Things to note:

- The "COMMON SERVICES" (S3 Service, IAM Service, Admin Service, HyperStore Service, Metadata DB, CMC, Config Minion, and Monitoring Agent) run on every node in your multi-DC system.
- Each data center has its own QoS DB slave and its own two Credentials DB slaves, for read performance optimization.
- The Config Controller backup is placed in a different DC than the Config Controller primary; and the same is true for the Cronjobs backup and primary.



HyperStore Services Distribution -- Two Data Centers Configured as a Single Service Region

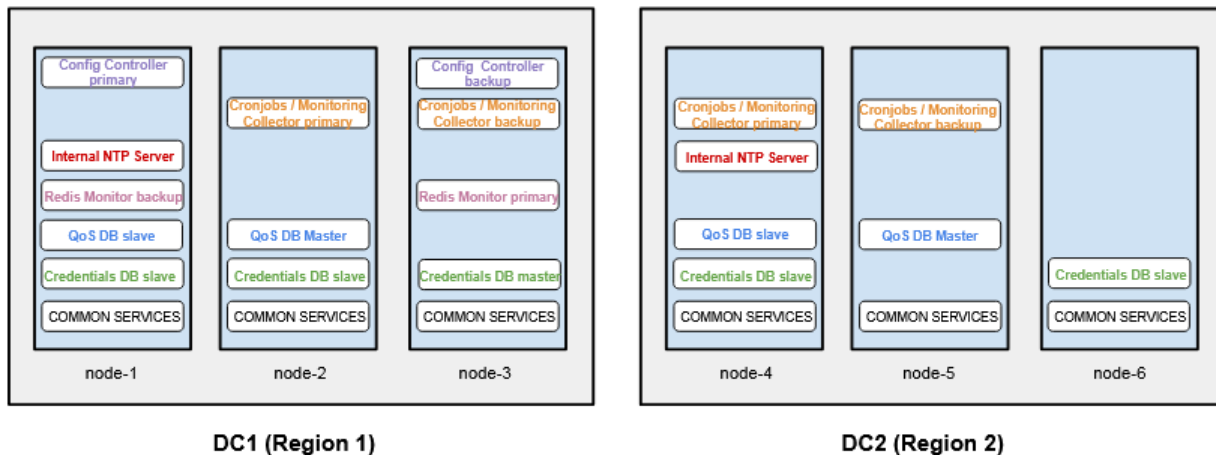
Note The Redis Monitor backup must remain in the same data center as the Redis Monitor primary, and this should be the same data center as where the Credentials DB master is located.

Note To check the current location of specialized services within your multi-DC HyperStore system, go to the CMC's **Cluster Information** page (**Cluster -> Cluster Config -> Cluster Information**).

1.7.5. Services Distribution -- Multi-Region

Proper distribution of HyperStore service components across multiple physical nodes is handled automatically by the HyperStore installer. This diagram shows a six-node system that spans two data centers, and this time the system is configured as two different service regions. Things to note:

- The "COMMON SERVICES" (S3 Service, Admin Service, HyperStore Service, Metadata DB, CMC, Config Minion, and Monitoring Agent) run on every node in your multi-region system. The IAM Service runs on every node in the default region only.
- The whole multi-region system is served by a single active Config Controller, a single Credentials DB Master, and a single active Redis Monitor.
- Each region has its own QoS DB Master and its own active Cronjob host.



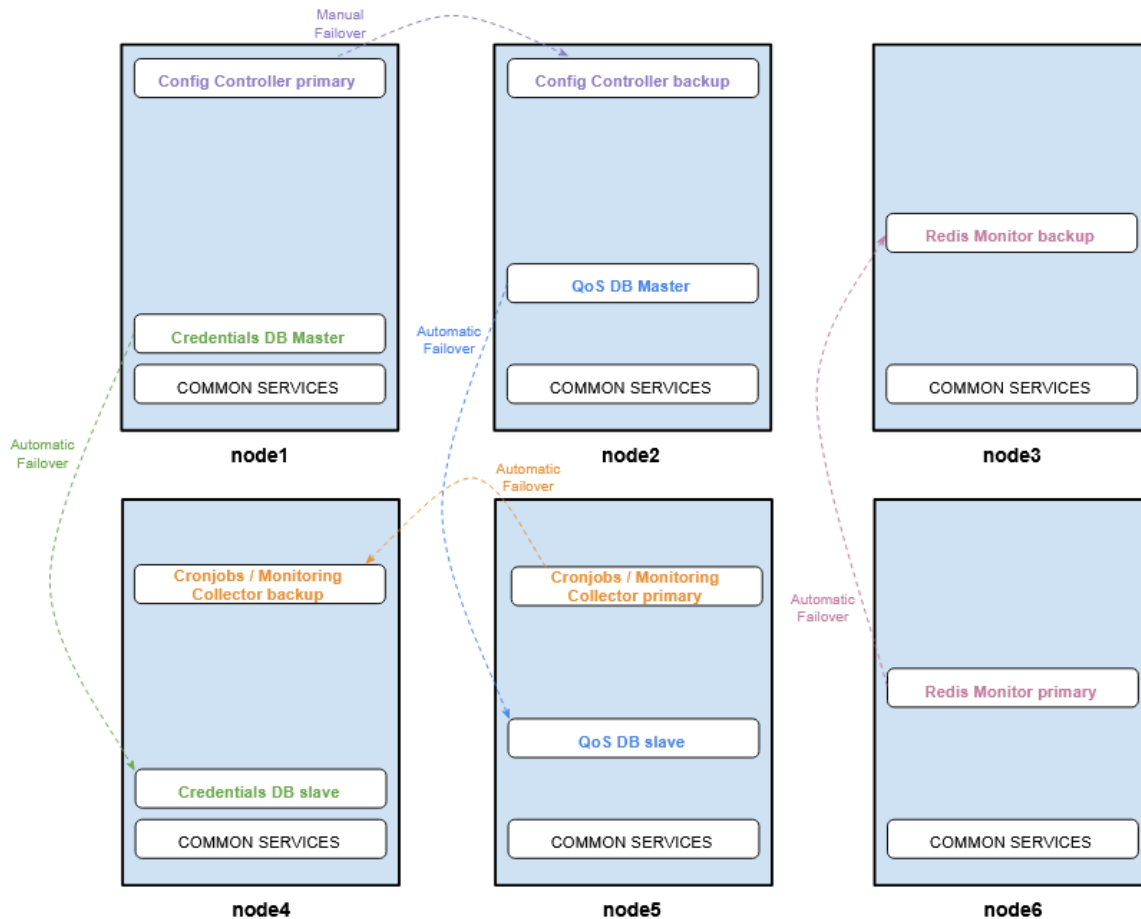
HyperStore Services Distribution -- Two Data Centers Configured as Different Service Regions

1.7.6. Specialized Services Availability

Along with the services that are common to every HyperStore node (such as the S3 Service, HyperStore Service, Metadata DB, and so on) your HyperStore system includes several specialized services that run only on certain nodes. The HyperStore installer [automatically distributes these services across your cluster](#). Each specialized service has a primary instance and a backup instance, and the installer ensures that for each specialized service the primary instance and backup instance are deployed on different nodes.

The diagram below illustrates how the system ensures high availability of these specialized services by supporting failover of each service type, from the primary instance to the backup instance. For nearly all service types, the system automatically detects a failure of the primary instance and automatically fails over to the backup instance. The one exception is the Config Controller role — by default system configuration, for the Config Controller you must [manually implement failover](#) if there's a problem with the primary instance (but see the first bullet point below the diagram).

The diagram shows six nodes, but the principles are the same regardless of how many nodes you have: specialized services are dispersed across the cluster, and the backup instance of any given service is deployed on a different node than the primary instance.



HyperStore Specialized Services -- High Availability

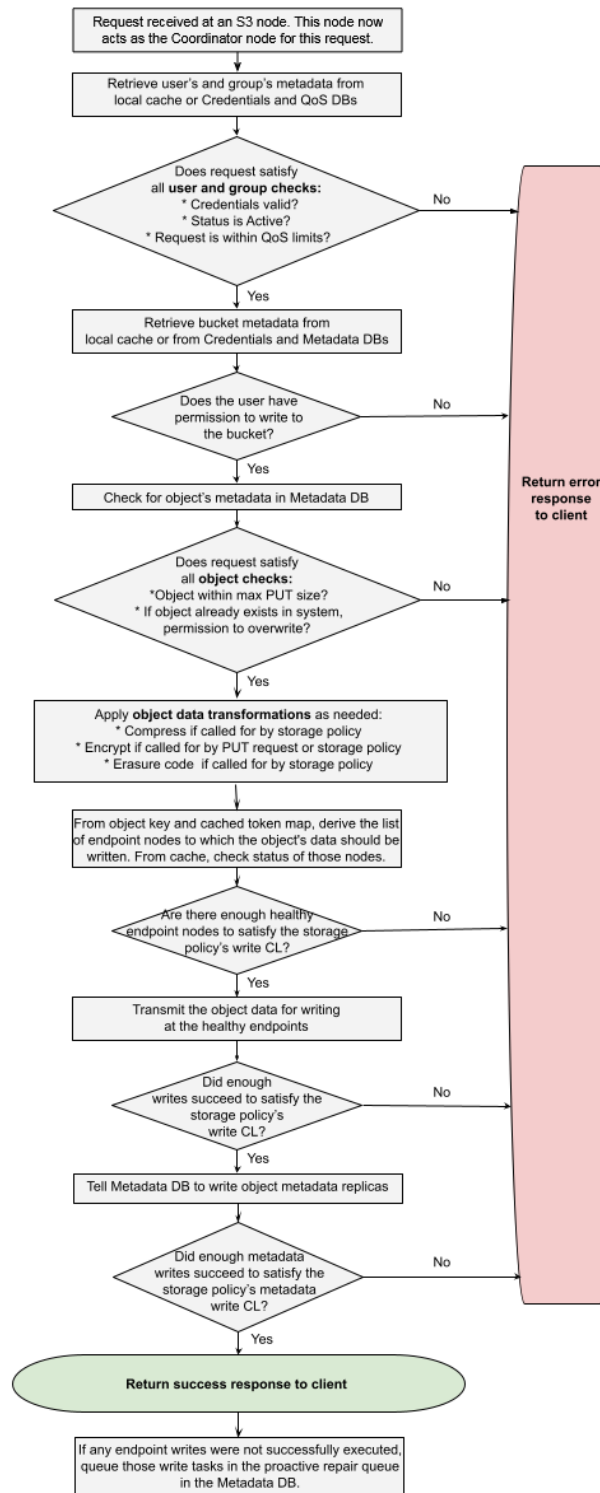
Note that:

- HyperStore supports a configurable option to have the Config Controller primary role fail over to the backup host automatically rather than requiring manual intervention. See [com-mon.configController.failover](#). By default automatic failover is disabled and manual intervention is required.
- The automatic failover of the Cronjobs and Monitoring Data Collector roles from the primary to the backup instance invokes the *cloudianInstall.sh* script and will fail if *cloudianInstall.sh* is already running. When you occasionally use *cloudianInstall.sh* for system configuration tasks, remember to exit the installer when you are done — do not leave it running.
- The automatic failover of the Cronjobs and Monitoring Data Collector roles from the primary to the backup instance will not occur until the primary instance has been down for 10 minutes.

1.7.7. S3 PUT Processing Flow

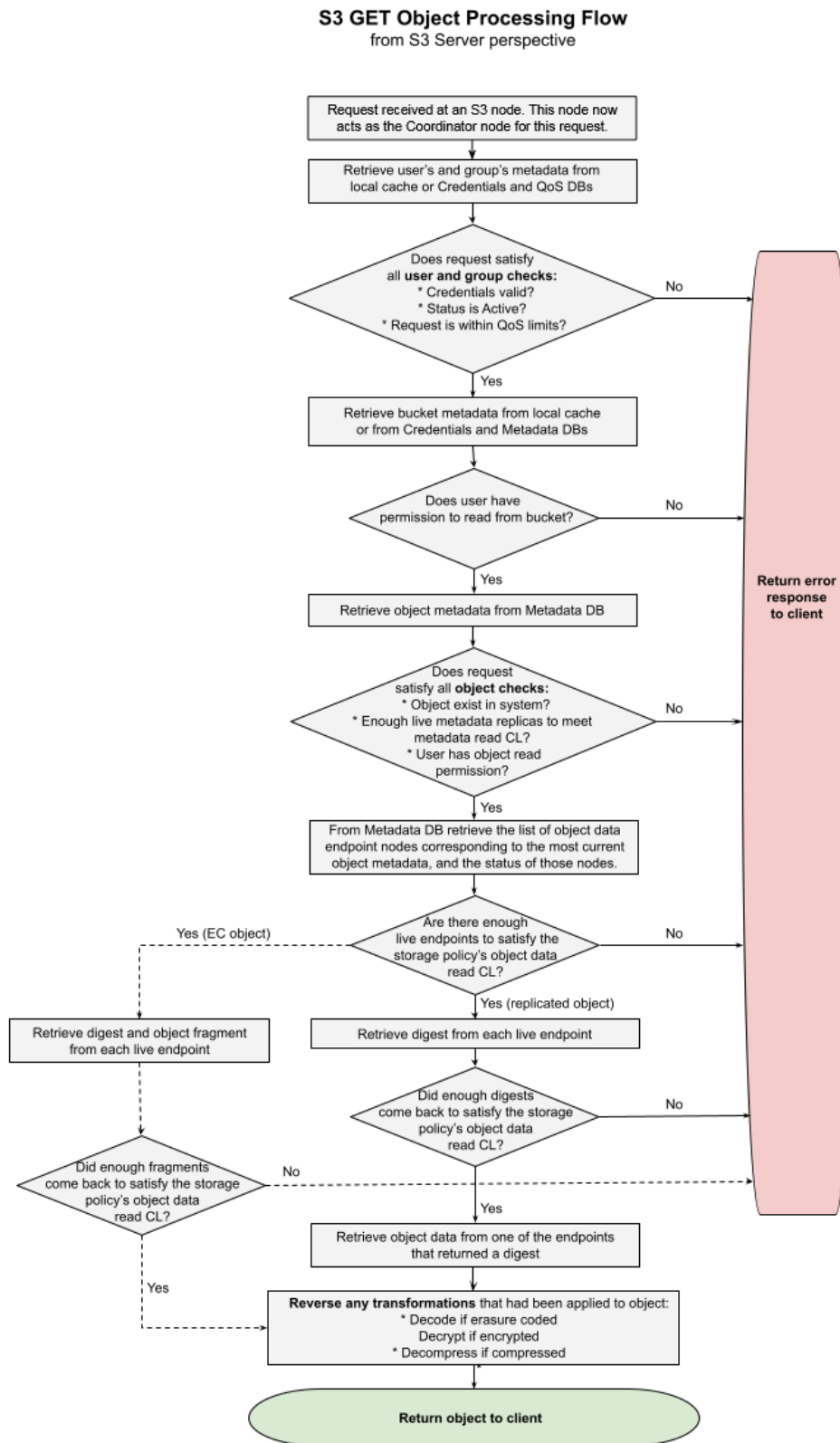
The diagram below shows the main aspects of how the HyperStore system processes an S3 PUT Object request. The flow is presented from the perspective of the S3 Service, which handles incoming S3 requests. The S3 Service runs on all of the nodes in your cluster.

S3 PUT Object Processing Flow from S3 Server perspective



1.7.8. S3 GET Processing Flow

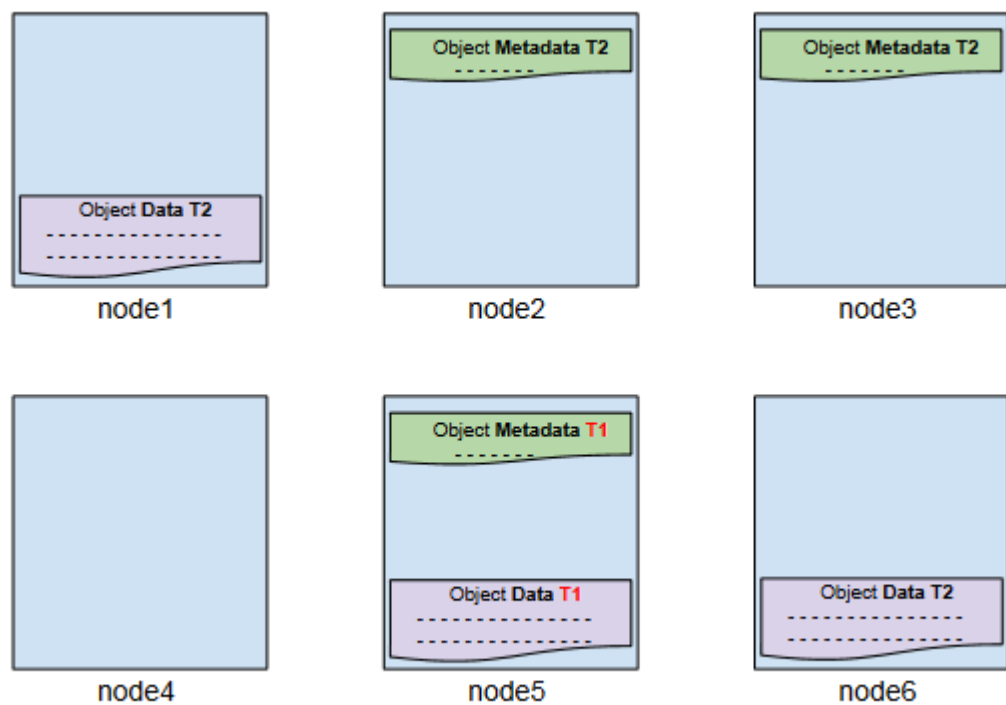
This diagram shows the main aspects of how the HyperStore system processes an S3 GET Object request, from the perspective of the S3 Service.



1.7.9. Strong Read-After-Write Consistency

Typically all the replicas of a given object, and all the replicas of that object's metadata, will be consistent — that is, all the replicas will be equally current. However, because HyperStore allows you to configure storage policies that utilize eventual consistency for writes, there may be times when an object's data replicas and/or metadata replicas are temporarily inconsistent. If a read request on the object comes into the system during such a time, by default HyperStore either returns the freshest data or — if no fresh replica is available — fails the request. In this way HyperStore by default ensures **strong read-after-write consistency**.

Consider a 3X replication scenario where QUORUM has been used as the write consistency level (which is the default configuration for replication storage policies). Suppose an S3 PUT of an updated version of an object has succeeded even though only two of three object data replica writes and only two of three object metadata replica writes succeeded. We then can temporarily have a condition like that shown in the following diagram, where "T2" indicates the timestamp of the new version of the data and metadata and "T1" indicates the out-dated version. (For example, perhaps *node5* was momentarily offline when the S3 write request came in; and now it's back online but [proactive repair](#) has not yet completed.)



If an S3 read request on the object comes into the system during this temporary period of stored data inconsistency, the system works as follows:

- As long as the read consistency level is set to at least QUORUM (the default for replication storage policies), the system will read at least two of the metadata replicas. Consequently it will read at least one of the fresh metadata replicas, with timestamp T2. If it reads one T1 metadata replica and one T2 metadata replica, it works with the metadata that has the freshest timestamp. The system then tries to retrieve an object data replica that has this same fresh timestamp.
- If object data replicas with the fresh timestamp are available, that object data is returned to the S3 client. If nodes are down in such a way that the only available object data replica is the outdated one, then the system fails the S3 request.

In this way, by default configuration HyperStore delivers strong consistency (specifically, strong read-after-write consistency).

Note Although not pictured here, HyperStore delivers strong read-after-write consistency for erasure coding storage policies by similar logic (in the case of erasure coded data, each of the k fragments required for an object data read must match against the newest timestamp found in the successful QUORUM read of object metadata).

Note HyperStore allows you to configure replication storage policies that use a read CL of ONE rather than QUORUM. This non-default configuration maximizes read availability and speed, but also increases the chances of returning a stale replica to the client. This is because -- if your write CL is QUORUM (the default) and your read CL is ONE (non-default) -- there is a chance of reading only a stale metadata replica and returning a stale object replica to the client.

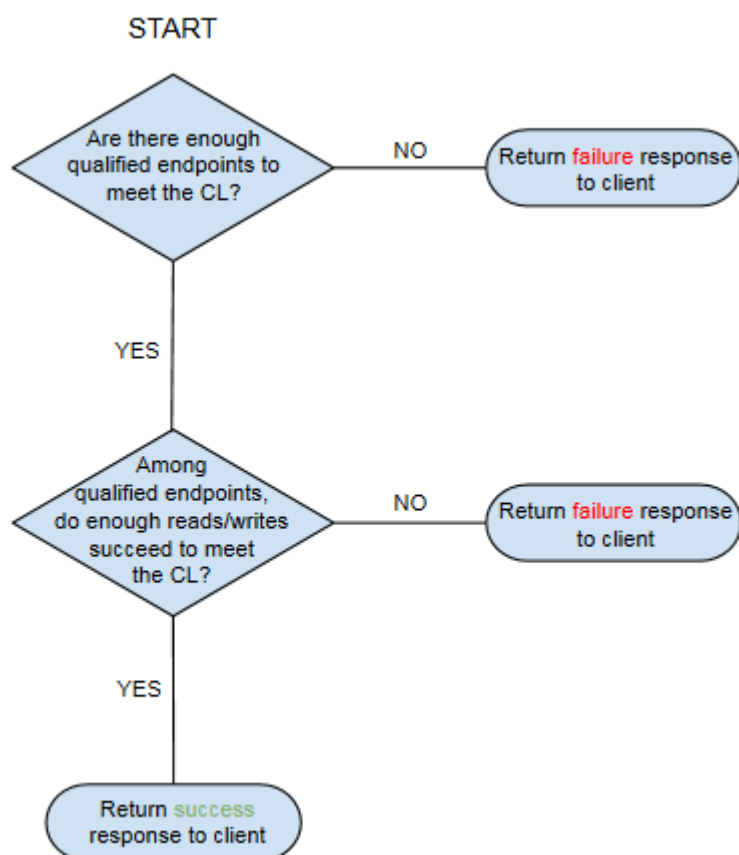
For more information on S3 write and read availability under various consistency level configurations, see ["Storage Policy Resilience to Downed Nodes"](#) (page 156).

1.7.10. Dynamic Consistency Levels

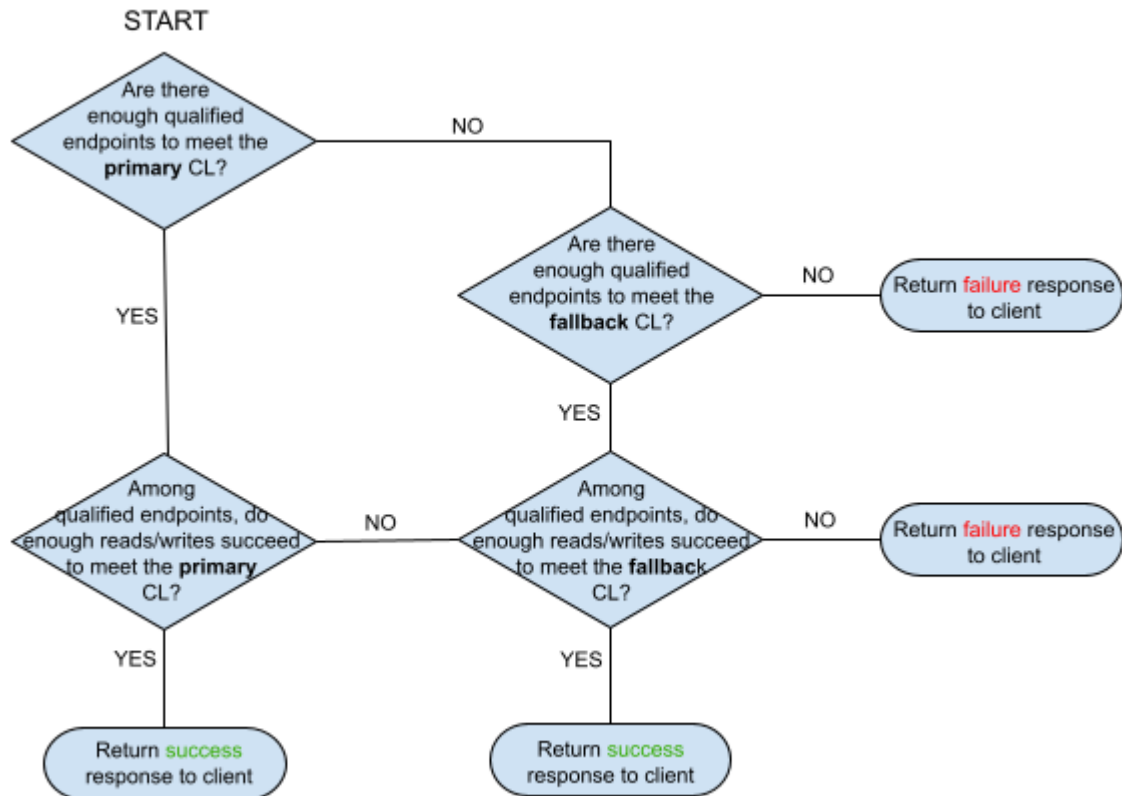
When you ["Add a Storage Policy"](#) (page 513) to your HyperStore system one of the policy dimensions that you configure is consistency requirements for writes and reads of object data and metadata. When doing so, you have the option of configuring "dynamic" consistency level requirements. With dynamic CLs, you specify two or more consistency levels that differ in strictness. If the stricter consistency level (the "primary" consistency level) cannot be met for a given S3 request, then the system tries to meet the less strict level (the "fallback" consistency level).

The first flow chart below illustrates the standard consistency level logic when only one consistency level (CL) is used per operation type. The second flow chart illustrates the logic for a two-tier dynamic consistency level configuration. Following the flow charts is a detailed text description of how the dynamic consistency level feature works, which includes discussion of what constitutes a "qualified endpoint".

1.7.10.1. Standard Consistency Level Logic



1.7.10.2. Dynamic Consistency Levels Logic



1.7.10.3. Dynamic Consistency Levels Logic Described

When the S3 Service processes an S3 PUT or GET of an object, it checks the system for a list of "endpoints" for that particular object — the nodes that the object data should be written to or read from. The system then checks whether any of the endpoint nodes is in any of these disqualifying conditions:

- HyperStore Service has been marked as down by the S3 Service
- Node has been put into Maintenance Mode by operator (see ["Start Maintenance Mode"](#) (page 489))
- Node is in a StopWrite condition due to all data disks being 90% full or more (relevant only for write requests)
- Disk on which requested data resides is [disabled](#) (relevant only for read requests)

Then, dynamic consistency levels can come into play at two different phases of S3 PUT or GET processing:

- If the number of qualifying endpoints meets the requirements of either the primary CL **or the fallback CL**, the system proceeds with trying to write to (or read from in the case of GET processing) those endpoints. If the number of qualifying endpoints does not meet the requirements of either the primary CL or the fallback CL, the system does not try to write to (or read from) the endpoints and the S3 request fails and an error is returned to the S3 client.
- When trying to write to (or read from) qualifying endpoint nodes, if the number of successful writes/reads meets the requirements of either the primary CL **or the fallback CL**, a success response is returned to the S3 client. If the number of successful writes/reads does not meet the

requirements of either the primary CL or the fallback CL, the S3 request fails and an error is returned to the S3 client.

1.7.11. How vNodes Work

Following is an in-depth look at HyperStore vNodes, including diagrams to illustrate the role that vNodes play in supporting high-availability object storage in a HyperStore cluster.

S3 object placement and replication within a HyperStore cluster is based on a consistent hashing scheme that utilizes an integer token space ranging from 0 to $2^{127}-1$. Traditionally, in a storage cluster based on consistent hashing, each physical node is assigned an integer token from the token space. A given node is then responsible for a **token range** that extends from the next-lower token assigned to a different node (excluding the token number itself), up to and including the given node's own token. Then, an integer hash value is calculated for each S3 object as it is being uploaded to storage. The object is stored to the node responsible for the token range in which the object's hash value falls. Replication is implemented by also storing the object to the nodes responsible for the next-higher token ranges.

Advancing beyond traditional consistent hash based storage, the HyperStore system utilizes and extends the "virtual node" (vNode) functionality originally introduced in Cassandra version 1.2. This optimized design assigns multiple tokens to each physical node. In essence, the storage cluster is composed of very many "virtual nodes", with multiple virtual nodes residing on each physical node. Each virtual node is assigned its own token and has its own token range for which it is responsible.

The HyperStore system goes a significant step further by assigning a different set of tokens (virtual nodes) to **each HyperStore data disk** on each host. With this implementation, each data disk on a host is responsible for a set of different token ranges and -- consequently -- a different inventory of object data. If a disk fails it affects only the object data on that one disk. The other disks on the host can continue operating and supporting their own data storage responsibilities.

The number of tokens that the system assigns to each host is based on the total combined storage capacity of the host's HyperStore data disks. Specifically, the **system determines the number of tokens to assign to a host by taking the total number of terabytes of HyperStore data storage capacity on the host, multiplying by .7, and then rounding off** to the nearest integer. Further, the system applies a lower bound of one token per HyperStore data disk and an upper bound of 512 tokens per host.

For example:

Number of Data Disks on Host	Size of Data Disks	Total TBs of Data Disk on Host	Number of Tokens Assigned To Host
2	500GB	1TB	2 (1TB X .7 = .7, but minimum is 1 token per data disk)
4	8TB	32TB	22 (32TB X .7 = 22.4, rounded = 22)
8	4 X 8TB 4 X 10TB	72TB	50 (72TB X .7 = 50.4, rounded = 50)
12	10TB	120TB	84 (120TB X .7 = 84)
24	16TB	384TB	268

Number of Data Disks on Host	Size of Data Disks	Total TBs of Data Disk on Host	Number of Tokens Assigned To Host
			(384TB X 0.7 = 268.8, rounded = 269)

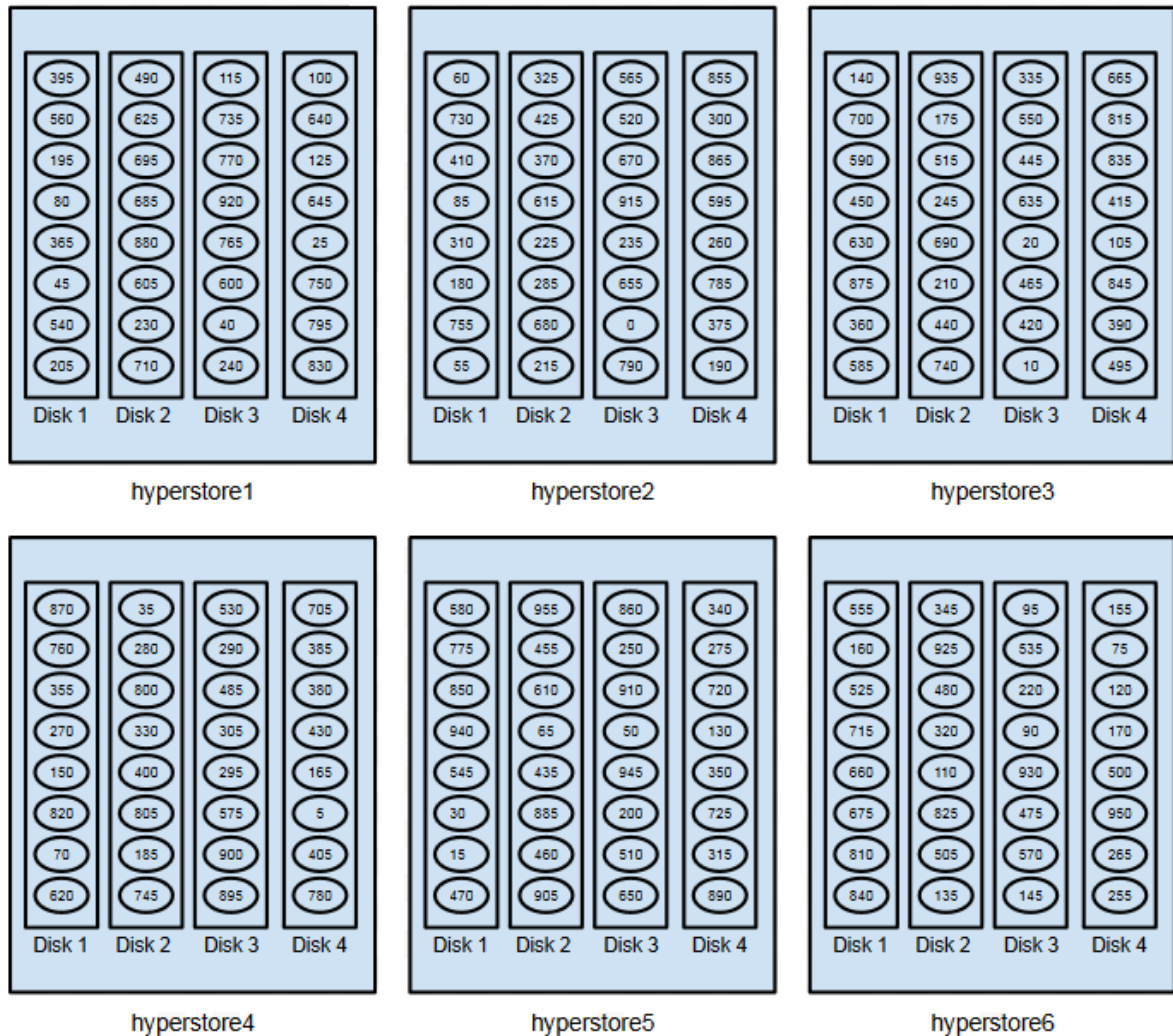
Note The data disk sizes in the table above are only for simple illustration of how the algorithm works. In calculations for actual hosts, the system uses not the disk raw size (the decimal-based size stated by the disk manufacturer) but rather the disk usable size after the disks are formatted and the file system is mounted (the binary-based size the operating system reports if you run the *lsblk* command on the host).

On each host, the host's assigned tokens -- and associated token ranges -- are automatically allocated to each HyperStore data disk in a manner such that storage capacity utilization should be [approximately balanced among the disks](#) on a given host.

In the [HyperStore File System](#) mounted to each HyperStore data disk there are sub-directories that demarcate each vNode's data.

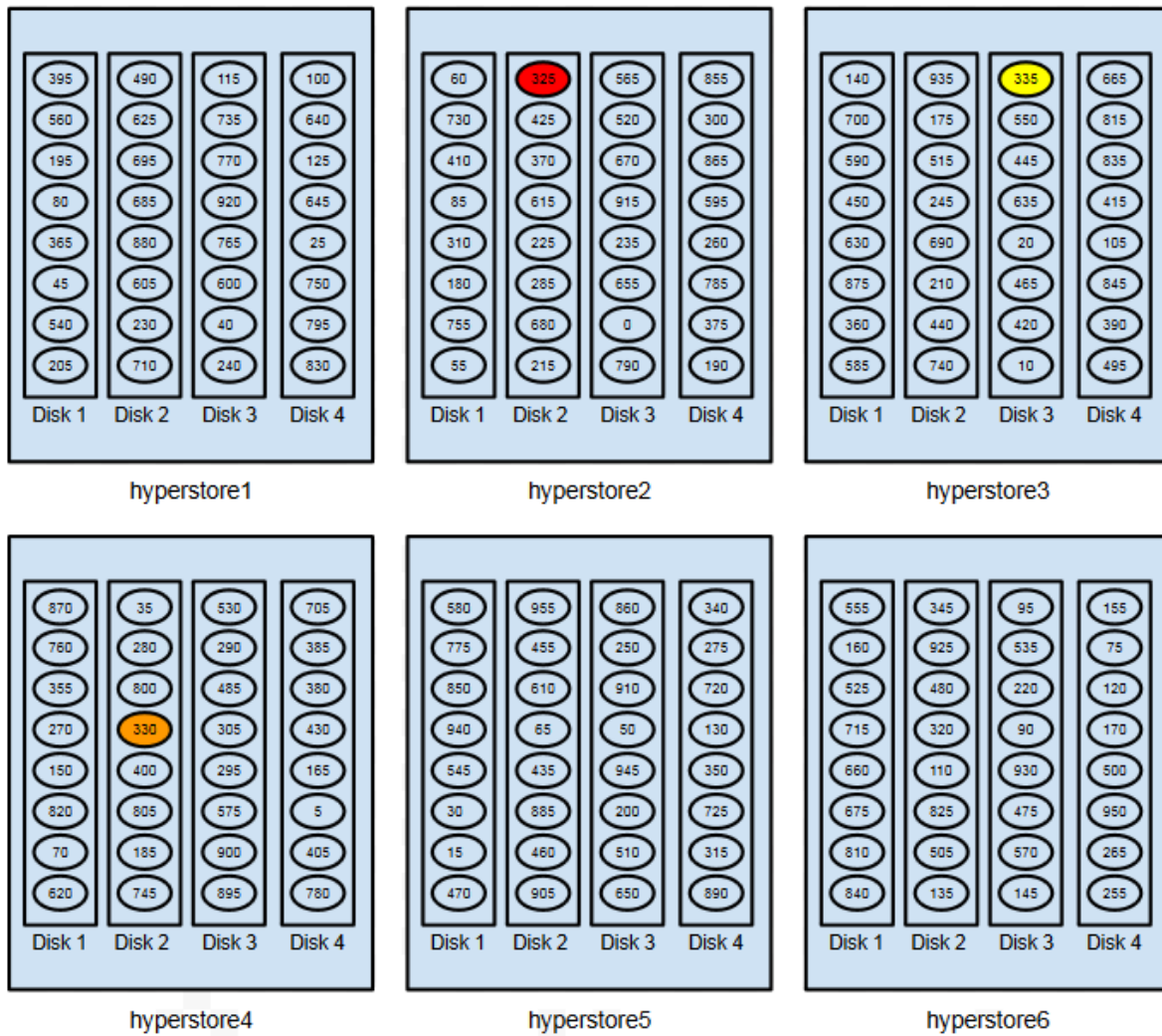
For illustration of how vNodes work to guide the distribution of data across a cluster, consider a cluster of six HyperStore hosts each of which has four disks designated for S3 object storage. Suppose that each physical host is assigned 32 tokens. And suppose for illustration that there is a simplified token space ranging from 0 to 960, and the values of the 192 tokens in this system (six hosts times 32 tokens each) are 0, 5, 10, 15, 20, and so on up through 955.

The diagram below shows one possible allocation of tokens across the cluster. Each host's 32 tokens are divided evenly across the four disks (eight tokens per disk), and that token assignment is randomized across the cluster.



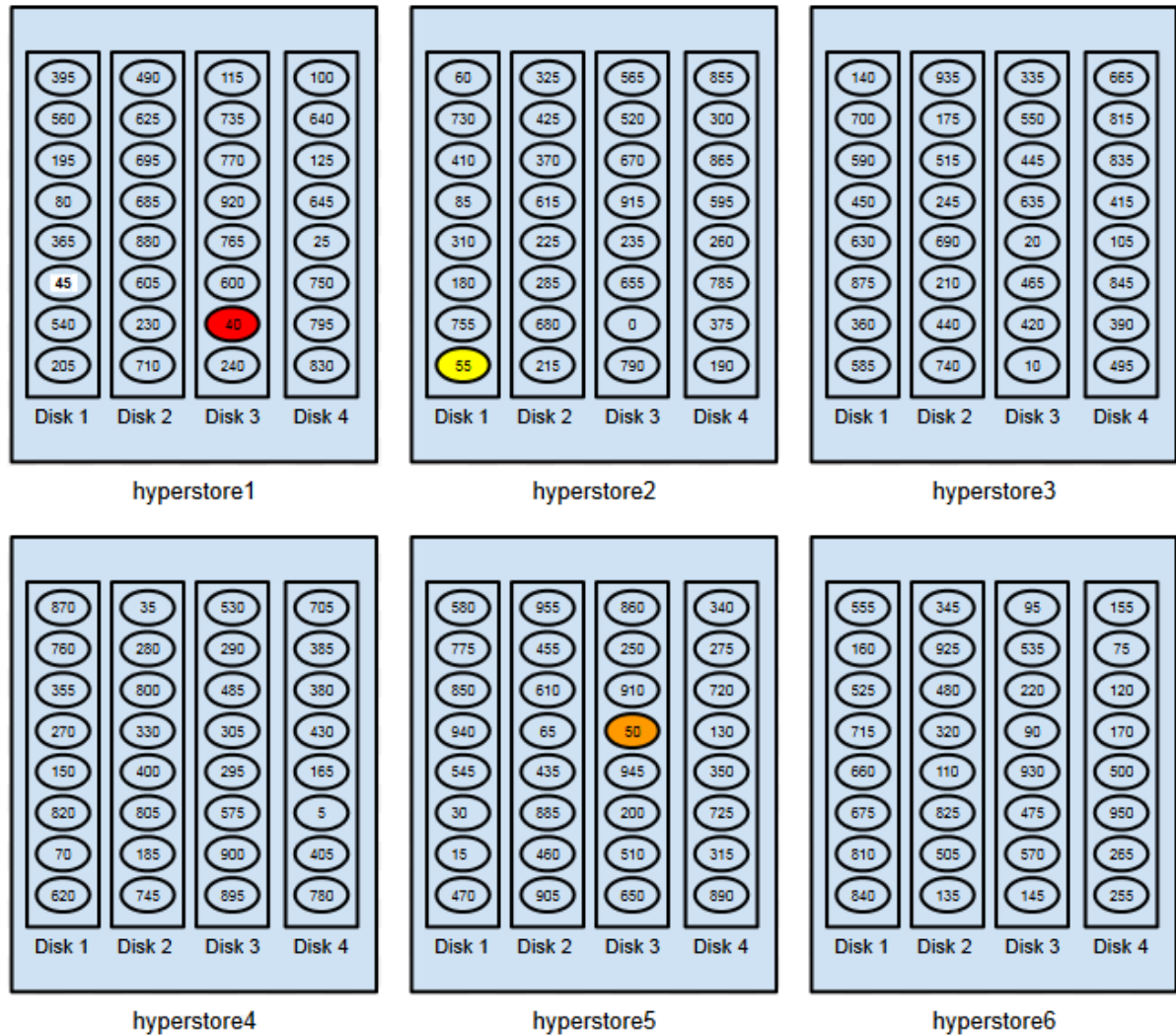
Now further suppose that you've configured your HyperStore system for 3X replication of S3 objects. And say that an S3 object is uploaded to the system and the hashing algorithm applied to the unique *<bucketname>/<objectname>* combination gives us a hash value of 322 (for this simplified example; in reality the system uses MD5 hashing). The diagram below shows how three instances or "replicas" of the object will be stored across the cluster:

- With its object name hash value of 322, the "primary replica" of the object is stored on the vNode responsible for the token range that includes the value 322. This is the vNode assigned token 325 (highlighted in red in the diagram below) -- this vNode has responsibility for a token range spanning from 320 (exclusive) up to 325 (inclusive). A simple way of identifying where the primary replica will go is that it's the vNode with the **lowest token that's higher than the object's hash value**. Note that the "primary replica" has no functional primacy compared to other replicas; it's called that only because its placement is based simply on identifying the disk that's responsible for the token range into which the object hash falls.
- The secondary replica is stored to the vNode that's assigned the next-higher token (330, highlighted in orange), which is located at hyperstore4:Disk2.
- The tertiary replica is stored to the vNode that's assigned the next-higher token after that (335, in yellow), which is at hyperstore3:Disk3.



Working with the same cluster and simplified token space, we can next consider a second object replication example that illustrates an important HyperStore vNode principle: no more than one of an object's replicas will be stored on the same physical host. Suppose that an S3 object is uploaded to the system and the object name hash is 38. The next diagram shows how the object's three replicas are placed:

- The primary replica is stored to the vNode that's assigned token 40 — at hyperstore1:Disk3 (red highlight in the diagram below).
- The vNode with the next-higher token — 45 (with white label) — is on a different disk (Disk1) on the same physical host as token 40, where the HyperStore system is placing the primary replica. Because it's on the same physical host, the system skips over the vNode with token 45 and places the object's secondary replica where the vNode with token 50 is — at hyperstore5:Disk3 (orange highlight).
- The tertiary replica is stored to the vNode with token 55, at hyperstore2:Disk1 (yellow highlight).



The Disk Perspective

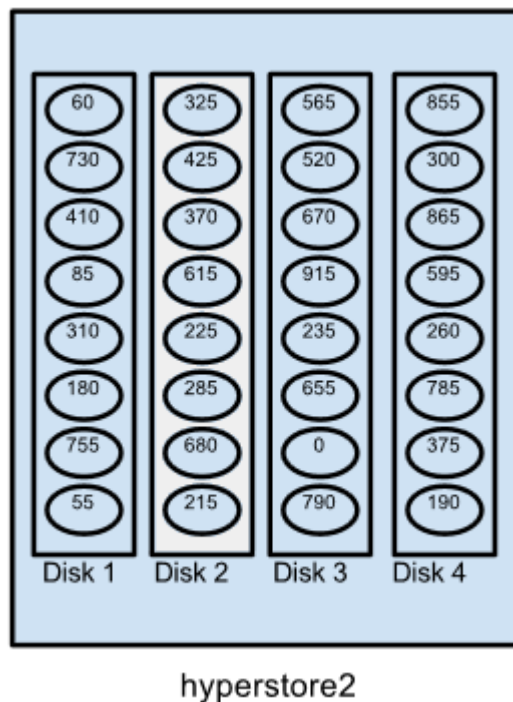
Now let's change perspective and see how things look for a particular disk from within the cluster. Recall that we've supposed a simplified token space with a total of 192 tokens (0, 5, 10, 15, and so on up to 955), randomly distributed across the cluster so that each host has 32 tokens and each host's tokens are evenly divided among its disks. We can zero in on hyperstore2:Disk2 – highlighted in the diagram below — and see that this disk has been assigned tokens 325, 425, 370, and so on.

Assuming the cluster is configured for 3X replication, on hyperstore2:Disk2 will be stored the following:

- In association with the disk's assigned token 325:
 - Primary replicas of objects for which the hash values are from 320 (exclusive) to 325 (inclusive)
 - Secondary replicas of objects for which the hash values are from 315 (exclusive) to 320 (inclusive)
 - Tertiary replicas of objects for which the hash values are from 310 (exclusive) to 315 (inclusive)

- In association with the disk's assigned token 425:
 - Primary replicas of objects for which the hash values are from 420 (exclusive) to 425 (inclusive)
 - Secondary replicas of objects for which the hash values are from 415 (exclusive) to 420 (inclusive)
 - Tertiary replicas of objects for which the hash values are from 410 (exclusive) to 415 (inclusive)
- And so on.

As noted previously, the HyperStore system when placing secondary and tertiary replicas may in some cases skip over tokens so as not to store more than one replica of an object on the same physical host. So this dynamic could result in additional responsibilities for hyperstore2:disk2 as a possible endpoint for secondary or tertiary replicas.



In the event that Disk 2 fails, Disks 1, 3, and 4 will continue fulfilling their storage responsibilities. Meanwhile, objects that are on Disk 2 persist within the cluster because they've been replicated on other hosts. (Whether those objects will still be readable by S3 clients will depend on how you have configured [consistency level](#) requirements.)

Multi-Data Center Deployments

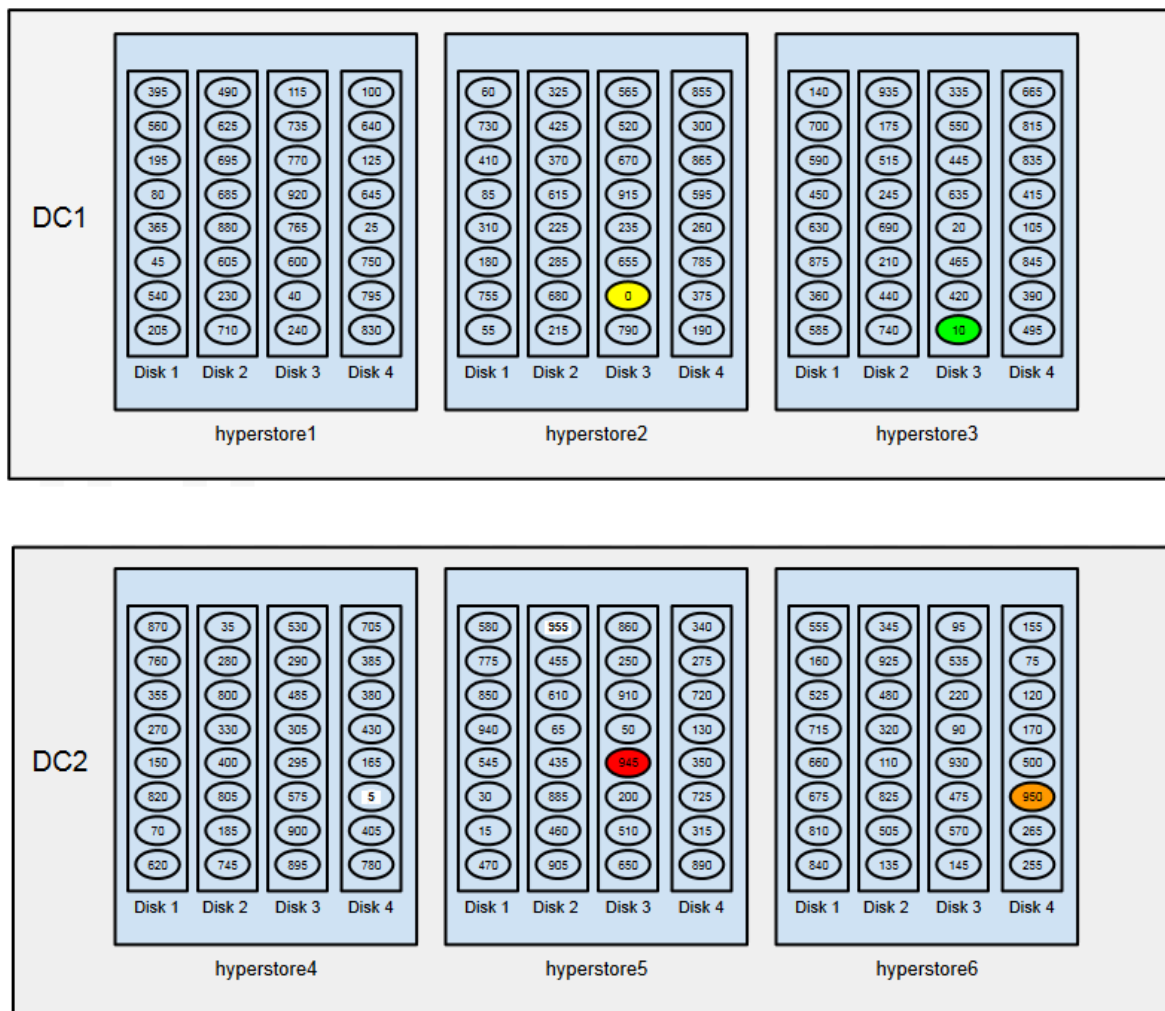
If you deploy your HyperStore cluster across multiple data centers within the same service region, your multiple data centers will be integrated in one unified token space.

Consider an example of a HyperStore deployment that spans two data centers — DC1 and DC2 — each of which has three physical nodes. As in our previous examples, each physical node has four disks; each host is assigned 32 tokens (vNodes); and we're supposing a simplified token space that ranges from 0 to 960. In this multi-DC scenario, the token space is divided into 192 tokens — 32 for each of the six physical hosts — which are randomly distributed across the six hosts.

Suppose also that S3 object replication in this deployment is configured at two replicas in each data center.

We can then see how a hypothetical S3 object with a hash value of 942 would be replicated across the two data centers:

- The first replica is stored to the vNode that's assigned token 945 (in red in the diagram below) — which is located in DC2, on hyperstore5:Disk3.
- The second replica is stored to vNode 950 (orange) — DC2, hyperstore6:Disk4.
- The next-higher vNode (955, with high-contrast label) is in DC2, where we've already met the configured replication level of two replicas — so we skip that vNode.
- The third replica is stored to vNode 0 (yellow) — DC1, hyperstore2:Disk3. Note that after the highest-numbered token (955) the token "ring" circles around to the lowest token (0). (In a more realistic token space there would be a token range spanning from the highest vNode token [exclusive] through the top of the token space and around to the lowest vNode token (inclusive)).
- The next-higher vNode (5, high-contrast label) is in DC2, where we've already met the configured replication level — so we skip that vNode.
- The fourth and final replica is stored to vNode 10 (green) — DC1, hyperstore3:Disk3.



Multi-Region Deployments

If you deploy a HyperStore system across multiple service regions, each region has its own independent

storage cluster -- with each cluster having its own 0 to $2^{127}-1$ token space, its own set of vNodes, and its own independent inventory of stored S3 objects. There is no per-object replication across regions.

Erasure Coded Data

When the HyperStore erasure coding feature is used, vNodes are the basis for distribution of encoded object fragments. Each of an object's $k+m$ erasure coded fragments is assigned a hash value (token) by the system, and then each fragment is stored to the vNode responsible for the token range that contains the fragment's token.

Cassandra Data

In a HyperStore system, Cassandra is used for storing object metadata and system metadata. In a typical deployment, on each HyperStore node Cassandra data is stored on the same RAID-mirrored disks as the OS. Cassandra data is not stored on the HyperStore data disks (the disks whose mount points are specified by the configuration setting "[hyperstore.directory.data](#)" (page 766)).

When vNodes are assigned to a host machine, they are allocated across only the host's HyperStore data mount points. vNodes are not allocated to the mirrored disks on which Cassandra data is stored.

Within a cluster, metadata in Cassandra is replicated in accordance with your storage policies. Cassandra data replication leverages vNodes in this manner:

- When a new Cassandra object -- a row key and its associated column values -- is created, the row key is hashed and the hash (token) is used to associate the object with a particular vNode (the vNode responsible for the token range that contains the Cassandra object's token). The system checks to see which host machine that vNode is located on, and the "primary" replica of the Cassandra object is then stored on the Cassandra disk(s) on that host.
- For example, suppose a host machine is assigned 96 vNodes, allocated across its multiple HyperStore data disks. Cassandra objects whose hash values fall into the token ranges of **any of those 96 vNodes** will get written to the Cassandra disk(s) on that host.
- Additional replicas of the Cassandra object (the number of replicas depends on your configuration settings) are then associated with next-higher-up vNodes and stored to whichever hosts those vNodes are located on — with the condition that if necessary vNodes will be "skipped" in order to ensure that each replica of the Cassandra object is stored on a different host machine.

vNode Benefits

vNodes provide several advantages over conventional one-token-per-host schemes, including:

- Token assignment is performed automatically by the system — there is no need to manually assign tokens when you first set up a storage cluster, or when you resize a cluster.
- For cluster operations that involve transferring data across nodes — such as data repair operations or replacing a failed disk or host machine — the operations complete faster because data is transferred in small ranges from a large number of other hosts.
- The allocation of different vNodes to each disk means that failure of a disk affects only a known portion of the data on the host machine. Other vNodes assigned to the host's other disks are not impacted.
- The allocation of different vNodes to each disk, coupled with [storage policies for replication or erasure coding](#), enable you to efficiently and safely store S3 object data without the overhead of RAID.

Chapter 2. Starting, Stopping, and Checking Status of Services

Subjects covered in this section:

- *Introduction (immediately below)*
- *"Using `hscctl` to Start, Stop, or Check Status of Services (Scope: Node, Data Center, Region, or System)" (page 91)*
- *"Using the CMC to Start, Stop, or Check Status of Services (Scope: Node)" (page 93)*
- *"Using `systemctl` to Start, Stop, or Check Status of Services (Scope: Node)" (page 94)*
- *"Shutting Down or Rebooting a Node" (page 95)*
- *"Automatic Service Start on Node Boot-Up or After a Service Crash" (page 95)*

There are a few methods for starting and stopping services on your HyperStore nodes. Your choice of a method comes down to:

- Preference and convenience. For example, if you're using `hscctl` to make a configuration change and then you need to restart a service to implement that change, it makes sense to also use `hscctl` to do the service restart.
- Scope. The different methods for starting and stopping services vary in their supported scope of action - their ability to execute the desired service action on a single node, a single data center, a single service region, or all nodes in your entire HyperStore system. Only the `hscctl` method supports all possible scopes.

Note If you have the HyperStore File Service running on auxiliary nodes and you need to start or stop that service or its corresponding K3s service, contact Clouddian Support for guidance. The methods described below do not work for the File Service.

2.1. Using `hscctl` to Start, Stop, or Check Status of Services (Scope: Node, Data Center, Region, or System)

To start, stop, or restart a service on all nodes in your whole HyperStore system, run the applicable `hscctl` command on the Config Controller node:

```
# hscctl service <start|stop|restart> <service> <scope>
```

The supported `<service>` values are:

- all
- cassandra
- clouddianagent
- cmc
- coredns (which underlies the [hsDNS feature](#))
- credentials
- firewall

- hyperstore
- iam
- k3s

Note For the *k3s* service -- which runs on File Service and Search Service nodes, if you've installed those optional services -- only a status check is supported. Starting, stopping, and restarting of the File and Search services, as needed for high availability, is managed automatically by K3s (which is a lightweight Kubernetes implementation).

- qos
- redismon
- s3
- s3-select (which supports the S3 Select feature -- i.e., the S3 API's *SelectObjectContent* call)
- sqs

The *<scope>* is one of *--nodes=<hostname or ALL>* or *--datacenters=<datacenter name>* or *--regions=<region name>*. Note that the *--datacenters* and *--regions* options are of potential value only if you have a multi-DC or multi-region HyperStore system, and can otherwise be ignored.

For example, to restart the S3 Service on all nodes in your HyperStore system:

```
# hscctl service restart s3 --nodes=ALL
```

To stop the Cassandra service on all nodes in Region1:

```
# hscctl service stop cassandra --regions=Region1
```

To restart both the S3 Service and the CMC on all nodes in your HyperStore system:

```
# hscctl service restart s3 cmc --nodes=ALL
```

To stop the HyperStore Service only on the node with hostname *host1*:

```
# hscctl service stop hyperstore --nodes=host1
```

Note Remember that the scope option name is plural (*--nodes*) even if you're specifying just one node.

To stop **all** of the services on node6:

```
# hscctl service stop all --nodes=node6
```

To start all of the services on node6 (after having stopped all the services, but not rebooted the node):

```
# hscctl service start all --nodes=node6
```

Note Having the *<hostname>* in these *stop all* and subsequent *start all* commands be the hostname of the Config Controller node itself is allowed, since *hscctl* continues to work on the node even after you've run the *stop all* command.

Whatever service action command and scope you use, the response will show you the progress and then the ultimate result. For example:


```
# hsctl service restart s3 --nodes=vm222
REGION  DATACENTERS  NODES  SERVICE ACTION  RESULT
region1 N/A          vm222  s3      restart OK
```

You can also use *hsctl* to **check on the current status of a service**:

```
# hsctl service status <service>
```

For example:

```
# hsctl service status cmc
REGION  DATACENTER  NODE  SERVICE STATE
region1 DC1      vm222  cmc    RUNNING
region1 DC1      vm223  cmc    STOPPED
region2 DC1      vm224  cmc    RUNNING
```

For another example:

```
hsctl service status all
REGION  DATACENTER  NODE  SERVICE      STATE
region1 DC1      vm222  cmc          RUNNING
region1 DC1      vm222  credentials  RUNNING
region1 DC1      vm222  iam          RUNNING
...
```

For inline help about the *hsctl service* commands:

```
# hsctl service --help
```

Or for inline help about a particular service sub-command (such as *start* or *stop* or *restart* or *status*):

```
# hsctl service <command> --help
```

2.2. Using the CMC to Start, Stop, or Check Status of Services (Scope: Node)

You can start, restart, or stop most services on a selected node by using the "Services Status" section of the CMC's **Node Status** page.

STATUS	SERVICE	IP ADDRESSES	LAST UPDATE	ACTION
✓	Admin	10.0.0.10	Mar-09-2018 13:16	RESTART
✓	Cassandra	10.0.0.10	Mar-09-2018 13:16	RESTART
⊙	HyperStore	10.0.0.10	Mar-09-2018 13:16	START
✓	Redis Mon (Primary)	10.0.0.10	Mar-09-2018 13:16	RESTART STOP
✓	Redis Cred (Master)	10.0.0.10	Mar-09-2018 13:16	RESTART STOP
✓	Redis Qos (Master)	10.0.0.10	Mar-09-2018 13:16	RESTART STOP
✓	S3	10.0.0.10	Mar-09-2018 13:16	RESTART

RESTART ALL

This same CMC display shows you the current status of services.

For more information including limitations, see [Node Status](#).

2.3. Using *systemctl* to Start, Stop, or Check Status of Services (Scope: Node)

You can start, restart, or stop a service on a node by logging into the node and using *systemctl* commands. These commands can be run from any directory on the target node.

```
# systemctl start|restart|stop|is-active <servicename>
```

For example, to restart the S3 service:

```
# systemctl restart cloudian-s3
```

To check the status of the S3 service:

```
# systemctl is-active cloudian-s3
active
```

If you are using the HyperStore Shell

The [HyperStore Shell \(HSH\)](#) supports using *systemctl* commands such as those above.

The table below shows all HyperStore services and their corresponding *<servicename>*.

Service	<servicename>
S3 Service and Admin Service. These two services start and stop together.	<i>cloudian-s3</i>
S3 Select Service. (This supports the S3 Select feature.)	<i>cloudian-s3-select</i>
IAM Service and STS Service. These two services start and stop together.	<i>cloudian-iam</i>
SQS Service	<i>cloudian-sqs</i> (by default this service is disabled; see the introduction to the SQS section in the Cloudian HyperStore AWS APIs Support Reference)
Cassandra Service (Metadata DB)	<i>cloudian-cassandra</i>
HyperStore Service	<i>cloudian-hyperstore</i>
Redis Credentials Service (Credentials DB)	<i>cloudian-redis-credentials</i>
Redis QoS Service (QoS DB)	<i>cloudian-redis-qos</i>
Redis Monitor	<i>cloudian-redismon</i>
Cloudian Management Console	<i>cloudian-cmc</i>
Cloudian Monitoring Agent	<i>cloudian-agent</i>
CoreDNS (which underlies the hsDNS feature)	<i>cloudian-coredns</i>

2.4. Shutting Down or Rebooting a Node

You can shut down or reboot a HyperStore host machine by logging into the machine and using `systemctl` commands.

To shut down the host:

```
# systemctl poweroff
```

To reboot the host:

```
# systemctl reboot
```

*If you are using the **HyperStore Shell***

The [HyperStore Shell \(HSH\)](#) supports using `systemctl` commands such as those above.

IMPORTANT ! If you are rebooting multiple nodes, make sure that each node is back up for at least one second before moving on to reboot the next node.

2.5. Automatic Service Start on Node Boot-Up or After a Service Crash

By default all of the HyperStore services on a host are configured to automatically start when the host is booted up. The controlling configuration setting is "[common.services.enabled](#)" (page 750).

Also, by default the system is configured so that a service that crashes on a node is automatically restarted on that node (this auto-restart feature applies only to the S3 Service, Admin Service, HyperStore Service, and Cassandra Service). The controlling configuration setting is "[common.services.autoRestart](#)" (page 749).

This page left intentionally blank

Chapter 3. Upgrading Your HyperStore Software Version

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Preparing to Upgrade Your System" \(page 97\)](#)
- ["Upgrading Your System" \(page 99\)](#)
- ["Actions to Complete After Upgrade" \(page 101\)](#)
- ["Installing a HyperStore Patch" \(page 102\)](#)
- ["Upgrading the HyperStore File Service" \(page 102\)](#)

The instructions that follow are for upgrading to HyperStore version 8.2 **from HyperStore version 8.1 or newer**. Also your existing HyperStore hosts must be running Rocky Linux 8.9 or 8.10 (or RHEL 8.9 or 8.10). **HyperStore version 8.2 does not support running on CentOS.**

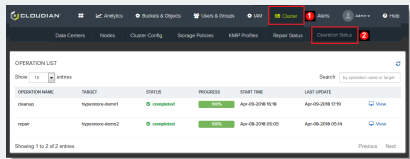
If you are already running HyperStore version 8.2 and are installing a patch release (with a 4-digit release number), you can jump directly to ["Installing a HyperStore Patch" \(page 102\)](#).

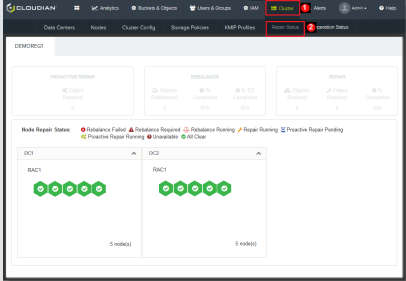
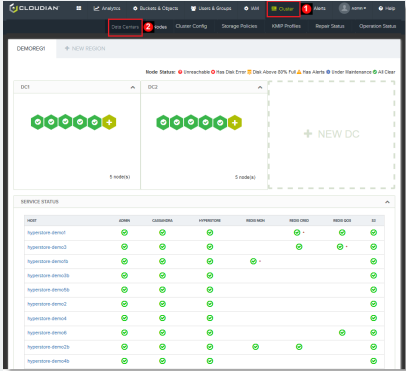
IMPORTANT ! Contact Cloudbian Support before upgrading if any of these conditions apply to your existing HyperStore system:

- * Your current HyperStore version is earlier than version 8.1.
- * One or more of your existing HyperStore hosts has less than 128GB RAM.
- * Your HyperStore hosts are using Xen or are in Amazon EC2.
- * Your HyperStore data disks are using Logical Volume Manager (LVM).

3.1. Preparing to Upgrade Your System

Before upgrading your HyperStore system, log into the CMC and under the **Cluster** tab use the system status display pages to make sure that your system is in a fully healthy, unrestricted state and that no major operations are in progress.

CMC Page	What to Look For
Operation Status 	Check whether there are any operations currently in progress in any of your service regions. • If any repair or repairrec operation is in progress when you launch the upgrade script, the script will fail in the pre-check stage and will not make any changes to your system. So before launching the upgrade you must wait for any in-progress repair to complete (or alternatively you can stop the repair by using the "-stop" option that is supported by the hsstool repair and hsstool repairrec commands). • If any cleanup, cleanupec, decommission, or rebalance operation is in progress, Cloudbian recommends that you wait until the operation completes before you perform the upgrade (or in

CMC Page	What to Look For
	the case of cleanup you can stop the cleanup by using the <code>-stop</code> option that is supported by the hsstool cleanup and hsstool cleanupc commands). Make sure there are no proactive repairs in progress. Note that in-progress proactive repairs will not display in the Operation Status page but will display in the Repair Status page. If a proactive repair is in progress, either wait until it completes or alternatively you can stop the proactive repair using the hsstool proactiverepair command with the <code>-stop</code> option. If a proactive repair is in progress, the upgrade script will fail in the pre-check stage and will not make any changes to your system.
	Make sure that all nodes and all services are up and that no node is in a restricted status. If a service is down or a node is in one of the restricted statuses noted below, the upgrade script will fail in the pre-check stage and will not make any changes to your system. - If any services are stopped, start them. For instructions see "Starting, Stopping, and Checking Status of Services" (page 91). - If any node is in Maintenance Mode, take it out of Maintenance Mode. For instructions see "Stopping Maintenance Mode" (page 490). - If any node is in "stop-write" condition, contact Cloudian Support before performing the upgrade. Also, if the Data Centers page indicates that any node has alerts, go to the Node Status page and then select that node and review its alerts. Resolve any serious issues before proceeding with the upgrade.

Note The upgrade script will automatically disable the auto-repair and proactive repair features -- so that no new repairs kick off during the upgrade -- and then after the upgrade completes the script will automatically re-enable the auto-repair and proactive repair features.

3.1.1. Additional Upgrade Preparation If Your System Currently Has Failed Disks

By default the HyperStore upgrade script will abort if it detects that any disks on your HyperStore nodes are failed or disabled. If you want to perform a HyperStore version upgrade while there are failed or disabled disks in your current system, take the following preparation steps **before starting the upgrade procedure**:

1. On the Config Controller node, in the staging directory for your **current** HyperStore system, open this file in a text editor:

```
CloudianInstallConfiguration.txt
```

2. In the file, change `INSTALL_SKIP_DRIVES_CHECK=false` to `INSTALL_SKIP_DRIVES_CHECK=true`. Then save and close the file.

3.2. Upgrading Your System

To perform the upgrade to HyperStore version 8.2:

1. Download the HyperStore product package (`CloudianHyperStore-8.2.bin` file) from the Cloudian Support portal into a working directory on the Configuration Master node (such as `/tmp` or your home directory -- **do not use the installation staging directory from your existing HyperStore system**). You can also download from the Support portal the signature file (`.sig` file) corresponding to the product package file -- you will need the signature file if you are using the HyperStore Shell to perform the upgrade.

Note If you are using the **HyperStore File Service** (which provides an SMB and/or NFS front end) in your existing HyperStore system, you will need to upgrade the File Service in a separate procedure after completing your HyperStore system upgrade (as described in "[Upgrading the HyperStore File Service](#)" (page 102)). There is a separate product package file for doing the File Service upgrade and you can download this from the Cloudian Support portal as well as downloading the HyperStore product package.

2. In the working directory run the commands below to unpack the HyperStore package:

```
# chmod +x CloudianHyperStore-8.2.bin
# ./CloudianHyperStore-8.2.bin
```

This creates a new installation staging directory named `/opt/cloudian-staging/8.2`, and extracts the HyperStore package contents into the staging directory.

If you are using the HyperStore Shell to perform the upgrade

If you are using the [HyperStore Shell \(HSH\)](#) to perform the upgrade, make sure you have in your working directory the signature file (`.sig` file) corresponding to the product package file, as well as the product package file. Then from the shell, run these commands to unpack the HyperStore package (rather than the commands stated above):

```
$ chmod +x CloudianHyperStore-8.2.bin
$ hsr --root CloudianHyperStore-8.2.bin
```

Note To perform the upgrade using the HSH you must be an HSH [Trusted user](#).

3. **Change into the new installation staging directory** and then launch the installer:

```
# cd /opt/cloudian-staging/8.2
# ./cloudianInstall.sh
```

```
Cloudian HyperStore(R) 7.5.3 Installation/Configuration
-----

0 ) Run Pre-Installation checks
1 ) Install Cloudian HyperStore
2 ) Cluster Management
3 ) Upgrade From a Previous Version
4 ) Advanced Configuration Options
5 ) Help
x ) Exit

Choice: █
```

If you are using the HyperStore Shell to perform the upgrade

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Configuration Master node you can launch the installer with this command:

```
$ hspkg install
```

Once launched, the installer's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

4. From the installer main menu enter "3" for "Upgrade from a Previous Version". Then at the prompt, confirm that you wish to continue with the upgrade.

When the upgrade process proceeds it upgrades one node at a time -- by shutting down the node, updating the packages on the node, and then restarting the node and the services on the node -- **until all nodes are upgraded**. Messages in the terminal will indicate the upgrade progress.

After the upgrade successfully completes, proceed to ["Actions to Complete After Upgrade"](#) (page 101).

Note

- * Once you've started the upgrade, you cannot `<ctrl>-c` out of it.
- * If you have initiated the upgrade through a remote terminal, and the connection between the terminal and the Configuration Master node is subsequently lost, the upgrade will continue.

IMPORTANT ! After the upgrade, do not delete the staging directory that was created when you unpacked the product package file (`/opt/cloudian-staging/8.2`). HyperStore will continue to require certain files in this directory throughout the time that you are using this HyperStore version.

3.2.1. Upgrade Errors

If an error occurs in the early, "pre-upgrade" stage -- as the configuration management system is updated and health checks are performed, before the upgrade of software packages starts on each node one by one -- roll-back is automatically implemented and a message will display in the terminal indicating the error and the roll-

back. If an error occurs later in the process during the upgrades of individual nodes, automatic retries are executed. The terminal will display information about such errors and the retry attempts.

If the upgrade is unable to advance beyond the pre-upgrade stage, or if it fails in the process of upgrading one or more nodes, the terminal will display basic information about the failure and you can get more details from the *cloudian-installation.log* file and the *cloudian-upgrade.log* file, both of which are generated in the installation staging directory. Try to resolve the problem(s) on the node(s) that failed to upgrade, and then run the upgrade process again (action number **3** from the installer's main menu). The upgrade process will resume from where it left off the first time.

The upgrade process also generates an *upgrade-logNconfig-<time-stamp>.tgz* "S.O.S" tar file (which packages together multiple upgrade-related files) in the staging directory that you can provide to Cloudian Support in the event that you need assistance in resolving any upgrade problems.

3.3. Actions to Complete After Upgrade

After all HyperStore nodes have been upgraded, verify that all services are running and that the HyperStore version is now 8.2:

1. On the Config Controller node, use the *hsctl* command line tool to confirm that all services are running:

```
# hsctl service status all
```

Note The SQS Service is disabled by default, but all other services should be running.

2. Use *hsctl* to confirm that your HyperStore version is now 8.2:

```
# hsctl version
```

3. Use the CMC to check on your upgraded system:
 - On the **Node Advanced** page, select command type "Info" then execute the "repairqueue" command to **verify that auto-repair is enabled** for replica, EC, and Cassandra data. (The installer disables auto-repair prior to executing the upgrade and then re-enables auto-repair at the end of the upgrade process.)
 - On the **Manage Users** page, confirm that you can retrieve users.
 - Log out of the CMC as system admin and log back in as a regular user, and then confirm that you can successfully download and upload objects.

Also after upgrade:

- If in your prior HyperStore version you had made any **customizations to the branding of the CMC interface**, only your customized logos and customized application name will be retained after the upgrade. You will need to re-implement any changes that you had made to the browser tab title or the color scheme or the login page banner, by again following the instructions for **"Rebranding the CMC UI"** (page 266).
- If you disabled the failed disk check before performing your upgrade (as described in **"Additional Upgrade Preparation If Your System Currently Has Failed Disks"** (page 98)), note that after you've completed the upgrade, in the new instance of *CloudianInstallConfiguration.txt* in the staging directory for your new HyperStore version the *INSTALL_SKIP_DRIVES_CHECK* setting is set back to its default of *false*. So the next time you upgrade your HyperStore version the check for failed drives will be executed, unless you once again disable the check by changing that setting to *true*.

- If you are using the **HyperStore File Service** in your system, after you've completed the main HyperStore system upgrade you must separately upgrade the File Service as described in "[Upgrading the HyperStore File Service](#)" (page 102). Although the File Service will continue to work during the time between completing the HyperStore system upgrade and the time you upgrade the File Service, certain types of configuration updates will not propagate to the File Service nodes, including HyperStore node status information tracked by *hsDNS* (if you are using *hsDNS*). Therefore you should promptly upgrade your File Service after completing your HyperStore system upgrade.

3.4. Installing a HyperStore Patch

To install a HyperStore patch release (with a 4-digit release number), follow these steps on the Config Controller node:

1. Log in to the Config Controller node and change into your current HyperStore version's installation staging directory. (Note: installing a patch does not create a new installation directory.)
2. Put the patch release package file in the installation staging directory if you haven't already done so.
3. Run these commands to install the patch:

```
# chmod +x <patch-package-name>
# hscctl patch install ./<patch-package-name>
```

For example:

```
# chmod +x CloudianHyperStorePatch-1.2.3.4.bin
# hscctl patch install ./CloudianHyperStorePatch-1.2.3.4.bin
```

In the event that you need to revert an installed patch you can do so with this command:

```
# hscctl patch revert
```

This reverts the most recently installed patch.

3.5. Upgrading the HyperStore File Service

If you are using the [HyperStore File Service](#) in your system, after you've completed the main HyperStore system upgrade to version 8.2 you must separately upgrade the File Service to version 8.2. Although the File Service will continue to work during the time period between completing the HyperStore system upgrade and the time you upgrade the File Service, certain types of configuration updates will not propagate to the File Service nodes during this time period, including HyperStore node status information tracked by *hsDNS* (if you are using *hsDNS*). Therefore you should promptly upgrade your File Service after completing your HyperStore system upgrade.

If you originally set up the HyperStore File Service when running HyperStore version 7.5.1 or later, you can use the *hscctl* command line tool to upgrade the File Services as described below. (If you originally set up the HyperStore File Service when running a HyperStore version older than 7.5.1, contact Clodian Support rather than following the procedure below.)

On your HyperStore Config Controller node run this *hscctl* command:

```
# hscctl file upgrade --old-master <IP address> <package>
```

where:

- `<package>` is the HyperStore File Services 8.2 release package name (including absolute path if the package is not in your current directory). This package file, which you can obtain from Cloudian Support if you don't already have it, is a different file than the main HyperStore package file that you used to upgrade to HyperStore 8.2.
- `--old-master <IP address>` is the IP address of whichever of your existing File Service nodes is currently serving as the Salt master. If you are unsure of the IP address of the existing Salt master you can run this Salt command on any of your existing File Service nodes to retrieve it:

```
salt-call grains.item master
```

After the upgrade of your File Services completes, log into the CMC and confirm that:

- The [Data Centers](#) page shows your File Service cluster (called a [File Service "pool"](#) in HyperStore 8.2 and later) at the bottom of the page.
- The [File Service Settings](#) page displays with configurable settings for your File Service pool.
- The [Add User](#) function includes an option to enable File Services for a new user and the [Edit User](#) function includes an option to enable File Services for an existing user.

This page left intentionally blank

Chapter 4. Setting Up Administration Features

4.1. HyperStore Shell (HSH)

4.1.1. HyperStore Shell (HSH) Feature Overview

The HyperStore Shell (HSH) is a restrictive command-line interface that allows administrators to log into and administer HyperStore nodes without having or needing *root* access to the nodes. A user logged into the HyperStore Shell can run common HyperStore commands and tools such as *hsstool* or *cloudianInstall.sh* as well as being able to run a limited range of Linux OS commands. But the HSH is more restrictive, and therefore more secure, than having administrators log into HyperStore nodes as *root* and use a standard Linux shell. With the HSH each administrator has their own unique login ID and password; the commands that administrators can execute are limited by the shell; and each administrator's actions are recorded to an audit log.

The HSH is present on each HyperStore node but **by default the HSH is disabled**. For information on enabling the HSH, and provisioning HSH users, see ["Enabling the HSH and Managing HSH Users" \(page 106\)](#). That section also describes an option to disable *root* password access to HyperStore nodes, so that administration of the nodes is performed exclusively by way of the HSH.

For information on the HyperStore commands and Linux OS commands supported by the HSH, see ["Using the HSH" \(page 111\)](#).

4.1.1.1. HSH Logging

Every user login to the HSH, and every command that an HSH user runs while logged in, is recorded to a dedicated HSH log. Also recorded are commands that an authenticated HSH user runs remotely, without initiating a login session.

Each HyperStore node has its own HSH log, recording HSH logins and commands run on that node. The log is configured as append only, and HSH users cannot modify this configuration.

For more information on HSH logging, see ["HyperStore Shell \(HSH\) Log" \(page 676\)](#).

4.1.1.2. The HSH and "Certified" Object Lock

If you want to use the "Certified" type of Object Lock, you must first enable the HSH and disable the root password. With the "Certified" type of Object Lock, the system will not allow the creation of Object Lock enabled buckets until after the HSH is enabled and the root password is disabled.

For information about Object Lock, see ["Object Lock Feature Overview" \(page 221\)](#).

For information about enabling HSH and disabling the root password, see ["Enabling the HSH and Managing HSH Users" \(page 106\)](#).

4.1.2. Enabling the HSH and Managing HSH Users

Subjects covered in this section:

- ["Enabling the HSH" \(page 106\)](#)
- ["Adding HSH Users" \(page 106\)](#)
- ["Elevating a Regular HSH User to the "Trusted" Role" \(page 108\)](#)
- ["Deleting an HSH User" \(page 109\)](#)
- ["Disabling the root Password" \(page 109\)](#)
- ["Configurable HSH Idle Timeout" \(page 110\)](#)

4.1.2.1. Enabling the HSH

By default the HyperStore Shell (HSH) is installed but disabled.

Note The exception is that HyperStore appliances delivered for new deployments in certain industry sectors with stringent security requirements may arrive on site with HSH already enabled (and the root password already disabled). If you're unsure whether this applies to you, consult with your Cloudbian representative.

To enable the HSH on all of your HyperStore nodes do the following:

1. Log into the Config Controller node as the *root* user.
2. Check to confirm that the HSH is currently disabled by running the following *hscctl* command:

```
# hscctl config get hsh.enabled
False
```

3. Set *hsh.enabled* to *true*:

```
# hscctl config set hsh.enabled=true
```

4. Push the configuration change out to the cluster:

```
# hscctl config apply hsh
```

5. Confirm that the HSH is now enabled:

```
# hscctl config get hsh.enabled
True
```

The HSH is now enabled in your system, but by default no users are able to log into the HSH and use it. To provision users who can use the HSH, see ["Adding HSH Users" \(page 106\)](#) below.

4.1.2.2. Adding HSH Users

HSH users map to your System Admin users in the CMC. For each System Admin user in the CMC, HyperStore will automatically create a corresponding HSH user account if triggered to do so as described below.

CMC System Admin User Type	How to Trigger Creation of Corresponding HSH User
The CMC default System Admin user, with user name "admin".	After installing or upgrading to HyperStore 7.2 or later, log into the CMC as the "admin" user and change the "admin" user's password . (Alternatively,

CMC System Admin User Type	How to Trigger Creation of Corresponding HSH User
	the password can be changed through the Admin API -- see the "user" section in the <i>Cloudian HyperStore Admin API Reference</i> .) This password change causes the system to create a corresponding HSH user.
Additional CMC System Admin users, created in HyperStore 7.1.x or earlier.	After the system has been upgraded to HyperStore 7.2 or later, a legacy CMC System Admin User can log into the CMC and change their password. (Alternatively, the password can be changed through the Admin API.) This password change causes the system to create a corresponding HSH user.
Additional CMC System Admin users, created in HyperStore 7.2 or later.	When an additional CMC System Admin user is created in HyperStore 7.2 or later (either in the CMC or through the Admin API), the system automatically creates a corresponding HSH user.

When HyperStore creates an HSH user corresponding to a CMC System Admin user:

- The **HSH user name is the CMC user name prefixed by "sa_"**. For example, for CMC user "admin" the HSH login name is "sa_admin"; and for CMC user "tom" the HSH login name is "sa_tom".
- The **HSH user login password is the same as the CMC user login password**. The HSH user login password cannot be managed separately from -- or differ from -- the corresponding CMC user login password. If an HSH user wants to change their password they should change their CMC password, and their HSH password will automatically change to match their new CMC password.

Note If you have **LDAP authentication** enabled for the System Admin group, then when HyperStore creates an HSH user corresponding to a CMC System Admin user the HSH user name will **not** have an "sa_" prefix (instead it will be the same as the CMC user name); and when the user logs into the HSH they will use their LDAP credentials and the system will verify those credentials against your LDAP service. Note that if a local Unix user account with that same user name already exists, a new HSH user account will not be created (the existing local user account will not be overwritten). For more information on LDAP authentication see "[LDAP Integration](#)" (page 278) and especially the sub-section "[LDAP Authentication of System Administrators and HSH Users](#)" (page 281).

Once an HSH user has been created, that user can use SSH to log into any HyperStore node. Upon login, the user's shell will be the HyperStore Shell. The prompt will appear as follows:

```
<username>@<hostname>$
```

For example:

```
sa_admin@hyperstore1$
```

You can confirm that you are in the HyperStore Shell by typing *help*:

```
sa_admin@hyperstore1$ help
HyperStore Shell
Version: 1.0.1-2, d6b3c8d46ecaaaa69b62af25c4e7d4270f8b4d7e(otp protocol: 1)
```

```
Commands:
```

```
...
```

For information on using the HyperStore Shell see ["Using the HSH" \(page 111\)](#).

Note HyperStore does not create corresponding HSH users for CMC Group Admin level users (or for regular users). Only System Admin level users are allowed HSH access.

4.1.2.3. Elevating a Regular HSH User to the "Trusted" Role

The HSH supports two types of HSH users:

- Regular HSH users
- Trusted HSH users (HSH users who have been granted the "Trusted" role)

While regular HSH users can run **most** of the commands that the HSH supports, Trusted HSH users can run **all** of the commands that the HSH supports. Put differently, there are some commands that only Trusted HSH users can run. For information about supported commands and which ones are available only to Trusted users, see ["Using the HSH" \(page 111\)](#).

The **HSH user *sa_admin*** -- the HSH user corresponding to the CMC default System Admin user -- **is automatically a Trusted HSH user**. By contrast, HSH users corresponding to additional System Admin users that have been created in the CMC are by default regular HSH users who do not have the Trusted role.

As a Trusted user the *sa_admin* user can elevate one or more regular HSH users so that they too are Trusted:

1. Log into the Config Controller node as the *sa_admin* user. Upon login you will be in the HyperStore Shell.
2. Run this command to add a regular HSH user to the Trusted role:

```
$ hspkg role -a <username> trusted
```

Be sure to specify the HSH user name (including the *sa_* prefix), not the CMC user name.

For example:

```
$ hspkg role -a sa_admin2 trusted
hsh.roles.trusted.users=sa_admin2
```

Note The response shows the current list of users who have been explicitly granted the Trusted role (which is just one user in the example above). The *sa_admin* user, who is automatically Trusted, will not appear in this list.

3. If you want to elevate any other regular HSH users to the Trusted role at this time, run the command from Step 2 again, with another HSH user name.
4. Apply the configuration change.

```
$ hctl config apply hsh
```

Once additional HSH users have been elevated to the Trusted role, then those Trusted users are also able to elevate regular HSH users to the Trusted role. That is, any Trusted HSH user has the ability to elevate regular HSH users to the Trusted role.

To see the current list of HSH users who have been explicitly granted the Trusted role, any Trusted user can run this command:


```
$ hspkg role trusted
hsh.roles.trusted.users=sa_admin2
```

Note again that the response will exclude the *sa_admin* user, who is automatically Trusted.

To remove an HSH user from the list of Trusted users, a Trusted user can run these commands:

```
$ hspkg role -d <username> trusted
$ hscctl config apply hsh
```

This does not delete the HSH user; it only demotes them to being a regular HSH user rather than a Trusted user.

Note If an HSH user that you elevate to the Trusted role (or remove from the Trusted role) is logged in to the HSH when you make the change, the change in their role status will not take effect until after the user logs out and then logs back in again.

4.1.2.4. Deleting an HSH User

If a System Admin user is deleted or suspended (made inactive) in the CMC or through the Admin API, the system automatically deletes that user's HSH user account.

In the case of a System Admin user who is made inactive, and then subsequently made active again, after being made active again that user must change their password in the CMC or through the Admin API in order to trigger the system to recreate a corresponding HSH account for the user.

The system does not support deleting or disabling the HSH account of a CMC System Admin user while their CMC account remains active.

4.1.2.5. Disabling the *root* Password

Optionally, you can disable the *root* account password on all HyperStore nodes so that no user has *root* password access to those nodes. You can do this if for security reasons it's desirable to have all administrators use only the restrictive HyperStore Shell when logging into and administering HyperStore nodes.

Note Disabling the *root* password is **required** if you want to use the ["Certified" type of Object Lock](#).

Before you can disable the *root* account password:

- HSH must be enabled (see ["Enabling the HSH"](#) (page 106))
- An HSH user account must be created for the CMC's default System Admin user (see ["Adding HSH Users"](#) (page 106))

Also before you disable the *root* account password, it's important to be aware that:

- If you disable the *root* password, and then you subsequently want *root* password access to your HyperStore nodes again, you will need to contact Cloudian Support for assistance. Once you disable *root* password access to HyperStore nodes **you cannot regain *root* password access without assistance from Cloudian Support.**
- Disabling the *root* password will prevent users from logging in to a HyperStore node as *root* using a password but it **will not prevent a *root* user from accessing a HyperStore node using an SSH key**, if in your environment SSH key access has been set up for the *root* user.

- Disabling the *root* password **will have no effect on any users who were granted *sudo* privileges** before you disabled the *root* password. Such users (if any) will continue to have the privileges granted to them by a node's *etc/sudoers* configuration. To help secure your HyperStore nodes, Cloudian, Inc. recommends that you manually remove the *sudo* privileges of all users, on all HyperStore nodes (including yourself if you have *sudo* privileges). Also if you have granted elevated permissions to any user or application through any other mechanism, remove those permissions.

To disable *root* password access to all HyperStore nodes:

1. As *root*, log into the Config Controller node and then change into the installation staging directory (*/opt/cloudian-staging/8.2*).

Note You must be logged in as *root* to disable the *root* password -- you cannot do it while logged in as an HSH user.

2. Launch the HyperStore installer.

```
# ./cloudianInstall.sh
```

3. In the installer main menu enter **4** for "Advanced Configuration Options", then at the next menu enter **m** for "Disable the root password".
4. Follow the prompts to disable the *root* password.

After exiting the installer, log out from the node. Then try to log back in as *root*, using the *root* password -- the login attempt should fail. Then log in as *sa_admin* or another HSH user, with that user's password. The login should succeed, and you should be in the HyperStore Shell.

Regaining *root* Password Access After Having Disabled It

If you disable *root* password access to your HyperStore nodes as described above, the only way to regain *root* password access is to contact Cloudian Support for assistance. This *root* password access will be temporary.

4.1.2.6. Configurable HSH Idle Timeout

The HyperStore Shell has a default idle timeout of 30 minutes. This means that by default an HSH session will terminate automatically if the HSH user has been completely inactive for 30 minutes. The idle timer:

- Is reset to zero whenever the HSH user presses any key. For example if the HSH user has been completely idle for 20 minutes and then they press any key, the timer is reset and the user could then be idle again for up to 30 minutes before their session is automatically terminated.
- Does not apply to long-running commands or utilities that the HSH user has initiated.

To change the idle timeout, on the Config Controller node run the following *hsctl* commands:

```
$ hsctl config set hsh.idleTimeout=<string>
$ hsctl config apply hsh
```

- Specify the *<value>* as *<n>s* for number of seconds or *<n>m* for number of minutes or *<n>h* for number of hours -- for example *90s* or *5m* or *2h*.
- The minimum allowed value is one minute and the maximum allowed value is twelve hours.
- This configuration can be performed by the *root* user, or while in the HSH if the HSH user has the Trusted role. If performed while in the HSH, the idle timeout configuration change will not apply until the HSH user logs out of the HSH and then logs back in again.

To check the current idle timeout configuration value:

```
$ hscctl config get hsh.idleTimeout
```

Note In the rare case of a root mode HSH session -- when assisted by Cloudbian Support as noted in ["Regaining root Password Access After Having Disabled It" \(page 110\)](#) -- there is a dedicated root mode idle timeout which by default is 15 minutes. This timeout can be configured as described above, except that the setting name is `hsh.rootMode.idleTimeout`. This cannot be set to a value larger than the current `hsh.idleTimeout` setting value.

4.1.3. Using the HSH

Subjects covered in this section:

- ["Starting and Ending an HSH Session" \(page 111\)](#)
- ["HyperStore Commands and Utilities Supported by the HSH" \(page 112\)](#)
- ["Linux Commands and Utilities Supported by the HSH" \(page 115\)](#)
- ["Platform-Specific Commands Supported by the HSH" \(page 116\)](#)
- ["Using sftp to Put or Get a File Via an HSH Account" \(page 117\)](#)
- ["Notes On Command Usage" \(page 117\)](#)

4.1.3.1. Starting and Ending an HSH Session

Once the HyperStore Shell (HSH) has been enabled and your HSH user account has been created (as described in ["Enabling the HSH and Managing HSH Users" \(page 106\)](#)), you can SSH into any HyperStore node using your HSH user name and password. Upon login to the node, your shell will be the HyperStore Shell. The prompt will appear as follows:

```
<username>@<hostname>$
```

For example:

```
sa_admin@hyperstore1$
```

To confirm that you are in the HyperStore Shell and **to view a list of HSH commands**, type `help`:

```
$ help
HyperStore Shell
Version: 1.0.1-2, d6b3c8d46ecaaa69b62af25c4e7d4270f8b4d7e(otp protocol: 1)

Commands:
...
```

To view HSH command inline help type `<command> --help` (or `<command> -h`):

```
$ hslog --help
View protected log files under the /var/log directory.
Usage:
hslog [flags] FILENAME
Flags:
-h, --help  help for hslog
```

To end your HSH session and disconnect from the HyperStore node, type `exit`.

```
$ exit
```

Note

- * On each node an HSH user's home directory is `/home/<username>` -- for example `/home/sa_admin`.
- * By default the HSH has an idle timeout of 30 minutes. For more information see ["Configurable HSH Idle Timeout"](#) (page 110).

Reattaching to an HSH Session

If you are logged into the HSH on a node and your SSH connection gets cut:

- Any long-running commands that were in-progress in that HSH session will continue to run
- You can log back into the HSH on that same node and you will be given the choice to reattach to the existing session or to start a new session

You can also manually detach from a session by using the keystroke sequence `ctrl-g d`. Then when you log in again you can choose whether to reattach to the existing session or to start a new session.

Note If an HSH user gets disconnected -- or intentionally detaches -- from a session, and then logs in and starts a new session, the user can have multiple concurrent sessions on a node. The system sets a limit of 9 concurrent sessions for a single HSH user on a node.

To end an existing session from which you are detached, reattach to it and then type `exit` on the command line.

4.1.3.2. HyperStore Commands and Utilities Supported by the HSH

The HyperStore Shell supports the use of these commands that are specific to HyperStore. Commands marked as "(requires Trusted role)" are available only to HSH users who have been granted the Trusted role (for more information on this role see ["Elevating a Regular HSH User to the "Trusted" Role"](#) (page 108)).

Note Do not precede these commands with a path -- for example, run `hsstool` as simply `hsstool` not as `./hsstool`.

Command	Purpose	More Information
<code>cloudian_sysinfo.sh</code>	This command serves the same purpose as the "Collect Diagnostics" function in the CMC.	"Collect Diagnostics" (page 487)
<code>decrypt_private_data.py</code>	Decrypt encrypted log field values in log file copies that have been uploaded to Cloudian Support.	"Maintaining Personal Data Privacy While Using Smart Support and Node Diagnostics" (page 121)
<code>hsctl</code> (requires Trusted role)	HyperStore cluster management tool that supports tasks such as viewing and changing configuration setting values, and stop-	"Using hsctl to Control HyperStore"

Command	Purpose	More Information
	ping, starting, or restarting services. Note that the HSH does not support the <i>hsctl</i> command <i>hsctl cmd</i> .	"Configuration" (page 693) "Using hsctl to Start, Stop, or Check Status of Services (Scope: Node, Data Center, Region, or System)" (page 91)
hslog	View a HyperStore log file	In shell, <i>hslog -help</i> Or for more detailed information see: "Using the HSH to View Logs" (page 691)
hspkg Below is a partial list of sub-commands; for the complete list see the inline help.	Manage HyperStore installation	In shell, <i>hspkg --help</i>
hspkg setup (requires Trusted role)	Launch the HyperStore host setup tool (<i>system_setup.sh</i>)	In shell, <i>hspkg setup --help</i> See also: "Preparing Your Nodes For HyperStore Installation" in the Cloudian HyperStore Installation Guide
hspkg install (requires Trusted role)	Launch the HyperStore installer (<i>cloudianInstall.sh</i>)	In shell, <i>hspkg install --help</i> See also: "Upgrading Your HyperStore Software Version" (page 97)

Command	Purpose	More Information
		"Installer Advanced Configuration Options" (page 847) "Using hscctl to Control HyperStore Configuration" (page 693) Note When launching the installer from the HyperStore Shell, the installer's "Uninstall Cloudian HyperStore" menu option is disabled.
hspkg role (requires Trusted role)	Manage HSH "Trusted" role membership	In shell, <code>hspkg role --help</code> Or for more detailed information see: "Elevating a Regular HSH User to the "Trusted" Role" (page 108)

Command	Purpose	More Information
hspkg version	Check the HyperStore software version	--
hsrun	Run a binary file or script that is cryptographically signed by Cloudian (such as a HyperStore release package file).	In shell, <i>hsrun -help</i>
hsstool	The HSH supports running any <i>hsstool</i> command, such as <i>hsstool repair</i> , <i>hsstool cleanup</i> , and so on	"hsstool" (page 863)
install_appliance_license.sh (requires Trusted role)	Install a new license for a HyperStore Appliance	In shell, <i>install_appliance_license.sh --help</i>
jetty_password.sh	Create a Jetty-obfuscated password, in connection with modifying the HTTP/S Basic Authentication password for the Admin Service.	"HTTP(S) Basic Authentication for Admin API Access" in the Cloudian HyperStore Admin API Reference
nodetool	Cassandra's native node and cluster management tool. Typically you should use <i>hsstool</i> instead.	--
rebrand_cmc.sh (requires Trusted role)	Manage image and resource files for rebranding the CMC interface.	"Rebranding the CMC UI" (page 266)
reset_expired_password (requires Trusted role)	This can be used if working with your Cloudian representative to set up a newly delivered HyperStore appliance.	--
search-mgr.sh	Relevant only if you have deployed the optional HyperStore Search Service.	"Preparing the Metadata Search Feature" (page 236)

4.1.3.3. Linux Commands and Utilities Supported by the HSH

The HyperStore Shell allows the use of **only** the following Linux commands and utilities. No other Linux commands or utilities can be run from within the shell. Commands marked as "(requires Trusted role)" are available only to HSH users who have been granted the Trusted role (for more information on this role see ["Elevating a Regular HSH User to the "Trusted" Role" \(page 108\)](#)).

Note For security reasons, for some commands the HyperStore Shell forbids certain arguments / sub-commands. If you try to run a command with one of its forbidden arguments the shell returns a message indicating that the argument is not permitted.

blkid	host	ping
cat	hostname	ps
cd	iostat	pwd
chmod	ip	rm
chronyc	ipmitool (requires Trusted role)	rmdir
cp	keytool	scp
curl	kill (requires Trusted role)	sftp
date	less	smartctl (requires Trusted role)
df	ls	ssh
diff	lsblk	systemctl
dmesg	lspci	tail
dmidecode	lsscsi	tar
domainname	md5sum	top
du (requires Trusted role)	mdadm (requires Trusted role)	uname
echo	mkdir	unzip
exit	mv	vi
file	netstat	vmstat
grep	ntpq	w
gzip	openssl (requires Trusted role)	wget
head	pgrep	zgrep

4.1.3.4. Platform-Specific Commands Supported by the HSH

The following commands pertain only to HyperStore appliances, and all of these commands require the Trusted role:

- slot_disk_map_1500.sh
- slot_disk_map_4000.sh
- slot_disk_map_hsx.sh
- slot_disk_map_x3650.sh
- sg_ses
- ledctl

The following commands are applicable only to Lenovo host machines, and both require the Trusted role:

- MegaCli64
- storcli64

The following command is applicable only to HPE host machines, and requires the Trusted role:

- `ssacli`

4.1.3.5. Using `sftp` to Put or Get a File Via an HSH Account

You can use an `sftp` client to upload a file to or download a file from a HyperStore node via an HSH account on that HyperStore node, but only if the file is one that the HSH is permitted to manage.

Note that you cannot use `sftp` **from** an HSH account -- when logged into a HyperStore node as an HSH user, you cannot run the `sftp` command on that node:

```
sa_admin@hyperstore1$ sftp sa_admin@10.10.10.12
-hsh: sftp: Command is not permitted or not found by this shell.
Press the TAB key or type 'help' to discover permitted commands.
```

Note For the list of files that the HSH has permission to access, run this command:

```
$ hspkg config --help
```

For more information see [Using the HSH to Manage Configuration Files](#).

4.1.3.6. Notes On Command Usage

Tab completion for command names and file names is supported.

Running multiple commands on one line is supported, using the operators ";", "&&", "||", and/or "|".

For security reasons, the HSH enforces the following **restrictions** on command usage:

- PATH is fixed for each command; you cannot specify a path when running a command.
- `sudo` is not allowed. Instead, commands that need root privilege (such as `systemctl`) are automatically given that privilege inside the HSH. (The HSH accomplishes this by careful management of the effective UID, which is only elevated to root when running a command as root is required.)
- Job control commands such as `ctrl-z` are not allowed.
- When listing files, the use of the wildcard character "*" is not allowed.

4.2. Smart Support

4.2.1. Smart Support and Diagnostics Feature Overview

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Smart Support" \(page 118\)](#)
- ["On-Demand Node Diagnostics" \(page 119\)](#)
- ["Data Privacy" \(page 120\)](#)

HyperStore includes two automated features that help you collaborate with Cloudian Support to keep your system running smoothly: Smart Support and on-demand Node Diagnostics.

4.2.1.1. Smart Support

Smart Support is a feature that enables Cloudian Support to help you maximize the uptime and performance of your HyperStore system. With Smart Support, your HyperStore system automatically transmits detailed system and node performance data to Cloudian Support once a day over a secure internet connection. Cloudian Support applications and personnel continually analyze this data to detect critical issues. Cloudian Support notifies you to make you aware of such issues and, if appropriate, to advise you on the actions that should be taken to keep your system running well.

IMPORTANT ! Cloudian's Smart Support feature is not a comprehensive remote monitoring program. Cloudian Support will notify you only in the case of critical issues, and not in real time. Primary responsibility for monitoring and managing your HyperStore system lies with you, the customer.

Specifically, the Smart Support mechanism provides Cloudian Support the following system and node information:

- Cluster topology information
- HyperStore license information
- Packages and dependencies version information
- Node hardware, OS, and software information, including (if applicable) information specific to HyperStore appliances
- Network interface configuration and network statistics for each node
- Storage capacity utilization information for each node
- RAM and CPU utilization information for each node
- Disk diagnostics data for each node
- Service status (service UP/DOWN) history for each node
- Service JMX statistics
- S3 transaction performance data for each node
- System performance data such as disk I/O and per-disk capacity usage for each node
- Current map of tokens to disks for each node
- Redis information to confirm sync between master and slave nodes
- Data repair operation status and history for each node
- Proactive repair queue information for the cluster
- HyperStore configuration files for each node
- System crontab configuration for each node
- Alerts for each node(the same as you would see in the CMC's [Alerts](#) page)
- System log files and HyperStore log files
- TLS certificate expiration dates

This information is collected daily on to one of your nodes (the node identified as the "System Monitoring/Cronjob Primary Host" in the CMC's **Cluster Information** page [**Cluster -> Cluster Config -> Cluster Information**]). Under `/var/log/cloudian` on the node there are these files:

- *diagnostics.csv* -- This is the live file into which the current day's performance statistics are continuously written.
- *diagnostics_<date/time>_<version>_<region>.tgz* -- Once a day the current *diagnostics.csv* file is packaged -- together with application and transaction log files -- into a *diagnostics_<date/time>_<version>_<region>.tgz* file. This file is transmitted to Clouidian Support once each day. A local copy of each *diagnostics_<date/time>_<version>_<region>.tgz* file remains on the node for 15 days before being automatically deleted.

The Smart Support feature is **enabled by default**. You have the option of disabling the feature, although this is not recommended. If you have concerns about data privacy, rather than disabling the Smart Support feature consider invoking the option to conceal private data as described in ["Maintaining Personal Data Privacy While Using Smart Support and Node Diagnostics"](#) (page 121).

Note Neither the Smart Support feature nor the On-Demand Node Diagnostics feature (described below) uploads private key information. Also, any passwords in configuration files are replaced by "*****" in the configuration file copies that are uploaded.

Automatic Support Case Creation for Failed Disks

As part of the Smart Support feature, if a data disk on a HyperStore node fails (becomes disabled), information about the failed disk is automatically sent to Clouidian Support within minutes. This triggers the automatic opening of a Support case for the failed disk. This occurs in addition to the alerting functions that bring the failed disk to your attention.

For HyperStore Appliances, automatic case creation is also performed for failed OS disks.

For more information on automated disk failure handling see ["Automated Disk Usage Management Feature Overview"](#) (page 124).

4.2.1.2. On-Demand Node Diagnostics

HyperStore also supports a mechanism that allows you to easily collect deep diagnostic data for a node that you are having a problem with. You can use the CMC's [Collect Diagnostics](#) function to trigger the collection and packaging of this Node Diagnostics data and send the package to Clouidian Support so that they can help you troubleshoot the problem. So while the fully automated Smart Support feature allows for proactive monitoring and support of your system as a whole, the on-demand Node Diagnostics feature allows for deep-dive reactive troubleshooting for problems that have occurred with specific nodes.

The Node Diagnostics package includes:

- System log files and HyperStore log files for the target node(s)
- Outputs from various system commands and HyperStore application commands for the target node(s)
- MBean data for the Java-based HyperStore services for the target node(s)
- Configuration files for the target node(s)
- Salt configuration files and logs from the system

On the target node(s), the Node Diagnostics mechanism packages all this data into */var/log/cloudian/cloudian_sysinfo/<hostname>_<YYYYMMDDhhmm>.tar.gz*. The system automatically sends this package to Clouidian Support, if you chose that option when you used the CMC to generate the package. The local copy is retained for 15 days before being automatically deleted.

Automatic Upload of Node Diagnostics for Support Cases

If you want, you can have the system automatically upload to Cloudian Support the node diagnostics logs that Cloudian Support needs whenever you open a support case. If you enable this feature, you will not need to collect and upload logs to Cloudian Support yourself when you open support cases. By default configuration this feature is disabled. To enable it set [jobs.system.onDemandLogCollector.enabled](#) to true.

4.2.1.3. Data Privacy

If you want to take advantage of the Smart Support and on-demand Node Diagnostics features but you have concerns about data privacy, there is a configurable option to conceal private data in the log copies that are uploaded to Cloudian Support. For more information see ["Maintaining Personal Data Privacy While Using Smart Support and Node Diagnostics"](#) (page 121).

See also:

- ["Configuring Smart Support and Node Diagnostics"](#) (page 120)
- ["Executing Node Diagnostics Collection"](#) (page 123)
- ["Cloudian Support Tunnel"](#) (page 123)

4.2.2. Configuring Smart Support and Node Diagnostics

The HyperStore Smart Support and Node Diagnostics features work out of the box and do not require any configuration changes. Optionally you can do the following with configuration settings:

- Have the daily Smart Support upload go to an S3 destination other than Cloudian Support, by customizing these settings:
 - ["common.phoneHome.url"](#) (page 749)
 - ["common.phoneHome.bucket"](#) (page 744),
 - ["common.phoneHome.auth.accessKey"](#) (page 744)
 - ["common.phoneHome.auth.secretKey"](#) (page 744)
- Have the daily Smart Support upload use a local forward proxy by customizing these settings:
 - ["common.phoneHome.proxy.host"](#) (page 748)
 - ["common.phoneHome.proxy.port"](#) (page 748)
 - ["common.phoneHome.proxy.auth.username"](#) (page 747)
 - ["common.phoneHome.proxy.auth.password"](#) (page 747)
- Disable the daily Smart Support upload by setting ["common.phoneHome.enabled"](#) (page 745) to false. **This is not recommended.** If you have concerns about data privacy, rather than disabling the Smart Support feature consider invoking the option to conceal private data as described in ["Maintaining Personal Data Privacy While Using Smart Support and Node Diagnostics"](#) (page 121).
- Have on-demand Node Diagnostics uploads (triggered by your using the CMC's Collect Diagnostics function [**Cluster -> Nodes -> Advanced**]) go to an S3 destination other than Cloudian Support, by customizing these settings:
 - ["common.sysInfo.url"](#) (page 753)
 - ["common.sysInfo.bucket"](#) (page 751)

- ["common.sysInfo.auth.accessKey"](#) (page 751)
- ["common.sysInfo.auth.secretKey"](#) (page 751)
- Have the system automatically upload to Cloudian Support the node diagnostics logs that Cloudian Support needs to process a case that you've opened. If you enable this feature, you will not need to collect and upload logs to Cloudian Support yourself when you open support cases. By default configuration this feature is disabled. To enable it set [jobs.system.onDemandLogCollector.enabled](#) to true.

Note HyperStore also supports a set of *hyperIQ.phoneHome.** settings. These settings are for **future** use, when the [Cloudian HyperIQ application](#) will support the ability to parse and analyze phone home files. At that point in time you will be able to use the *hyperIQ.phoneHome.** settings to specify a local, on-premise bucket to which to write daily phone home files for HyperIQ consumption **in addition to** having the system upload daily phone home files to Cloudian Support. Currently HyperIQ does not yet support the ability to process phone home files and so in HyperStore 8.2 the *hyperIQ.phoneHome.** settings are empty and do not yet have any utility. When HyperIQ does start supporting this ability, this will be reflected in your HyperIQ documentation and in your HyperStore documentation.

4.2.3. Maintaining Personal Data Privacy While Using Smart Support and Node Diagnostics

For HyperStore operators, [Smart Support](#) is an extremely valuable feature that enables Cloudian Support to proactively detect issues of concern in your HyperStore system, bring the issues to your attention, and work with you to promptly resolve them. Likewise, the [on-demand Node Diagnostics](#) feature helps Cloudian Support serve you better when troubleshooting a known problem with a particular HyperStore node. Both features entail the upload of system data to Cloudian Support through a secure S3 interface.

For some HyperStore customers, the need to maintain data privacy or data sovereignty might seem to preclude taking advantage of these valuable Support features. However, HyperStore provides a configurable option to conceal private data in the logs that are uploaded to Cloudian Support through Smart Support and on-demand Node Diagnostics. This option makes the benefits of these Support features available to organizations with stringent data privacy requirements, such as:

- Government or military organizations
- Organizations in sensitive industry sectors such as financial services
- Organizations subject to data privacy regulations such as the European Union's General Data Protection Regulation (GDPR)

By using the "conceal private data" option, such organizations can get the best in proactive Support (leveraging daily Smart Support uploads) and reactive Support (leveraging on-demand Node Diagnostics uploads) without loss of data privacy or sovereignty.

How the "Conceal Private Data" Option Works

If your organization has stringent data privacy requirements, you can use the "conceal private data" option to have your HyperStore system encrypt **personally identifiable user information** in the log copies that get uploaded to Cloudian Support as part of the Smart Support and Node Diagnostics features. Specifically, the conceal private data option encrypts:

- User IDs
- Email addresses

- IAM role session information
- S3 bucket names and object names
- S3 client IP addresses and hostnames

Note The conceal private data option does not encrypt HyperStore cluster IP addresses and hostnames. It only encrypts S3 client IP addresses and hostnames.

The encryption process uses an encryption key that is unique to your HyperStore system and is generated automatically when you do a fresh installation of -- or upgrade to -- HyperStore version 7.4 or newer. Cloudian does not have access to the encryption key, so Cloudian cannot decrypt the encrypted log field values. Only you can decrypt an encrypted log field value, using a tool that comes with your HyperStore system. Your ability to decrypt an encrypted log field value will come into play if Cloudian Support is analyzing logs while working with you to troubleshoot an issue, and Cloudian Support asks you to provide the decrypted value of a particular field from a relevant log entry.

Note This encryption feature encrypts sensitive log field values only in the log file **copies** that are uploaded to Cloudian Support. It does not alter in any way the original log files.

Enabling the "Conceal Private Data" Option

The "conceal private data" option for Smart Support and Node Diagnostics is **disabled by default**. To enable this option do the following:

1. To conceal user IDs, email addresses, role session information, and S3 client IPs and hostnames, run this *hsctl* command on the Config Controller node:

```
# hsctl config set common.phoneHome.concealPrivateData.enabled=true
```

2. Optionally, if you want to also conceal bucket names and object names, run this command:

```
# hsctl config set common.phoneHome.concealPrivateData.bucketObjectNames=true
```

3. Apply your change(s):

```
# hsctl config apply common
```

No service restart is necessary.

Decrypting an Encrypted Log Field Value

If you enable the feature that encrypts sensitive fields in log copies uploaded to Cloudian Support (as described above), and the Cloudian Support team is analyzing log copies while working with you to troubleshoot a system issue, the Cloudian Support team may ask you to decrypt an encrypted value from a relevant log entry.

Here is an excerpt from a log file entry with an encrypted value. An encrypted field appears as "Enc:<value>". In the excerpt below the encrypted value is shown in bold:

```
2023-09-03 00:00:05,503|Enc:<3Wi5Nacd9lMJgNQJ:kk2krSUIcNWiHRIY>|...
```

In this example, if Cloudian Support needed you to decrypt the value to aid in resolving a support case, Cloudian Support would provide you the encrypted string `3Wi5Nacd9lMJgNQJ:kk2krSUIcNWiHRIY` through the Support case.

You would then log into the Config Controller node and run this command to decrypt the encrypted string:

```
# /opt/cloudian/tools/decrypt_private_data.py encrypted-string
```

In this example it would be:

```
# /opt/cloudian/tools/decrypt_private_data.py 3Wi5Nacd9lMJgNQJ:kk2krSUIcNWiHRLY
10.50.40.125
```

If you are using the **HyperStore Shell**

If you are using the [HyperStore Shell \(HSH\)](#), from any directory on the Config Controller node you can run the decryption tool without specifying a path to the tool:

```
$ decrypt_private_data.py 3Wi5Nacd9lMJgNQJ:kk2krSUIcNWiHRLY
10.50.40.125
```

You would then provide Cloudian Support with the decrypted string value -- 10.50.40.125 in this example.

Note The `decrypt_private_data.py` tool uses your system's unique encryption key to decrypt the string.

4.2.4. Executing Node Diagnostics Collection

To generate deep-dive Node Diagnostics in support of troubleshooting one or more problem nodes, use the CMC's ["Collect Diagnostics"](#) (page 487) function. When using that function you will be able to select one or more nodes for which to collect diagnostics. You will also be able to choose whether to have the diagnostics package(s) that you generate get uploaded to Cloudian Support (or an alternative S3 destination, if you have [configured the system to support this option](#)).

A node's diagnostics package will be created on the node at path `/var/log/cloudian/cloudian_sys-info/<hostname>_<YYYYMMDDhhmm>.tar.gz`.

4.2.4.1. Automatic Upload of Node Diagnostics for Support Cases

If you want, you can have the system automatically upload to Cloudian Support the node diagnostics logs that Cloudian Support needs to process a case that you've opened. If you enable this feature, you will not need to collect and upload logs to Cloudian Support yourself when you open support cases. By default configuration this feature is disabled. To enable it set [jobs.system.onDemandLogCollector.enabled](#) to true.

4.2.5. Cloudian Support Tunnel

In some circumstances Cloudian Support may want to work with you through a "support tunnel" to help you resolve an issue with your HyperStore system. The support tunnel is implemented by a service that is bundled with the HyperStore product, but it is **disabled by default**.

At Cloudian Support's direction, you may be asked to use some of these support tunnel related commands on a HyperStore node:

Action	Command
Enable the support tunnel service	<code>systemctl enable cloudian-support-tunnel</code>
Start the support tunnel service	<code>systemctl start cloudian-support-tunnel</code>
Check the support tunnel service status	<code>systemctl status cloudian-support-tunnel</code> (this status check is one method of obtain-

Action	Command
	ing a support "token" that Cloudian Support may transmit to you in some circumstances)
Stop the support tunnel service	<code>systemctl stop cloudian-support-tunnel</code>
Disable the support tunnel service	<code>systemctl disable cloudian-support-tunnel</code>

Also, in some circumstances Cloudian Support may direct you to edit this support tunnel configuration file on a HyperStore node:

```
/etc/cloudian/support-tunnel/support-tunnel.yaml
```

Note Do not edit this file unless directed to do so by Cloudian Support.

If you are using the [HyperStore Shell](#) as a trusted user you can perform any of the above tasks, at the direction of Cloudian Support (you can run the `systemctl` commands and you can use [hspkg config commands](#) to view or edit the `support-tunnel.yaml` file).

4.3. Automated Disk Usage Management

4.3.1. Automated Disk Usage Management Feature Overview

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Automatic Correction of Substantial Disk Usage Imbalance" \(page 124\)](#)
- ["Automatic Stop of Writes to a Disk at 90% Usage" \(page 125\)](#)
- ["Automatic Stop of Writes to a Node at 90% Usage" \(page 125\)](#)
- ["Dynamic Object Routing" \(page 127\)](#)

HyperStore has mechanisms for automatically correcting imbalances of data disk utilization on each node, and automatically discontinuing new writes to disks or nodes that are near capacity. Some aspects of these automated mechanisms are configurable.

4.3.1.1. Automatic Correction of Substantial Disk Usage Imbalance

HyperStore is implemented in such a way that among multiple data storage disks on the same host the disk usage will typically be well balanced. However, the system also supports an automated mechanism for detecting and rectifying disk usage imbalance if it does occur. On a configurable interval (by default 72 hours) the system checks for a configurable degree of disk usage imbalance on each node (by default a 10% delta between a given disk's utilization and the average disk utilization on the node). If imbalance is detected whereby a particular disk's usage exceeds the node's average disk utilization by more than the configured delta, the system migrates one or more storage tokens from the over-used disk to less-used disks on the same node. From that point forward the less-used disks will store an increased share of newly uploaded object data while the over-used disk stores a reduced share of newly uploaded object data. Note that this mechanism **does not move**

existing data from one disk to another; rather, it impacts the share of new data load allocated to each disk going forward.

The same imbalance-correcting logic applies if the system detects that a particular disk's usage is lower than the node's average disk utilization by more than the configured delta. The system automatically migrates one or more storage tokens from the node's more heavily utilized disks to the under-utilized disk.

You do not need to run a repair operation in connection with this disk usage balancing feature. The disk usage rebalancing mechanism is not intended to move existing data between disks. It is designed to impact only objects uploaded after the time that the automated token migration was executed. If you were to perform a repair, existing objects on the over-used disk will not follow a migrated token to the token's new home on a less-used disk. For more about how the token migration works see ["Dynamic Object Routing" \(page 127\)](#).

Note that the disk usage balancing feature applies only to **HyperStore data disks** (where S3 object data is stored). It does not apply to disks that store only Cassandra, Redis, or the OS.

For more information on configuring this feature see ["Configuring Disk Usage Balancing" \(page 127\)](#).

4.3.1.2. Automatic Stop of Writes to a Disk at 90% Usage

Every 30 minutes the capacity usage level of each HyperStore data disk in the system is checked. If 90% or more of a disk's capacity has been filled the system automatically moves all of the disk's tokens to other disks that are at less than 90% usage, on the same node. The tokens moved away from the 90% full disk are allocated among the other disks on the node in an approximately even manner, but with optimizations in the event that some of the other disks have significantly more available free space than others.

The object data on the 90% full disk remains readable and the disk will still support S3 get requests and delete requests, but any S3 writes associated with the token ranges that used to be on the disk are now directed to the new locations of those token ranges, on other disks on the same node. This disk-level **"stop-write"** condition triggers a warning message in the HyperStore application log, which in turn results in an alert being generated in the CMC's [Alerts](#) page.

For a disk that reaches stop-write condition, if its capacity usage level later falls below 85% it exits stop-write condition and the tokens that had been moved away from it are moved back to it.

Note By capping the portion of a disk's capacity that goes to object data storage at 90%, HyperStore preserves 10% of each disk's capacity for system needs such as those associated with repair or rebalance operations.

4.3.1.3. Automatic Stop of Writes to a Node at 90% Usage

If **all** of a node's data disks have reached a stop-write condition -- because each disk has hit 90% disk usage -- then the **whole node goes into a "stop-write" condition**. When a node is in "stop-write" condition:

- **The S3 Service is not able to write object data to the token ranges assigned to that node. This may result in the failure of some S3 PUT requests from client applications.** Whether failures occur or not depends on the consistency requirements that you have configured for your [storage policies](#), and on the availability of the other nodes in your system. For example if you are using 3X replication with a write consistency level of ALL, then an S3 PUT of an object will fail if one of the three endpoint token ranges for the object (as determined automatically by a hash of the bucket name and object name) is a token range on the node that's in stop-write condition. With 3X replication and a write consistency level of QUORUM, then an S3 PUT of an object will fail if one of the object's three endpoint token ranges is

on the stop-write node and either of the nodes hosting the object's other two endpoint token ranges is also unavailable (such as if one of those other nodes is down, or is in stop-write condition). Note that HyperStore does **not** reallocate tokens from a node that's in stop-write condition to other nodes in the system. [Dynamic token reallocation](#) is supported only between the disks on a node -- not between nodes.

Note After detecting that a node is in stop-write condition, the S3 Service will mark that node as unavailable for object data writes and will stop sending object data write requests to that node (rather than continuing to send object data write requests to that node and having all those requests fail).

- The S3 Service is still able -- while implementing requests from S3 client applications -- to read object data from that node and to delete object data from that node.
- Object metadata can still be written to the Cassandra database on that node, since this is on the OS and metadata disk(s) not the data disks. The stop-write feature applies only to the data disks.
- The [hsstool repair](#), [hsstool repairec](#), and *hsstool decommission* operations cannot write to a node that's in stop-write condition. If a node is in stop-write condition, then writes that *hsstool* operations direct to that node will fail (and in the operation's status metrics these failures will increment the "Failed count").
- When a node goes into stop-write condition a critical message appears in the CMC's [Dashboard](#) page; an alert is generated in the [Alerts](#) page; and the node is marked by a red disk-stack icon in the [Data Centers](#) page.

IMPORTANT ! To avoid having disks and nodes go into a stop-write condition, closely monitor your system's current and projected disk space usage and **expand your cluster well in advance of disks and nodes becoming nearly full**. See ["Capacity Monitoring and Expansion"](#) (page 656).

Getting a Node Out of Stop-Write Condition

A node will exit the "stop-write" condition -- and the S3 layer will resume sending object data write requests to the node -- if enough data is deleted from the node's data disks to reduce the node's **average disk utilization to 85% usage or lower**. At this point all of the node's tokens -- and consequently all of the S3 object data writes on the node -- will be allocated to the data disks that are currently at 85% usage or lower. Disks that are still above 85% utilization will not be assigned any tokens and will not support writes.

When you have a node in stop-write condition there are two ways in which disk space utilization on the node can be reduced such that the node starts accepting writes again:

- You can **delete objects from your HyperStore system**. In HyperStore each object's replicas or erasure coded fragments are stored on multiple nodes and there is not a way to target only the replicas and fragments on a specific node for deletion. Rather, you can delete objects from the system as a whole so that the overall disk space utilization in the system is reduced. This will have the effect of also reducing disk space utilization on the node that's in stop-write condition. You can delete objects either through the S3 interface or through the special Admin API call that lets you efficiently delete all the objects in a specified bucket (in the *Cloudian HyperStore Admin API Reference*, see the "bucketops" section). You can use the CMC's [Node Status](#) page to periodically check the node's disk space usage.

Note When you delete objects using the S3 interface or Admin API, the objects are immediately flagged for deletion but the data is not actually removed from disk until the running of the hourly batch delete cron job.

- You can **add nodes to your cluster**, and execute the associated rebalancing and cleanup operations (for instructions see ["Adding Nodes"](#) (page 579)). This will have the effect of reducing data utilization on your existing nodes, including the node that's in the stop-write condition. Note that rebalance and cleanup are long-running operations and so this approach to getting a node out of stop-write condition may take several days or more depending on how much data is in your system.

4.3.1.4. Dynamic Object Routing

In implementing its disk usage balancing and disk stop-write features the system uses a dynamic token-to-disk mapping scheme that allows tokens to be migrated from one disk to another disk on the same host without existing data following the migrated tokens. With Dynamic Object Routing, an object replica (or EC fragment) associated with a given token range **stays with the disk where that token range was located when the object was originally uploaded**. The system keeps track of the location of each token -- the host and mount-point to which each token is assigned -- across time. Then, based on an object's creation timestamp, the system can tell where the object's replicas (or EC fragments) are located based on where the relevant token ranges were located at the time that the object was first uploaded. In this way object data can be kept in place - and read or updated at its original location -- even if tokens are moved.

Dynamic Object Routing allows for low-impact automated token migrations in circumstances such as disk usage imbalance remediation, disk stop-write implementation, and [automated disk failure handling](#).

Note This automated token migration occurs **only between disks on a node -- not between nodes**.

4.3.2. Configuring Disk Usage Balancing

With the HyperStore data disk usage balancing feature there are the questions of how often to check the disk usage balance on each host and what degree of imbalance should trigger a token migration.

By default, each node is checked for disk imbalance every 72 hours, and token migration is triggered if a disk's utilization percentage differs from the average disk utilization percentage on the node by more than 10%. For example, if the average disk space utilization on a node is 35%, and the disk space utilization for Disk4 is 55%, then one or more tokens will be migrated away from Disk4 to other disks on the node (since the actual delta of 20% exceeds the maximum allowed delta of 10%). For another example, if the average disk utilization on a node is 40%, and the disk utilization for Disk7 is 25%, then one or more tokens will be migrated to Disk7 from the other disks on the node.

The settings for adjusting the frequency of the disk balance check is:

- **"hyperstore.disk.check.interval"** (page 767) (default = 72 hours)

Note In connection with the HyperStore ["stop-write"](#) feature, if you want to customize the disk usage check interval (default 30 minutes) or the "stop-write" threshold (default 90%) or the "start-write"

threshold (default 85%), consult with Clouddian Support. These settings are not in HyperStore configuration files by default.

4.3.3. Triggering a Disk Usage Balance Check

If you don't want to wait until the next automatic check of disk usage balance (which by default occurs every 72 hours for each node), you have the option of using a JMX command to immediately trigger a disk balance check on a specific node.

To trigger an immediate disk usage balance check:

1. Use *JConsole* to access the HyperStore Service's JMX port (19082 by default) on the host for which you want the balance check to be performed.
2. Access the *com.gemini.cloudian.hybrid.server.disk* → *VirtualNodePartitioner* MBean, and under "Operations" execute the "shuffletoken" operation.

The operation will run in a background thread and may take some time to complete. If the space utilization for any disk is found to differ from the node's average disk utilization by more than 10%, then tokens will automatically be migrated between disks.

4.3.4. Checking Disk Usage and Health Status

You can use the CMC to check disk status for each disk on each node in your HyperStore system. Go to the CMC's [Node Status](#) page, select a node, and then review the "Disk Detail Info" section of the page. Here you will see

- The current space utilization information for each disk on the selected node.
- The current health status of each disk on the selected node. Each data disk's status is communicated by a color-coded icon, with the status being one of OK, Error, or Disabled.

4.4. Automated Disk Failure Management

4.4.1. Automated Disk Failure Management Feature Overview

In the event of failure of a HyperStore data disk — a disk where S3 object data is stored — the HyperStore system by default automatically detects the failure and takes the disk offline. To detect disk failures, for each node the HyperStore system does the following:

- Continuously monitors the HyperStore Service application log for error messages indicating a failure to read from or write to a disk (messages containing the string "HSDISKERROR").
- Once each hour, tries to write one byte of data to each HyperStore data disk. If any of these writes fail, */var/log/messages* is scanned for messages indicating that the file system associated with the disk drive in question is in a read-only condition (message containing the string "Remounting filesystem read-only"). This recurring audit of disk drive health is designed to proactively detect disk problems even during periods when there is no HyperStore Service read/write activity on a disk.

If HyperStore Service application log errors regarding a drive occur in excess of an error rate threshold (100 errors in the space of 30 minutes), or if the proactive audit detects that a drive is in read-only condition, then HyperStore by default **automatically disables the drive**.

When a drive is automatically disabled, the system will no longer direct writes or reads to that drive. The storage tokens from the disabled disk are automatically moved to the other disks on the host, so that new writes associated with those token ranges can be directed to the other disks. The disabled disk's data is not recreated on the other disks, and so that data is unreadable on the host. (In typical circumstances the data should be readable from its replicas -- or erasure coded fragments -- on other hosts in the system.)

In the case of a multi-path (MPIO) device, the device is automatically disabled only if all paths to the device have failed.

Note

* You can tell that a disk is disabled by viewing its status in the ["Disk Detail Info"](#) section of the CMC's **Node Status** page.

* If you have the [Smart Support](#) feature enabled (as it is by default), then when a data disk on one of your HyperStore nodes becomes disabled information about the failed disk is automatically sent to Cloudian Support within minutes. This triggers the automatic opening of a Support case for the failed disk. For HyperStore Appliances, automatic case creation is also performed for failed OS disks.

* Failures of DELETE operations do not count toward determining whether a disk needs to be automatically disabled. Instead the determination is based on read and write log errors and the proactive write mechanism as described above.

4.4.1.1. Limitations on Automatic Disk Failure Handling

The automatic disk disabling feature works only if you have multiple HyperStore data disks on the host. If there is only one HyperStore data disk on the host, the system will not automatically disable the disk even if errors are detected.

Also, the automatic disk disabling feature works only if you are using *ext4* file systems mounted on raw disks (which is the only officially supported configuration). If you've installed HyperStore on nodes with unsupported technologies such as Logical Volume Manager (LVM) or XFS file systems, the automatic disk disabling feature will be deactivated by default and the HyperStore system will not take any automatic action in regard to disk failure. Also, the automatic disk disabling feature does not work in Xen or Amazon EC2 environments. Contact Cloudian Support if you are using any of these technologies.

4.4.2. Disk Error Alerts

HyperStore sends a "Disk Error" alert email to the system administrator(s) if a disk read/write error occurs in the HyperStore Service application log. An alert also appears in the CMC, in both the [Alerts](#) page and the [Node Status](#) page.

Note that such an alert does not necessarily indicate that the disk has been automatically disabled. This is because the alert is triggered by the appearance of a single "HSDISKERROR" message in the HyperStore Service application log, whereas the automatic disabling action is triggered only if such messages appear at a rate exceeding an internally predefined threshold (100 errors in the space of 30 minutes).

4.4.3. Responding to a Disabled Disk

If a disk has been automatically disabled by HyperStore in response to disk failure, you have two options for restoring service on that drive:

- **Re-enable the same disk.** You might choose this option if, for example, you know that some cause other than a faulty disk resulted in the drive errors and the automatic disabling of the disk.

HyperStore provides a highly automated method for bringing the same disk back online. For instructions see ["Enabling a HyperStore Data Disk" \(page 640\)](#).

- **Replace the disk.** You would choose this option if you have reason to believe that the disk is bad.

HyperStore provides a highly automated method for replacing a disk and restoring data to the new disk. For instructions see ["Replacing a HyperStore Data Disk" \(page 642\)](#).

With either of these methods, **any tokens that were migrated away from the disabled disk will be automatically migrated back to it (or its replacement)**. Object data that was written in association with the affected tokens while the disk was disabled — while the tokens were temporarily re-assigned to other disks on the host — will remain on those other disks and will be readable from those disks (utilizing HyperStore ["Dynamic Object Routing" \(page 127\)](#)).

Note As part of the [Smart Support](#) feature, if a data disk on one of your HyperStore nodes fails (becomes disabled), information about the failed disk is automatically sent to Cloudbian Support within minutes. This triggers the automatic opening of a Support case for the failed disk. For HyperStore Appliances, automatic case creation is also performed for failed OS disks.

4.4.4. Checking Disk Usage and Health Status

You can use the CMC to check disk status for each disk on each node in your HyperStore system. Go to the CMC's [Node Status](#) page, select a node, and then review the "Disk Detail Info" section of the page. Here you will see

- The current space utilization information for each disk on the selected node.
- The current health status of each disk on the selected node. Each data disk's status is communicated by a color-coded icon, with the status being one of OK, Error, or Disabled.

4.5. Automated Data Repair

4.5.1. Automated Data Repair Feature Overview

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Repair-On-Read" \(page 131\)](#)
- ["Proactive Repair" \(page 131\)](#)
- ["Scheduled Auto-Repair" \(page 132\)](#)
- ["Operator-Initiated Data Repair" \(page 132\)](#)

Through its [storage policies](#) feature, HyperStore provides you the option of using eventual consistency for writes of S3 object data and metadata. For example, in the context of a 3X replication storage policy you can configure a policy such that the system returns a success response to an S3 client's PUT Object request so long as two of the three replicas can be written at the time of the request. As a second example, in the context of a 4+2 erasure coding storage policy you can configure a policy to return a success response to a PUT Object request so long as five of the six erasure coded fragments can be written at the time of the request.

Eventual consistency can reduce S3 write request latency and increase S3 write availability while still providing a high degree of data durability assurance. Eventual consistency also means that for a given object, there may be times when not all of the object's intended replicas or EC fragments exist in the system. For example, in a 3X replication context there may be times when only two replicas of an object exist in the system, rather than the intended three replicas.

HyperStore **automatically** implements several mechanisms to detect and replace missing replicas or EC fragments: repair-on-read, proactive repair, and scheduled auto-repair.

4.5.1.1. Repair-On-Read

Whenever a read request is processed for a particular **replicated** object, all replicas of the object are checked and any missing or out-of-date or corrupted replicas are automatically replaced or updated asynchronously. This repair process occurs regardless of whether or not the read request succeeds -- regardless of whether or not there are enough replicas of the object to satisfy the configured read [consistency requirements](#) -- so long as there is at least one valid replica of the object in the system.

Repair-on-read is also performed for reads of **erasure coded** objects, if there are enough fragments to decode the object but one or more of the object's fragments are missing. For example, if 4+2 erasure coding is being used and the system when reading an object finds only 4 fragments for the object, the read will succeed and also the system will asynchronously replace the object's 2 missing fragments.

Typically with erasure coded data, repair-on-read occurs only when the read request succeeds (only when there are enough fragments to decode the object). The exception is in a multi-DC environment if Replicated EC is being used and the read consistency requirement is LOCAL_QUORUM. In this scenario repair-on-read may be asynchronously performed even in the case of a failed read request. This occurs if a read fails because LOCAL-QUORUM could not be achieved (there are an insufficient number of fragments in the local DC), but the local object data can be repaired by streaming in copies of the object's fragments from the other DC (if the needed fragments exist in the other DC). Note that in this scenario only the local object data gets repaired -- no repair-on-read for the object occurs in the other DC.

By contrast, if Replicated EC is being used and the read consistency requirement is ANY_QUORUM, read-on-repair for an object will occur only if the read request succeeds -- only if, in total across the DCs, there are enough unique fragments to read the object but one or more of the object's fragments are missing. In this scenario the repair will occur asynchronously in all DCs that participate in the storage policy.

4.5.1.2. Proactive Repair

When a node has been down or unreachable, then when the node comes back online it is automatically brought up to date by proactive repair. Proactive repair works by reading from the Metadata DB a list of objects for which a write succeeded in the system as a whole but failed on that node. Working from this object list, proactive repair streams in any locally missing replicas by copying them from nodes where they do exist; or, in the case of erasure coded objects, the proactive repair process re-generates the locally missing fragment. This all happens automatically without need for operator action.

Proactive repair covers several circumstances wherein writes may succeed in the system as a whole but fail to be written to a particular endpoint node:

- The endpoint node was momentarily unavailable
- The endpoint node had been unavailable for long enough for the [S3 layer to mark it as temporarily down](#) and stop sending write requests to it (by default this occurs after a few minutes of a high rate of timeouts or other errors returned by the node)

- The endpoint node was in a [stop-write condition](#) and so the S3 layer stopped sending write requests to it (by default this occurs when all data disks on the node are 90% full or more)
- The endpoint node was [marked down for maintenance](#) by an operator and so the S3 layer stopped sending write requests to it

The maximum time for which proactive repair jobs can be queued for a node that is unavailable is 4 hours by default, and is configurable. Also configurable is the regular interval at which each node in the cluster checks whether there are any queued proactive repair jobs for itself, and executes those jobs if there are any (default = 1 hour). For more information see ["Configuring Automatic Data Repair" \(page 133\)](#).

4.5.1.3. Scheduled Auto-Repair

To ensure that your HyperStore data is protected to the degree that you intend, the system automatically runs node repairs on scheduled intervals. By default:

- To repair replicated object data (associated with Replication storage policies), each node is scheduled to have [hsstool repair](#) automatically run on it once every 30 days.
- To repair erasure coded object data (associated with EC storage policies), each node is scheduled to have [hsstool repairec](#) automatically run on it once every 29 days. This operation also repairs any replicated object data associated with EC Hybrid storage policies.
- To repair metadata in Cassandra, each node is scheduled to have [hsstool repaircassandra](#) automatically run on it once every 7 days.

Note that when a repair operation is running on a target node, the scope of repair activity will extend to other nodes as well. In the case of repair of replicated object data, repair of a target node will also make sure that for objects that fall within the target node's primary token range, the objects' replicas also are present on the other nodes where they are supposed to be. That same repair dynamic is true also for repair of replicated Cassandra metadata.

In the case of erasure coded object data, for single data center storage policies and for multi- data center replicated EC storage policies, repair of a target node has the effect of assessing and repairing all erasure coded data on all nodes within the data center where the target node resides. And for multi- data center distributed EC storage policies, repair of a target node has the effect of assessing and repairing all erasure coded data in all of the participating data centers.

Note The intervals for scheduled auto-repair are configurable as described in ["Configuring Automatic Data Repair" \(page 133\)](#).

4.5.1.4. Operator-Initiated Data Repair

In most circumstances HyperStore's automatic data repair mechanisms -- read-on-repair, proactive repair, and scheduled auto-repair -- are sufficient to keep the data in your cluster complete and consistent. However, there are occasions when you will need to manually trigger a repair operation:

- After removing a dead node from your cluster. See ["Removing a Node" \(page 603\)](#).
- When a node is brought back online after being unavailable for longer than the configurable maximum time that proactive repair can handle. See ["Restoring a Node That Has Been Offline" \(page 612\)](#).

For repair command syntax see [hsstool repair](#) and [hsstool repairec](#) and [hsstool repaircassandra](#).

4.5.2. Configuring Automatic Data Repair

By default, HyperStore's automatic data repair functions are configured in a way that's suitable for typical HyperStore deployments. However, there are a few settings that you can modify if you wish.

The [Scheduled Auto-Repair](#) feature is configurable by these settings in the CMC's [Configuration Settings](#) page:

- Replicas Repair Interval (default = every 30 days)
- EC Repair Interval (default = every 29 days)
- Cassandra Full Repair Interval (default = every 7 days)

Note If you wish, you can have some or all of the auto-repairs of replica data and erasure coded data use the "computedigest" option to combat bit rot. This feature is controlled by the "[hyperstore.repair.scheduled.computeDigest.frequency](#)" (page 776) setting. By default "computedigest" is not used in auto-repair runs.

The [Proactive Repair](#) feature is configurable by these settings:

- The setting "[hyperstore.repair.proactive.interval](#)" (page 773) controls how frequently each HyperStore node checks to see whether any proactive repair jobs are queued for itself, and executes those jobs if there are any. HyperStore nodes automatically make this check upon start-up, and then again at this configurable interval (default is 1 hour). This recurring check is necessary because even if a node hasn't been down, there may be need to execute proactive repair — for example, if network issues had made the node temporarily unreachable as other nodes were processing S3 write requests. In the case of a newly added node, this recurring check will also take care of repairing any data that failed to be streamed into the node during the rebalance operation.

Note If for some reason you want to trigger proactive repair on a particular node immediately, you can do so by running the [hsstool proactiverepair](#) command with the "-start" option.

- The setting "[hyperstore.repair.proactive.retryQueue.timeout](#)" (page 774) sets the maximum time for which proactive repair jobs can be queued for a node that is unavailable (default is 4 hours). This time limit prevents the Metadata DB from being over-loaded with metadata relating to proactive repair, and ensures that proactive repair is used only for its designed purpose, which is to repair object data from a relatively brief time period. If a node is down for longer than this, then when you bring the node back online you will need to let proactive repair complete and then run a manual repair also. For detail see "[Restoring a Node That Has Been Offline](#)" (page 612).

4.5.3. Checking Data Repair Status

Cluster-wide summary information about current data repair activity is available in the CMC's [Repair Status](#) page. This includes information about any in-progress proactive repair, scheduled auto-repair, or operator-initiated repair. This CMC page also indicates whether proactive repair is pending for a node.

Repairs that you have initiated via the CMC can also be tracked in the [Operation Status](#) page. If you initiate a repair via the command line, you can track its progress by executing [hsstool opstatus](#) (either on the command line or via the CMC's [Node Advanced](#) page).

To view the cluster-wide schedule for auto-repairs, execute the [hsstool repairqueue](#) command on any node (either on the command line or via the CMC's **Node Advanced** page).

4.5.3.1. Repair Completion Alerts

Whenever a proactive repair, an auto-repair, or an operator-initiated repair finishes its run, HyperStore sends an alert email to the system administrator(s). An alert also appears in the CMC, in both the [Alerts](#) page and the [Node Status](#) page. The alert indicates the final status of the repair run: either COMPLETED, FAILED, or TERMINATED (if interrupted by an operator).

You can customize these alerts — including an option for having SNMP traps sent — in the CMC's [Alert Rules](#) page.

For detailed repair status information for a recently finished repair run — for example, to get more information about a FAILED repair run that you've been alerted to — in the CMC go to the **Node Advanced** page and from the "Info" command group execute the [hsstool opstatus](#) command for the node on which the repair ran.

4.5.4. Disabling or Stopping Data Repairs

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Temporarily Disabling Automatic Data Repairs" \(page 134\)](#)
- ["Stopping In-Progress Data Repairs" \(page 136\)](#)

HyperStore allows you to temporarily disable its automatic data repair features, and also to stop data repairs that are in progress.

IMPORTANT ! The scheduled auto-repair feature and the proactive repair feature are both important for maintaining data integrity in your system. Do not permanently disable either of these features.

4.5.4.1. Temporarily Disabling Automatic Data Repairs

If you wish you can temporarily disable HyperStore's scheduled auto-repair feature and its proactive repair feature, if for some reason you do not want either type of repair to automatically start up for some period of time.

Note The system **automatically** disables the auto-repair and proactive repair features when you perform a HyperStore version upgrade and when you add nodes to your cluster; and the system automatically re-enables the auto-repair and proactive repair features at the conclusion of those operations. So the steps below are only needed if you want to temporarily disable these features in circumstances other than system upgrade or system expansion.

To disable all automatic data repairs in a service region, go to the CMC's **Node Advanced** page and execute the maintenance command *autorepair* with the Disable option. This will prevent any new schedule-based auto-repairs or proactive repairs from launching. The target node can be any node in the service region; automatic data repairs will be disabled throughout the service region regardless of which node receives the command. Leave the "Type" option unselected.

Cloudian CMC interface showing the 'Advanced' node configuration page. The 'Cluster' tab is selected (1), the 'Nodes' sub-tab is active (2), and the 'Advanced' section is expanded (3). The 'Command Type' is set to 'Maintenance', the 'hstool Command' is 'autorepair', and the 'Target Node' is 'hyperstore-demo1'. Under 'Options', the 'Disable' radio button is selected, and the 'Type' dropdown is set to 'Replicas'. An 'EXECUTE' button is at the bottom right.

Note Disabling automatic data repairs prevents new scheduled auto-repairs and new proactive repairs from launching, but it **does not stop repairs that are currently in-progress**. For information about doing the latter see ["Stopping In-Progress Data Repairs"](#) (page 136).

After completing the system operation that you are undertaking, be sure to **re-enable automatic data repairs**.

Cloudian CMC interface showing the 'Advanced' node configuration page. The 'Cluster' tab is selected (1), the 'Nodes' sub-tab is active (2), and the 'Advanced' section is expanded (3). The 'Command Type' is set to 'Maintenance', the 'hstool Command' is 'autorepair', and the 'Target Node' is 'hyperstore-demo1'. Under 'Options', the 'Enable' radio button is selected, and the 'Type' dropdown is set to 'Replicas'. An 'EXECUTE' button is at the bottom right.

Temporarily Disabling Just Certain Types of Automatic Data Repairs

If you wish you can disable just a certain type of automatic data repair, rather than disabling all automatic data repairs.

To disable just scheduled auto-repair for replicated object data, or just scheduled auto-repair for erasure coded object data, or just scheduled auto-repair for Cassandra metadata, go to the CMC's **Node Advanced** page and execute the maintenance command *autorepair* with the Disable option and with a "Type" specified (*Replicas* or *EC* or *Cassandra*).

To disable just proactive repair go to the CMC's **Node Advanced** page and execute the maintenance command *proactiverepair* with the Disable option.

The screenshot shows the Cloudian CMC interface. The top navigation bar includes 'Cluster' (highlighted with a red box and '1'), 'Alerts', 'Admin', and 'Help'. Below this, the 'Nodes' tab is selected (highlighted with a red box and '2'). The 'Advanced' sub-tab is also selected (highlighted with a red box and '3'). The main form contains the following fields:

- Command Type:** A dropdown menu set to 'Maintenance'.
- hsstool Command:** A dropdown menu set to 'proactiverepair'.
- Target Node:** A dropdown menu set to 'hyperstore-demo1'.
- Options:** Radio buttons for 'Enable', 'Disable' (selected), 'stop', and 'start'. A dropdown menu is set to 'All'.
- Description:** 'Enable/Disable proactive repair of storage data (cluster-wide setting). Start/Stop proactive repair of storage data on target node. For more information about this setting, please see the online Help.'
- EXECUTE** button.

Be sure later to **re-enable whichever type of automatic data repair you disabled**, by using the same interface.

4.5.4.2. Stopping In-Progress Data Repairs

On rare occasions, you may want to stop a data repair that's in progress on a particular node.

- To stop an in-progress repair of replicated object data, use [hsstool repair](#) with the "-stop" option.
- To stop an in-progress repair of erasure coded object data, use [hsstool repairec](#) with the "-stop" option.
- To stop an in-progress repair of Cassandra metadata, use [hsstool repaircassandra](#) with the "-stop" option.
- To stop an in-progress proactive repair, use [hsstool proactiverepairq](#) with the "-stop" option.

It does not matter whether the repair in progress was initiated automatically by the system or initiated by an operator -- in either case you can stop it with the "-stop" option.

Note A typical Cassandra metadata repair would take minutes, and a typical proactive repair would take minutes or hours, but a replicated object data repair or erasure coded object data repair for a node may take days (depending on the amount of data involved and on network bandwidth). If a replica repair or EC data repair is in progress you can check either the CMC's [Operation Status](#) page or the [Repair Status](#) page to see how far along the repair is and approximately how much longer it will take, before deciding whether to stop it. The **Repair Status** page also has progress information for proactive repairs.

4.6. Setting Up Alerts and Notifications

HyperStore has a variety of pre-configured alert rules, mostly geared toward hardware utilization levels (such as disk capacity usage or CPU utilization) and services status (such as a service being down or having errors in its application log). An alert rule defines the condition that triggers a particular alert, and how the system will communicate the alert. You can review the pre-configured alert rules, edit the rules, and create additional rules in the CMC's **Alert Rules** page. For more information see ["Alert Rules" \(page 560\)](#).

By default, alerts are communicated to you only by appearing in the CMC's **Alerts** page (see ["Alerts" \(page 555\)](#)). However, **it is strongly recommended that you supply a system administrator email address** to which HyperStore will send notification emails when alerts are triggered. You can supply the email information (system administrator address and other required information such as your SMTP server's FQDN) in the CMC's **Configuration Settings** page. For more information see ["SMTP/Email Settings for Alerts/Notifications" \(page 498\)](#). Once you've supplied the system with the information required to send emails to a system administrator, the pre-configured alert rules will include sending email to the administrator's address when an alert is triggered.

If you wish you can also have HyperStore send SNMP traps when alerts are triggered. To do so you need to take two steps:

1. Supply the SNMP destination information in the CMC's **Configuration Settings** page. For more information see ["SNMP Trap Destination Settings" \(page 500\)](#).
2. In the CMC's **Alert Rules** page, modify each of the pre-configured alert rules so that they include sending an SNMP trap. (Or modify a subset of the alert rules if you want SNMP traps sent for some alerts and not for others.) The pre-configured alert rules do not include sending an SNMP trap by default.

4.7. Creating Additional System Admin Users

HyperStore comes with one pre-provisioned system administrative user, with user name *admin*. You can log into the CMC as the *admin* user -- as described in [Accessing the Clouidian Management Console](#) -- and perform a variety of system monitoring and administration tasks, including provisioning service users and user groups.

While logged into the CMC as the *admin* user you can use the CMC's ["Manage Users" \(page 400\)](#) page to create additional system admin users if you wish (with different user names). When creating a new user in that page you can select the user type from a drop-down list, and "System Admin" is one of the options.

Once created, a new system admin user will have the same administrative capabilities as the *admin* user: they will be able to log into the CMC and perform system monitoring and administration tasks, including user provisioning (and including the ability to create additional system admin users). System admin users can also delete other system admin users, with the exception that the pre-provisioned system admin user named *admin* cannot be deleted.

System admin users cannot use the CMC to create their own S3 storage buckets or upload/download objects. However:

- System admin users can manage buckets and objects on behalf of regular service users, [if your system is configured to allow this](#).
- System admin users other than the default system admin user (the pre-provisioned user named *admin*) can use third party S3 applications to create their own S3 buckets and upload/download objects.

If you have the [HyperStore Shell](#) enabled, each system admin user will be able to log into and use the HyperStore Shell. For more information see ["Enabling the HSH and Managing HSH Users"](#) (page 106).

Note You can also create and manage system admin users through the Admin API, similar to creating and managing other types of users. For more information see the "user" section of the *Cloudian HyperStore Admin API Reference*.

4.8. Forwarding Log Messages to a Central Log Server

If you wish you can have HyperStore forward application log and request log messages to a central logging server using *rsyslog*. For details see ["logging.syslog.remote.*"](#) (page 803).

Chapter 5. Setting Up Service Features

5.1. hsDNS

5.1.1. hsDNS Feature Overview

hsDNS is a HyperStore component that can act as an authoritative nameserver residing on all HyperStore nodes and providing DNS based service discovery and round-robin load balancing for HyperStore services (S3, Admin, CMC, IAM , and SQS). Built on [CoreDNS](#) technology, hsDNS automatically manages DNS resolving of all service endpoints within your HyperStore cluster, taking into account the up/down status of service instances on each node as well as the up/down status of nodes. hsDNS regularly issues health checks both at the application level and at the node level, and automatically updates DNS records so as to only publish the addresses of currently health service instances.

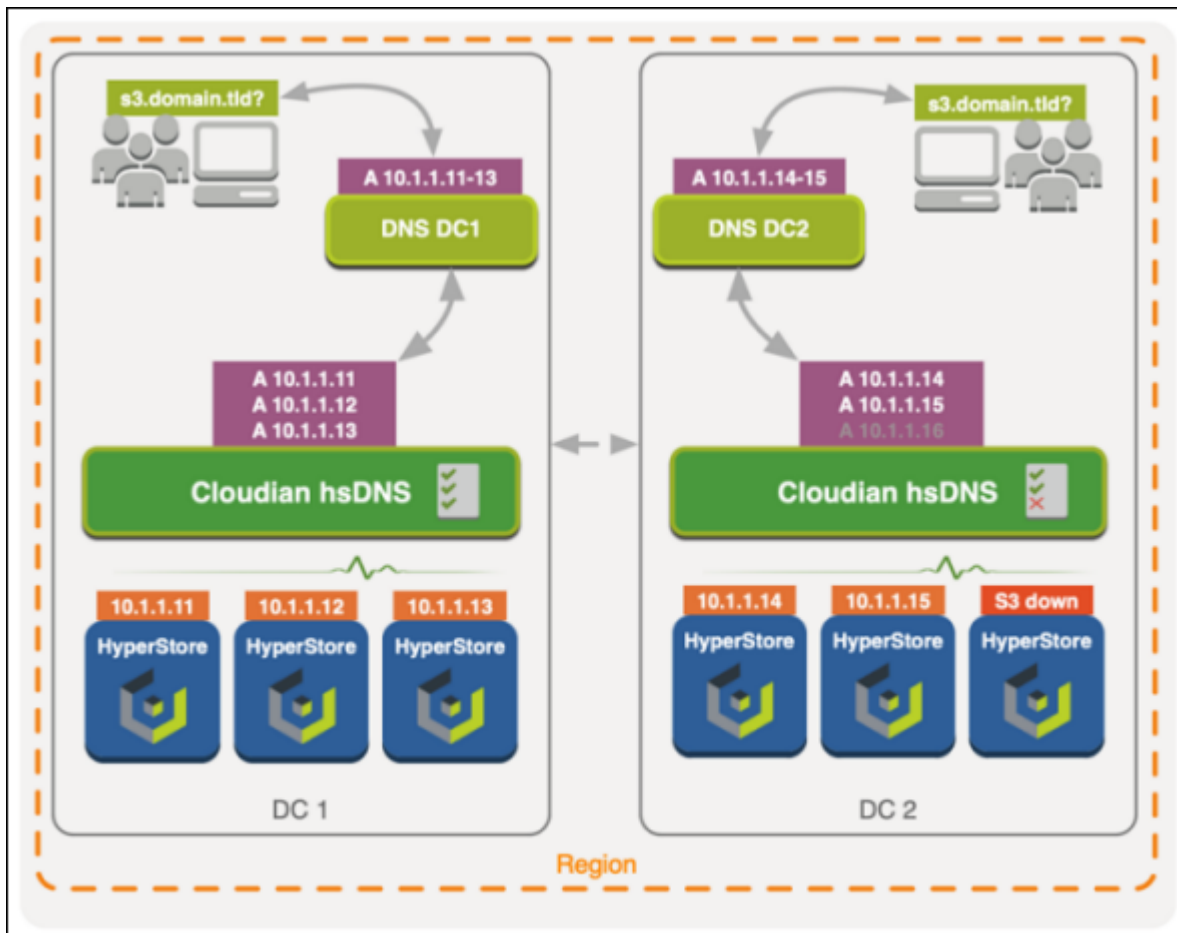
Leveraging HyperStore's resilient, high performance Salt infrastructure, hsDNS is well-suited to provide HyperStore service endpoint resolution and DNS based load balancing in production environments -- as well as in testing, development, and evaluation environments.

By default hsDNS is disabled. If you enable hsDNS, then by default hsDNS provides service endpoint resolution only for **internal network requests** (requests originating from within the HyperStore cluster). However, there is a configuration option to expose hsDNS to frontend network access so that you can hook hsDNS into your existing DNS infrastructure by using DNS delegation. In this way hsDNS can indirectly support end client requests for HyperStore service endpoints, as well as directly supporting internal HyperStore cluster requests.

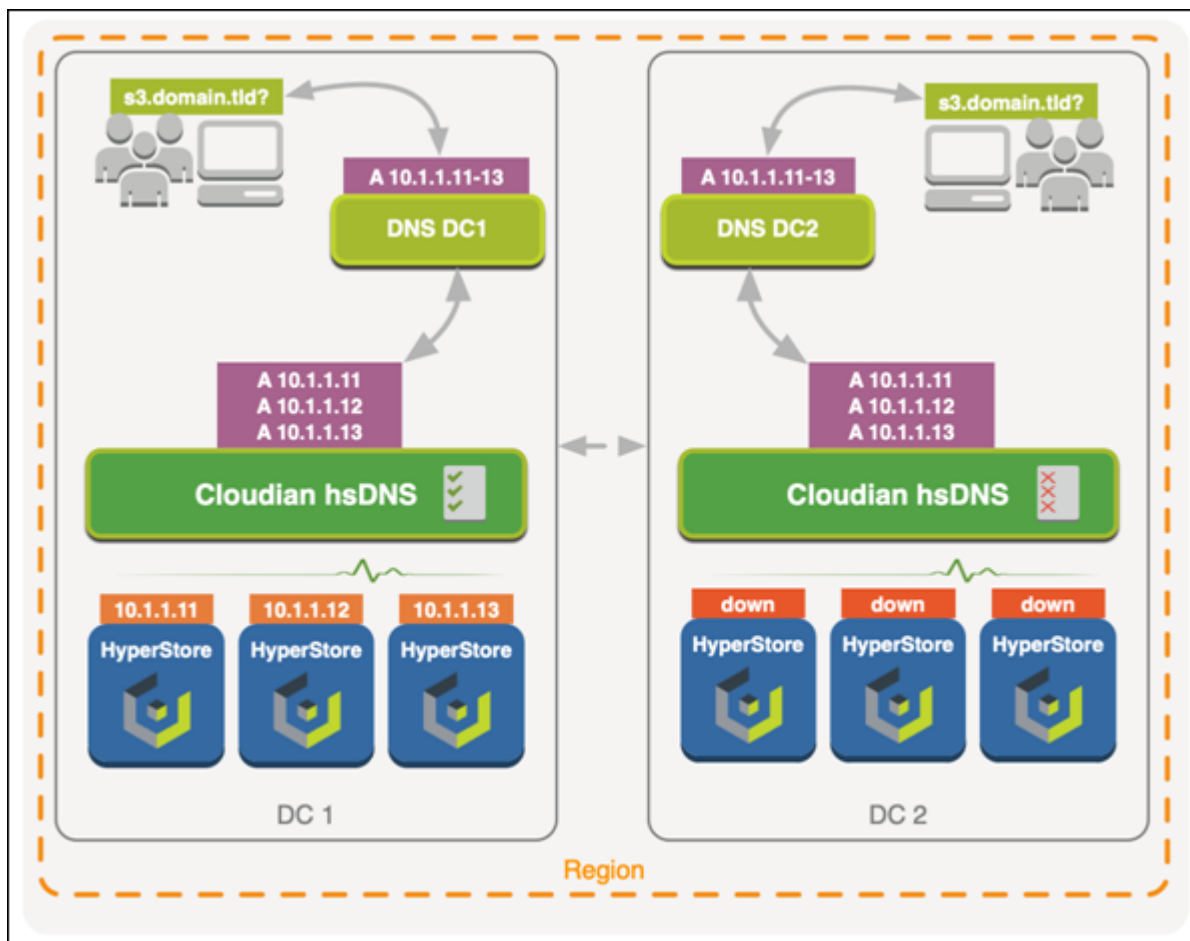
For more information see ["Setting Up hsDNS" \(page 142\)](#).

5.1.1.1. Location Affinity

In the context of a multi-region and/or multi-DC HyperStore deployment, hsDNS by default supports "location affinity", such that service requests are steered to the closest healthy service instances. In this first diagram -- in which DNS delegation is used -- requests incoming to each DC are directed to healthy service instances within that DC. Note that in DC2, one instance of the S3 Service is down, so only the other two instances from that DC are published.



In this second diagram, there are no healthy S3 Service instances in DC2, and so hsDNS in that DC publishes the addresses of the three healthy S3 Service instances in DC1.



As soon as any DC2 instance of the S3 Service is back up it will again be published by hsDNS in that DC, instead of the DC1 instances.

In a HyperStore deployment that has multiple service regions, there are some exceptions to the standard local affinity behavior :

- Because the Admin API Service, IAM Service, and SQS Service run only in the default service region, all requests for those services will resolve to service instances in the default region regardless of the region of request origin.
- S3 requests will resolve to service instances in the region corresponding to the region-specific S3 service endpoint regardless of the region of request origin.

Note You have a configuration option to disable location affinity completely, so that hsDNS publishes all healthy instances of a service regardless of DC of residence.

5.1.1.2. CMC Exclusion from Round Robin Load Balancing

For most HyperStore services hsDNS maintains a pool of IP addresses, corresponding to the currently health instances of each service; and DNS round robin load balancing is employed to balance load across the healthy service instances (in response to queries for a given service, hsDNS returns a different healthy IP address each time, in rotation). The exception is the CMC. To facilitate session stickiness, hsDNS returns just

one CMC instance IP address per DC, while maintaining a pool of healthy CMC instance IP addresses so that failover can be used if that initial CMC instance becomes unavailable.

5.1.2. Setting Up hsDNS

By default [hsDNS](#) is disabled. You can enable it in either of these ways:

- During HyperStore installation, when launching the installer (*cloudianInstall.sh*) you can use the *configure-hsdns* command line option to enable hsDNS from the outset of your HyperStore deployment. For more information about HyperStore installation see your *Cloudian HyperStore Installation Guide* or the "Installing HyperStore" section of the HyperStore Help.
- In an already installed HyperStore system for which you did not enable hsDNS during installation, you can enable hsDNS by using the *hsctl* command line tool on the Config Controller node to set the *lb.hsDNS.enabled* configuration setting to *true*:

```
# hsctl config set lb.hsDNS.enabled=true
# hsctl config set common.configController.failover=auto
# hsctl config apply lb common
```

At any time, the easiest way to check the current enabled/disabled status of hsDNS is by using *hsctl* on the Config Controller node:

```
# hsctl config get lb.hsDNS.enabled
false
```

The *hsctl* tool is also the only means by which to change hsDNS's configuration, if you wish. The table below summarizes hsDNS's main default behaviors (if you've enabled hsDNS) and shows which configuration setting to use if you want to change the behavior.

Default hsDNS Behavior	Configuration Setting
hsDNS only runs on the internal network in which HyperStore resides, providing DNS based service discovery and load balancing for HyperStore intra-cluster communications. The HyperStore firewall does not allow external (front-end) network access to hsDNS. To have hsDNS also resolve requests that ultimately originate from front-end clients -- such as S3 user applications -- requires setting up DNS delegation as noted in the configuration setting description.	"lb.hsDNS.public" (page 801)
In the context of a multi-region and/or multi-DC HyperStore deployment, hsDNS adheres to "location affinity", such that service requests are steered to the closest healthy service instances.	"lb.hsDNS.locationAffinity" (page 800)
In its DNS records hsDNS publishes HyperStore nodes' default/public IP addresses, not back-end IP addresses.	"lb.hsDNS.publishBackend" (page 802)

hsDNS also supports these additional settings if you want to further fine tune its behavior:

- ["lb.hsDNS.healthcheck.probe.type"](#) (page 800)
- ["lb.hsDNS.healthcheck.probe.interval"](#) (page 800)
- ["lb.hsDNS.healthcheck.probe.consult.addrs"](#) (page 799)

- ["lb.hsDNS.healthcheck.probe.consult.policy"](#) (page 799)
- ["lb.hsDNS.ttl"](#) (page 802)

5.2. GPUDirect Storage

5.2.1. GPUDirect Storage Feature Overview

Subjects covered in this section:

- *Introduction (immediately below)*
- ["How HyperStore With GDS Works"](#) (page 143)
- ["Contact Cloudian For More Information"](#) (page 146)

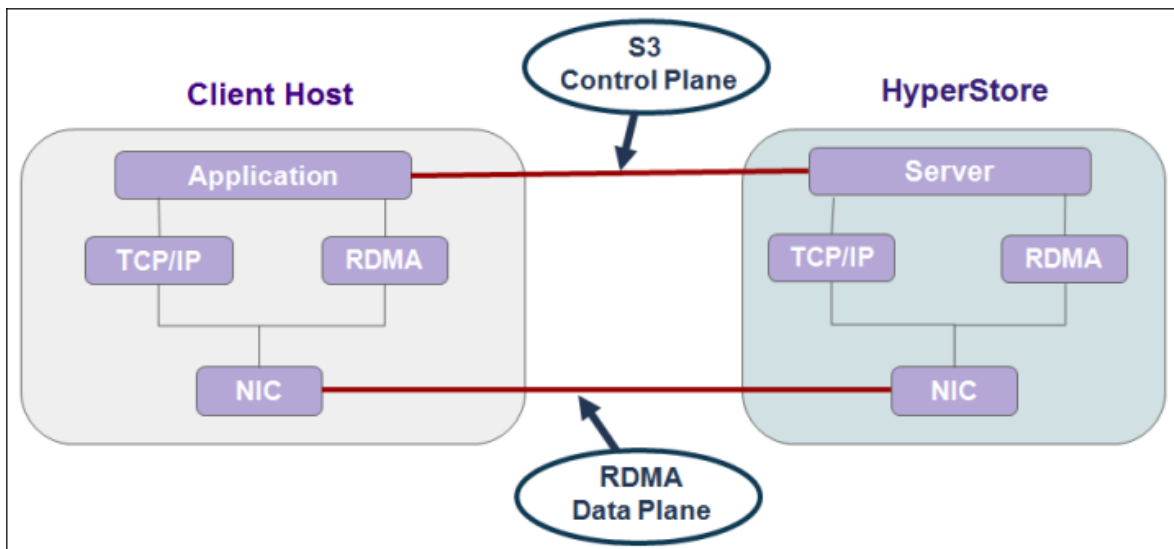
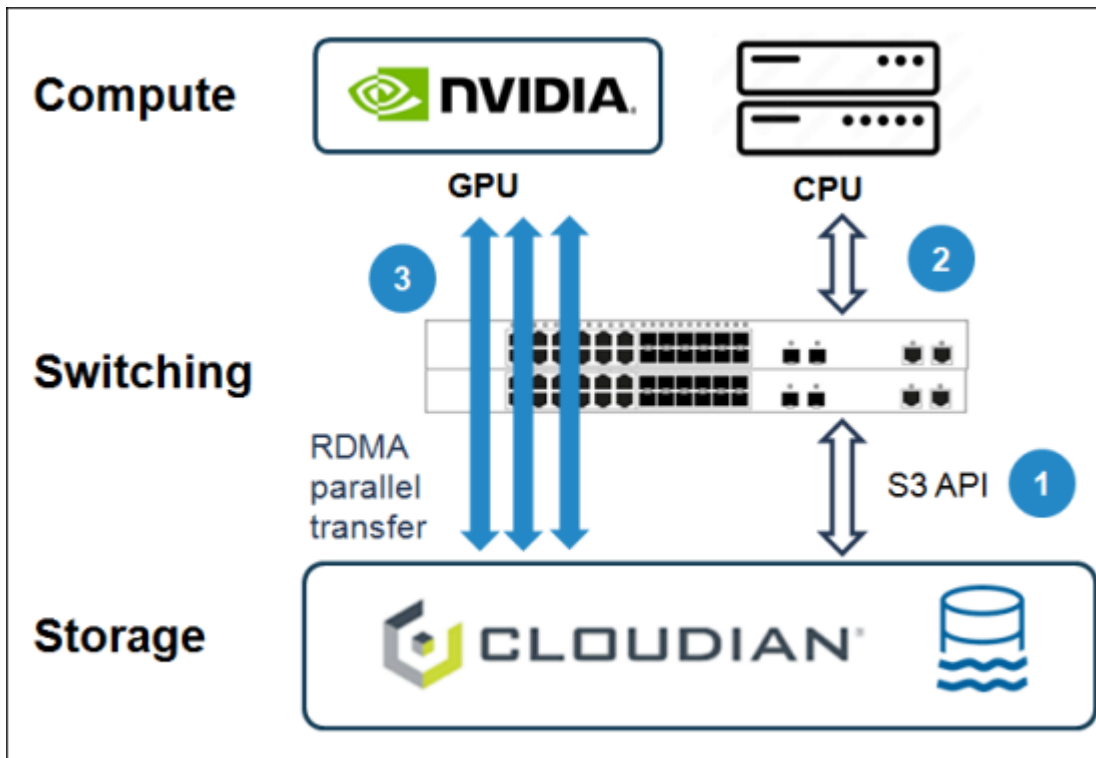
HyperStore supports NVIDIA® GPUDirect® Storage (GDS) to accelerate S3 read and write data transfer between compute nodes and the HyperStore object storage tier. GDS uses Remote Direct Memory Access (RDMA) to facilitate direct object data transfer between HyperStore nodes and GPU memory on the compute nodes, while the S3 protocol serves as the control plane. GDS substantially boosts S3 data throughput while reducing latency, and takes pressure off of CPUs by minimizing the CPUs' role in S3 data transfer operations. This performance profile makes HyperStore with GDS ideally suited to serve as a scalable data lake for AI and other GPU-centric workloads.

In the current HyperStore version (8.2), GDS is supported **only for new HyperStore installs** -- not for HyperStore systems upgraded from previous versions.

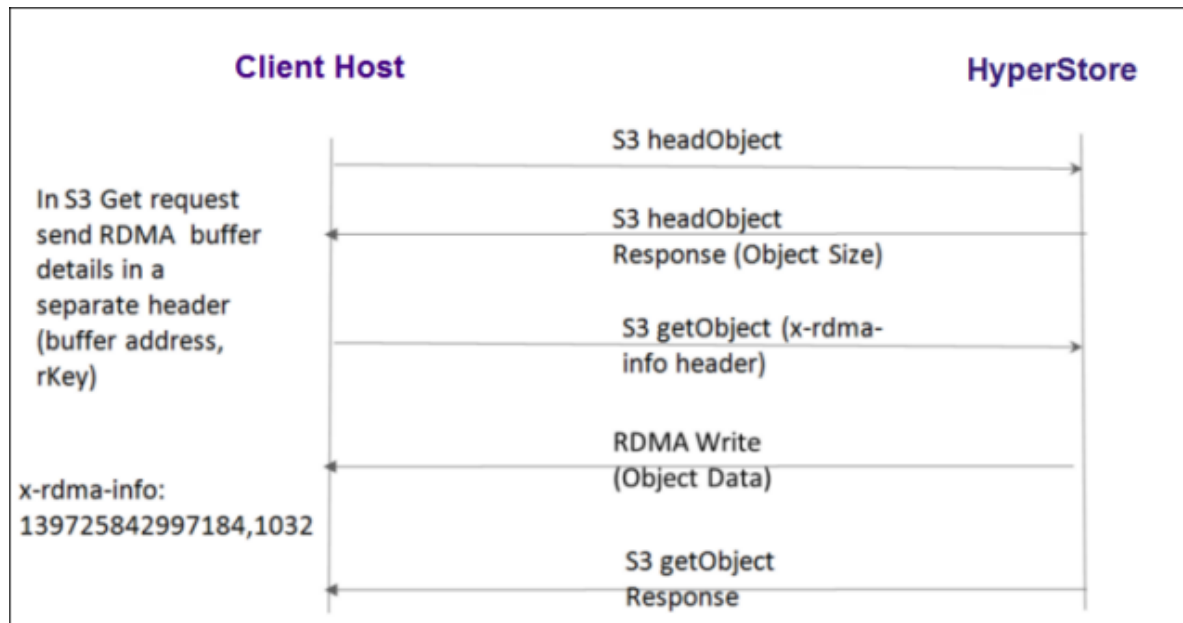
5.2.1.1. How HyperStore With GDS Works

At a high level, the data flow between client nodes and HyperStore nodes with GPUDirect Storage enabled is as follows:

1. **Data request submitted to HyperStore.** A request is initiated via S3 API to move data between HyperStore object storage and GPU memory on the client host. Essentially, S3 calls act as the control plane for the transaction, making use of a custom HyperStore S3 extension header that specifies that the object data transfer is to be implemented by RDMA.
2. **Client tier's CPU usage is limited to S3 control plane activity.** CPU and system RAM are used only for S3 protocol call and response processing, and are bypassed during the actual transfer of object data between the client tier and the HyperStore object storage tier.
3. **Direct GPU memory access.** RDMA is used to transfer data directly between HyperStore object storage nodes and GPU memory on the client nodes, leveraging RDMA-capable NICs on both the HyperStore side and the client side.



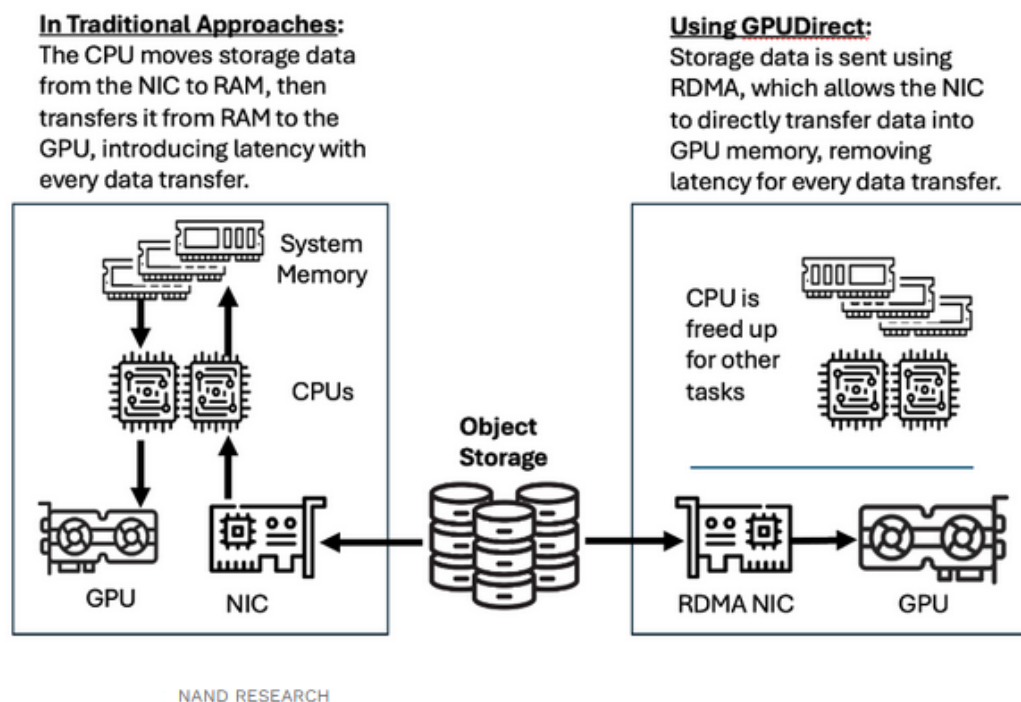
The flow chart below shows the example of an S3 GET Object request with HyperStore GDS enabled. The GET Object request is preceded by a HEAD Object request that obtains the object size, and then the GET Object request includes a special extension header that triggers the use of RDMA for the object data transfer.



HyperStore supports GDS for GET Object, PUT Object, and Multi-Part Upload operations.

Note The request header *x-rdma-info* is a HyperStore-specific extension to the S3 API, which triggers the use of RDMA by the HyperStore S3 Service. Special GDS-supporting S3 client technology is required to generate S3 requests that include this header.

The diagram below contrasts the object data transfer stage of an S3 GET Object transaction as performed without GDS (traditional approach) and with GDS.



5.2.1.2. Contact Clodian For More Information

For information about host and network requirements for using GDS, and for information about setting up GDS as part of a new HyperStore installation, contact your Clodian Sales representative.

5.2.2. GDS Setup

This section provides a high level overview of how to set up a new HyperStore system that uses GPUDirect Storage functionality (HyperStore with GDS).

IMPORTANT ! HyperStore with GDS is supported **only for new HyperStore 8.2 system installations**. HyperStore with GDS is not supported for systems that are upgraded to HyperStore 8.2 from an earlier version.

- ["HyperStore Host System Requirements"](#) (page 146)
- ["Installing HyperStore Software"](#) (page 146)
- ["Enabling GDS Support After Installing HyperStore Software"](#) (page 146)
- ["Obtaining GDS-Capable S3 Client Technology"](#) (page 147)

5.2.2.1. HyperStore Host System Requirements

To use HyperStore with GDS you must use NVIDIA® networking gear as well as GPUs and server hardware that support GPUDirect Storage. For details contact your Clodian sales representative. Your Clodian representative can:

- Work with you to define your **system sizing** needs, based on your HyperStore with GDS use case.
- Provide you with a **reference architecture**.
- Provide you the special **HyperStore GDS ISO file** for installing the Rocky Linux OS on your HyperStore host machines. The OS that you install from this ISO file includes the NVIDIA® Mellanox® ConnectX® drivers that are required in order to use HyperStore with GDS.
- Tell you how to obtain the newly developed **GDS-capable S3 client technology** that can utilize HyperStore's GDS functionality. Clodian has worked together with NVIDIA to develop this new S3 client technology. (At this time, standard off-the-shelf S3 client applications do not yet support GDS.)

5.2.2.2. Installing HyperStore Software

After you've used the HyperStore GDS ISO to install the Rocky Linux OS (with included ConnectX drivers) on each of your HyperStore host machines, you can continue with the HyperStore preparation and installation in the standard way. For instructions see the ***Clodian HyperStore Installation Guide***.

After installing HyperStore 8.2 software, continue with the steps below.

5.2.2.3. Enabling GDS Support After Installing HyperStore Software

- ["Installing Custom NVIDIA S3 Libraries On Each HyperStore Node"](#) (page 147)
- ["Enabling GDS in Your HyperStore System"](#) (page 147)

Installing Custom NVIDIA S3 Libraries On Each HyperStore Node

For **each HyperStore host**, do the following:

1. Log in to the host by SSH and install NVIDIA Compute Unified Device Architecture (NVIDIA CUDA®) packages:

```
# dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
# dnf config-manager --add-repo https://developer.download.nvidia.com/compute/cuda/repos/rhel8/x86_64/cuda-rhel8.repo
# dnf -y install nvidia-gds
```

2. Copy custom S3 NVIDIA libraries to `/usr/local/cuda/targets/x86_64-linux/lib/`:

```
# cp /root/nvidia/cuobj_v0.4/src/lib/*.so* /usr/local/cuda/targets/x86_64-linux/lib/
# cp /root/nvidia/cuobj_v0.4/src/lib/*.so* /lib64/ # NEW FOR NON GPU CLIENTS
# cd /root/nvidia/cuobj_v0.4/src/
# dnf -y install cmake
# dnf -y install gcc-c++
# dnf -y install numactl-devel
# dnf install rdma-core-devel
# mkdir -p build
# cd build
# ln -s ../lib/libcufile.so.1.13.0 ../lib/libcufile.so
# cmake -DUSE_CUDA=OFF ..
# make
# cmake --install .
```

Enabling GDS in Your HyperStore System

After you've installed the custom NVIDIA S3 libraries on each HyperStore node, log in to the [Config Controller node](#) (the node on which you executed the HyperStore cluster installation) by SSH and run following `hsctl` commands to enable the GPUDirect Storage feature in your HyperStore system:

```
# hsctl config set common.rdma.enabled=true
# hsctl config apply common
# hsctl service restart s3 hyperstore cmc --nodes=ALL
```

To confirm that GDS has been successfully enabled in your system, [log in to the Clodian Management Console \(CMC\)](#) and go to the **"Cluster Information"** (page 490) page. There, the "GPUDirect" status field should show "Enabled":

5.2.2.4. Obtaining GDS-Capable S3 Client Technology

Standard off-the-shelf S3 client applications do not yet support GPUDirect Storage. Clodian has worked closely with NVIDIA to develop S3 client technology that integrates with the NVIDIA CUDA object client library to generate the special S3 extension headers that invoke HyperStore's GDS capabilities. Contact your Clodian representative for information about how to obtain S3 client technology that you can use together with a GDS-enabled HyperStore system.

5.3. Storage Policies

5.3.1. Storage Policies Feature Overview

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Supported Erasure Coding Configurations for a Single DC" \(page 149\)](#)
- ["Multi- Data Center Storage Policies" \(page 150\)](#)
- ["Hybrid Storage Policies" \(page 151\)](#)

Storage policies are ways of protecting data so that it's durable and highly available to users. The HyperStore system lets you pre-configure one or more storage policies. When users create a new storage bucket they choose which pre-configured storage policy to use to protect data in that bucket. **Users cannot create buckets until you have created at least one storage policy.**

For each storage policy that you create you can choose from either of two data protection methods:

- **Replication** — With replication, a configurable number of copies of each data object are maintained in the system, and each copy is stored on a different node. For example, with 3X replication 3 copies of each object are stored, with each copy on a different node.
- **Erasure coding** — With erasure coding, each object is encoded into a configurable number (known as the "k" value) of data fragments plus a configurable number (the "m" value) of redundant parity fragments. Each of an object's "k" plus "m" fragments is unique, and each fragment is stored on a different node. The object can be decoded from any "k" number of fragments. To put it another way: the object remains readable even if "m" number of nodes are unavailable. For example, in a 4+2 erasure coding configuration (4 data fragments plus 2 parity fragments), each object is encoded into a total of 6 unique fragments which are stored on 6 different nodes, and the object can be decoded so long as any 4 of those 6 fragments are available.

In general, erasure coding requires less storage overhead -- the amount of storage consumption above and beyond the original size of the stored objects, in order to ensure data persistence and availability -- than replication. Put differently, for a given degree of data protection erasure coding is more efficient in utilizing raw storage capacity than is replication.

For example, while 3X replication incurs a 200% storage overhead, 4+2 erasure coding incurs only a 50% storage overhead. Or stated in terms of storage capacity utilization efficiency, 3X replication is 33% efficient (for instance with 12TB of available storage capacity you can store 4TB of net object data) whereas 4+2 erasure coding is 67% efficient (with 12TB of available storage capacity you can store 8TB of net object data). On the other hand, erasure coding results in somewhat longer request processing latency than replication, due to the need for encoding/decoding.

In light of its benefits and drawbacks, erasure coding is best suited to **long-term storage of large objects that are infrequently accessed**.

Note Storage policies that you create with HyperStore 7.5.1 or later can use a hybrid approach as noted in ["Hybrid Storage Policies" \(page 151\)](#) below.

Regardless of whether you use replication or erasure coding, if your HyperStore system spans multiple data centers, for each storage policy you can also choose how data is allocated across your data centers — for

example, you could have a storage policy that for each S3 object stores 3 replicas of the object in each of your data centers; and a second storage policy that erasure codes objects and stores them in just one particular data center (for more information see ["Multi- Data Center Storage Policies"](#) (page 150)).

Also as part of the configuration options for each storage policy, you can choose whether to compress and/or encrypt stored objects. You can also optionally choose a workload type, such as one of the major backup applications (the workload type feature leverages rules based partitioning of object metadata storage to optimize performance of S3 *ListObjects* operations). Also, if you have set up [object metadata search](#) in your HyperStore system, for each storage policy you can choose whether to enable object metadata search for the buckets that use that storage policy.

Individual storage policies are **not confined to dedicated nodes or disks**. Instead, all policies utilize all the resources of your cluster, and data stored in association with a particular policy will tend to be spread fairly evenly across the cluster (with the exception that you can limit a policy to a particular data center as noted above). This helps to ensure that regardless of how many or what types of storage policies you configure, and regardless of how much data is stored in association with particular policies, the physical resources of your entire cluster — disks, CPU, RAM — will be used in an approximately even manner.

Note By default the system allows a maximum of 25 storage policies. This maximum is controlled by the configuration setting ["hyperstore.storagePolicy.max"](#) (page 779). Edit this setting if you want to have more than 25 storage policies in your HyperStore system.

IMPORTANT ! Erasure coding storage policies created with HyperStore versions 5.x or 6.x included a functionality that has been deprecated. The functionality, which identified more than $k+m$ endpoints as possible write destinations for an erasure coded object's $k+m$ fragments, ultimately was found to pose potential problems that outweighed the benefits (and so erasure coding storage policies created in HyperStore 7.0.1 and later returned to using the more conventional approach of identifying only $k+m$ endpoints for erasure coded writes). Support for legacy storage policies with the deprecated functionality will end in a future HyperStore release (the exact timing is still to be determined). Starting with HyperStore 7.5.2, the system automatically detects if you have any of these deprecated storage policies in your system. If you do have any of these deprecated storage policies, then:

- * A warning will display in the CMC **Dashboard**. The warning will indicate that you have one or more deprecated storage policies in your system and that you will need to contact your Cloudbian Sales Representative to plan for migrating data from those deprecated storage policies to supported storage policies.
- * In the CMC's **Storage Policies** page -- which lists all your storage policies -- a warning symbol will display next to the names of the deprecated storage policy or policies. Hover text shows the warning description.
- * Users creating new buckets will not be able to use a deprecated storage policy (such policies will not display in the list of available storage policies that users see when creating a new bucket).

5.3.1.1. Supported Erasure Coding Configurations for a Single DC

HyperStore supports several erasure coding configurations, in terms of "k" value (number of data fragments) and "m" value (number of parity fragments):

- 4+2
- 6+2

- 8+2
- 9+3
- 12+4

The choice among these supported EC configurations is largely a matter of how many HyperStore nodes you have in the data center. For example, compared to a 4+2 configuration, 6+2 EC provides the same degree of data availability assurance (objects can be read even if 2 of the involved nodes are unavailable), while delivering a higher level of storage efficiency (4+2 is 67% efficient whereas 6+2 is 75% efficient). So 6+2 may be preferable to 4+2 if you have at least 8 HyperStore nodes in the data center.

Likewise, 9+3 EC provides a higher degree of protection and availability than 6+2 EC (since with 9+3 EC, objects can be read even if 3 of the involved nodes are unavailable) while delivering the same level of storage efficiency (both 6+2 and 9+3 are 75% efficient). So 9+3 may be preferable to 6+2 if you have at least 12 HyperStore nodes in the data center.

Note If you want to use a $k+m$ configuration other than those mentioned above, contact Cloudbian Support or your Cloudbian Sales representative to see whether your desired configuration can be supported.

Note For detailed information on S3 write and read availability under various combinations of cluster size and storage policy configuration, see "[Storage Policy Resilience to Downed Nodes](#)" (page 156).

5.3.1.2. Multi- Data Center Storage Policies

If your HyperStore system is deployed across multiple data centers, for each storage policy that you create you can configure a data center assignment scheme for the policy. This determines which of your data centers to use for storing data, for each storage policy.

For storage policies that use replication only, in a multiple data center environment you can choose how many replicas to store in each data center -- for example, for each object store 3 replicas in DC1 and 2 replicas in DC2.

For erasure coding storage policies, you have the option of replicating the $k+m$ fragments in each of the participating DCs (so that each participating DC stores $k+m$ fragments), or distributing the $k+m$ fragments across the participating DCs (so that there are a combined total of $k+m$ fragments across the participating DCs).

For replicated erasure coding, by default your $k+m$ options are:

- 4+2
- 6+2
- 8+2
- 9+3
- 12+4

In each of the above configurations the $k+m$ fragments can be replicated across multiple DCs.

For distributed erasure coding, the supported options depend on how many data centers you are using in the storage policy. You must use at least 3 DCs for this type of policy, and by default your $k+m$ options are as indicated in the table below:

# of Participating DCs	Supported "k"+"m"	How Fragments Will Be Distributed
3	5+4	3 fragments per DC
	7+5	4 fragments per DC
4	8+4	3 fragments per DC
5	6+4	2 fragments per DC
6	8+4	2 fragments per DC
	7+5	2 fragments per DC
7	10+4	2 fragments per DC
8	10+6	2 fragments per DC
9	10+8	2 fragments per DC

Note If you want to use a $k+m$ configuration other than those mentioned above, contact Cloudian Support or your Cloudian Sales representative to see whether your desired configuration can be supported.

Note For any type of storage policy in a multiple data center environment, you have the option of configuring the policy such that data is stored in some of your data centers and not others — for example, you can create a policy that stores data in DC1 and DC2 but not in DC3. Note, however, that DC3 may be involved in processing S3 requests associated with buckets that use this policy. By default there is only one S3 service endpoint per region, and incoming S3 requests may resolve to any DC within the region. If the S3 Service in DC3 receives an S3 PUT request in association with a policy that stores data only in DC1 and DC2, it will transmit the uploaded object on to DC1 and DC2 (it will not be stored in DC3). Likewise, if DC3 receives an S3 GET request in association with a policy that stores data only in DC1 and DC2, then DC3's S3 Service will get the object from DC1 or DC2 and pass it on to the client. If you want more absolute barriers so that for example DC3 never touches DC2's data and vice-versa, you need to set up your system so those DCs are in different [service regions](#).

5.3.1.3. Hybrid Storage Policies

Starting with HyperStore version 7.5.1: When creating a new storage policy that uses an erasure coding data distribution scheme, you can set an object size threshold such that objects larger than the threshold will be erasure coded and objects at or smaller than the threshold will be protected by replication rather than erasure coding. By default the threshold is 200KiB.

This Hybrid Storage Policy feature allows for a "best of both" storage strategy, given that key data storage objectives are best served by erasure coding for large objects and by replication for small objects.

For more information see ["Create Storage Policy -- Hybrid Policy Threshold"](#) (page 538).

Note This feature is supported only for new storage policies created in HyperStore 7.5.1 or later.

5.3.2. Consistency Levels

To boost data durability and availability, HyperStore implements replication or erasure coding for object data and replication for object metadata. This entails distributing each object's data and metadata to multiple end-point nodes across the cluster. When you create storage policies, along with configuring a replication or erasure coding scheme you will also configure consistency levels for writes and reads. Consistency levels impose requirements as to **what portion of the data and metadata writes or reads associated with each S3 request must be successfully completed before the system can return a success response** to the S3 client. If the consistency requirements cannot be met for a given S3 request at a given time -- due to one or more endpoint nodes being unavailable -- an HTTP 503 error response is returned to the client. An endpoint node could be unavailable for example because the node is down, or is unreachable on the network, or (in the case of writes of object data) is in ["stop-write" condition](#).

Below is the list of consistency levels supported by the HyperStore system. Your consistency level options when configuring a storage policy will be limited by the data distribution scheme (replication or erasure coding, single DC or multi-DC) that you have selected for that policy.

- [ALL](#)
- [QUORUM](#)
- [EACH QUORUM](#)
- [LOCAL QUORUM](#)
- [ANY QUORUM](#)
- [ONE](#)

For detailed information on S3 write and read availability under various combinations of cluster size, data distribution scheme, and consistency level settings, see ["Storage Policy Resilience to Downed Nodes"](#) (page 156).

For a description of how HyperStore by default ensures **strong read-after-write consistency**, see ["Strong Read-After-Write Consistency"](#) (page 79).

Note In the case of writes, if the consistency requirement is met by something less than completing writes of all replicas (or all erasure coded fragments), then after returning a success response to the client the system continues to try to complete the remaining writes. If any of these writes fail they will later be recreated by [automatic data repair](#).

Note As an advanced option you can also configure "dynamic" consistency levels, whereby the system will try to achieve a "fallback" consistency level if the primary consistency level cannot be achieved. For more information see ["Dynamic Consistency Levels"](#) (page 80).

5.3.2.1. Note About Object Data Replica Reads

For replication based storage policies, the descriptions and examples in this documentation state that part of the read consistency requirement is being able to read X number of object data replicas. This is a simplification. Technically, what needs to be readable in order to satisfy a read consistency requirement is the **file digests** of X number of object data replicas. A [file digest](#) is an object data file "header" -- a small bit of file-identifying information including file name, size, timestamp, and MD5 hash -- which is stored in RocksDB on the same disk as the corresponding object data replica file. To determine whether or not an object data replica

file is present on a given endpoint, the system tries to read that object data replica's file digest. This is much faster than reading the object data file itself.

If the read consistency requirements are met for an S3 GET operation -- for reading the required number of object metadata replicas (in the Metadata DB) and the digests for the required number of object data replicas -- the system then retrieves just one object data replica file in order to return the object data to the S3 client. For example to meet a read consistency requirement of ALL, the system must be able to read all the object's metadata replicas in the Metadata DB, and all the object's data replicas' file digests in the Checksum DB -- and then it retrieves one object data replica and returns it to the client.

5.3.2.2. Note About Bucket Content List Reads

In the documentation of the supported consistency levels such as "ALL", "QUORUM", and so on, when read consistency requirements are discussed the focus is on reads of individual objects -- that is, the consistency requirements for successfully implementing S3 *GetObject* requests. It's worth noting however that your configured read consistency requirements also apply to bucket content list reads -- that is, when implementing S3 *ListObjects* requests.

Metadata for objects is stored in two different types of record in the Metadata DB: object-level records (with one such record for each object) and bucket-level records that identify the objects in a bucket (along with some metadata for each of those objects). Both types of object metadata are replicated to the same degree. So for example, in a 3X replication storage policy, for each object the object-level metadata record is replicated three times in the cluster and for each bucket the bucket-level object metadata records are replicated three times in the cluster.

A *GetObject* request requires reading the object's object-level metadata record and a *List Objects* request requires reading the bucket's bucket-level object metadata records. Whatever read consistency requirements you set for a storage policy apply not only to reads of individual objects but also to reads of bucket content lists. So for example if you use a QUORUM read consistency requirement, then in order to successfully execute a *List Objects* request the system must be able to read a QUORUM of the bucket-level object metadata records for the bucket.

5.3.3. Metadata Replication

Subjects covered in this section:

- ["Object Metadata Replication" \(page 153\)](#)
- ["System Metadata Replication" \(page 154\)](#)

5.3.3.1. Object Metadata Replication

HyperStore [object metadata](#) is stored in the Metadata DB, and is protected by replication. The degree to which object metadata is replicated depends on the type of storage policy being used.

Object Metadata in Replication Storage Policies

In the case of storage policies that protect object data by replication, the corresponding object metadata is replicated to the same degree as the object data. For example with a 3X replication storage policy, the system stores three replicas of each object (with each replica stored on a different node, in the HyperStore file system) and three replicas of each object's metadata (with each replica stored on a different node, in the Metadata DB). In a multi-DC replication storage policy that retains two replicas of each object in DC1 and one replica of each

object in DC2, the object metadata will also be replicated in this same manner -- two replicas in DC1 and one replica in DC2.

Object Metadata in Erasure Coding Storage Policies

In the case of storage policies that protect object data by erasure coding, the object metadata is protected by replication -- not erasure coding. This is because the object metadata associated with a given object is very small in size, and therefore not appropriate for erasure coding.

When you create an erasure coding storage policy through the CMC, you can choose the object metadata replication factor to use for that storage policy.

Note The object metadata replication factor is configurable for storage policies that you create in HyperStore version 7.5.3 and later. In erasure coding storage policies created in earlier versions of HyperStore the object metadata replication factor was $2m-1$ (by system default).

Two Types of Object Metadata Record

Metadata for an object is stored in two different types of record in the Metadata DB: one record specific to the object (called "skinny row" object metadata) and one or more bucket-level records that get updated with metadata for the object (called "wide row" object metadata). The latter are used for listing the contents of a bucket, among other purposes. For more detail see ["Object Metadata Storage in the Metadata DB" \(page 232\)](#).

Both types of object metadata are replicated to the same degree. So for example, in a 3X replication storage policy, for each object the "skinny row" object metadata is replicated three times in the cluster and the "wide row" object metadata is replicated three times in the cluster.

The skinny row metadata has the same key format as the object data (*bucketname/objectname*) and so has the same hash token and will be written to the same nodes as the object data -- or a subset of those nodes in the case of an erasure coding storage policy. The wide row metadata has a different key format and hash token and so may be written to different nodes than the object data if the cluster exceeds the minimize size required for the storage policy.

Within the Metadata DB, both types of object metadata record are part of the *UserData_<policyid>* keyspaces.

5.3.3.2. System Metadata Replication

HyperStore stores system metadata in several Metadata DB keyspaces: the *AccountInfo* keyspace (for user account information), the *Reports* keyspace (for usage reporting data), and the *Monitoring* keyspace (for system monitoring data). The replication level for this system metadata is set by the operator during HyperStore system installation (with a default of 3 replicas per service region).

Subsequent to installation the system may automatically raise the replication factor for the *AccountInfo* keyspace depending on the storage policies that you create. But for the *Reports* keyspace and the *Monitoring* keyspace the replication factor will remain as set during installation (3 replicas per service region by default).

Note If your original HyperStore installation was a version earlier than 7.5.3, then in your system the *Reports* and *Monitoring* keyspaces may have also been automatically adjusted to a higher replication factor in response to storage policies that you've created. But if your original HyperStore installation is

version 7.5.3 or later, then only the *AccountInfo* keyspace replication factor adjusts automatically in response to your storage policies.

5.3.4. Creating and Managing Storage Policies

In the CMC's **Storage Policies** page you can do everything you need to do in regard to creating and managing storage policies:

- ["Add a Storage Policy"](#) (page 513)
- ["Edit a Storage Policy"](#) (page 542)
- ["Designate a Default Storage Policy"](#) (page 543)
- ["Disable a Storage Policy"](#) (page 544)
- ["Delete a Storage Policy"](#) (page 544)

At all times you must have one and only one **default storage policy** defined in each of your HyperStore service regions. The default policy is the one that will be applied when users create new buckets without specifying a policy.

Note By default the system allows a maximum of 25 storage policies. This maximum is controlled by the configuration setting **"hyperstore.storagePolicy.max"** (page 779). Edit this setting if you want to have more than 25 storage policies in your HyperStore system.

5.3.4.1. Assigning a Storage Policy to a Bucket

When users create a new bucket they can select a storage policy to apply to the data that they will store in that bucket. This can be done either through the CMC or through other S3 applications that invoke HyperStore extensions to the standard S3 API.

IMPORTANT ! After a bucket is created, it cannot be assigned a different storage 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.

Assigning a Storage Policy to a Bucket (CMC)

CMC users can select a storage policy when they create a new bucket in the CMC's **Buckets** page. For details see ["Add a Bucket"](#) (page 307).

If a user does not explicitly select a policy when creating a new bucket, the system's current default storage policy is automatically applied to the bucket.

Assigning a Storage Policy to a Bucket (S3 API)

To select a storage policy for a new bucket, S3 client applications use an "x-gmt-policyid" request header when submitting a *PUT Bucket* request. For details see the S3 section in the **Cloudian HyperStore AWS APIs Support Reference**

5.3.4.2. Retrieving Storage Policy Usage Through the Admin API

Creating or modifying storage policies through the Admin API is not supported. However, you can use the API to retrieve a list of buckets that use each storage policy, with the [GET /bppolicy/bucketsperpolicy](#) method. For more information see the "bppolicy" section of the *Cloudian HyperStore Admin API Reference*.

5.3.5. Finding an Object's Replicas or EC Fragments

HyperStore lets you quickly determine which nodes are storing the replicas or erasure coded fragments of a specified S3 object. You can do this through either the CMC or the command line.

5.3.5.1. Finding an Object's Replicas or EC Fragments (CMC)

In the CMC you can find an object's replicas or fragments by using the [Object Locator](#) page.

5.3.5.2. Finding an Object's Replicas or EC Fragments (Command Line)

To find an object's replicas or erasure coded fragments locations using *hsstool* on the command line:

- [hsstool whereis](#)

5.3.6. Storage Policy Resilience to Downed Nodes

When nodes are down in your cluster, HyperStore S3 service availability for object writes and reads is a function of several factors including:

- The storage policy applied to the objects -- particularly the data distribution scheme (such as 3X replication or 4+2 erasure coding) and the configured consistency level requirements.
- The number of nodes in the cluster.
- The number of nodes that are down.

The tables that follow below indicate HyperStore S3 write and read availability for common single-DC storage policy configurations, **in scenarios where either one or two nodes are down**. For simplicity the tables refer to nodes as being "down", but the same logic applies if nodes are unavailable for other reasons such as being inaccessible on the network, or in a [stop-write condition](#), or in [maintenance mode](#).

Note The erasure coding tables assume that the system default behavior applies so that for each object with its $k+m$ number of data fragments, the number of object metadata replicas created is $2m-1$ (for example 3 object metadata replicas per object in the context of a 4+2 erasure coding policy). You may wish to discuss with your Cloudian representative whether the default of $2m-1$ is right for your use case, as versus the alternative of creating $2m+1$ number of object metadata replicas (which can be achieved by customizing a 'hidden' configuration setting during system set-up). The latter behavior places more storage load on the Metadata DB but also improves read availability in some scenarios.

Single DC, 3X Replication

With a **3X replication storage policy**, the system's ability to support writes and reads when 1 or 2 nodes are down depends on your consistency level configuration and on whether you have 3, 4, or 5 or more nodes in the cluster.

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	3 Nodes in Cluster	4 Nodes in Cluster	5 or More Nodes In Cluster
Writes	ALL	1	All writes fail All writes fail With 1 node down and 2 nodes up, the CL's requirement of 3 data replica writes and 3 metadata replica writes cannot be met for any objects.	Writes succeed for some objects and fail for others. Writes succeed for some objects and fail for others. With 1 node down and 3 nodes up, whether the CL's requirement of 3 data replica writes and 3 metadata replica writes can be met for a given object depends on where the object's 3 write endpoints are located (based on <i>bucketname/objectname</i> hash value). The requirement will be met for objects for which all 3 write endpoints are on the up nodes, and not for objects for which a write endpoint is on the down node.	Writes succeed for some objects and fail for others. Writes succeed for some objects and fail for others With 1 down and 4 or more nodes up, whether the CL's requirement of 3 data replica writes and 3 metadata replica writes can be met for a given object depends on where the object's 3 write endpoints are located (based on <i>bucketname/objectname</i> hash value). The requirement will be met for objects for which all 3 write endpoints are on the up nodes, and not for objects for which fewer than 3 write endpoints are on the up nodes.
		2	All writes fail All writes fail With 2 nodes down and 1 node up, the CL's requirement of 3 data replica writes and 3 metadata replica writes cannot be met for any objects.	All writes fail All writes fail With 2 nodes down and 2 nodes up, the CL's requirement of 3 data replica writes and 3 metadata replica writes cannot be met for any objects.	Writes succeed for some objects and fail for others Writes succeed for some objects and fail for others With 2 nodes down and 3 or more nodes up, whether the CL's requirement of 3 data replica writes and 3 metadata replica writes can be met for a given object depends on where the object's 3 write endpoints are located (based on <i>bucketname/objectname</i> hash value). The requirement will be met for objects for which all 3 write endpoints are on the up nodes, and not for

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	3 Nodes in Cluster	4 Nodes in Cluster	5 or More Nodes In Cluster
					objects for which fewer than 3 write endpoints are on the up nodes.
	QUORUM (default)	1	All writes succeed All writes succeed With 1 node down and 2 nodes up, the CL's requirement of 2 data replica writes and 2 metadata replica writes can be met for all objects.	All writes succeed All writes succeed With 1 node down and 3 nodes up, the CL's requirement of 2 data replica writes and 2 metadata replica writes can be met for all objects -- since even objects for which the down node is a write endpoint will still have 2 endpoints among the up nodes.	All writes succeed All writes succeed With 1 node down and 4 or more nodes up, the CL's requirement of 2 data replica writes and 2 metadata replica writes can be met for all objects -- since even objects for which the down node is a write endpoint will still have 2 endpoints among the up nodes.
		2	All writes fail All writes fail With 2 nodes down and 1 node up, the CL's requirement of 2 data replica writes and 2 metadata replica write cannot be met for any objects.	Writes succeed for some objects and fail for others Writes succeed for some objects and fail for others With 2 nodes down and 2 nodes up, whether the CL's requirement of 2 data replica writes and 2 metadata replica writes can be met for a given object depends on where the object's 3 write endpoints are located (based on <i>bucketname/objectname</i> hash value). The requirement will be met for objects for which 2 of the 3 write endpoints are on the up nodes, and not for objects for which only 1 write endpoint is on the up nodes.	Writes succeed for some objects and fail for others Writes succeed for some objects and fail for others With 2 nodes down and 3 or more nodes up, whether the CL's requirement of 2 data replica writes and 2 metadata replica writes can be met for a given object depends on where the object's 3 write endpoints are located (based on <i>bucketname/objectname</i> hash value). The requirement will be met for objects for which at least 2 write endpoints are on the up nodes, and not for objects for which fewer than 2 write endpoints are on the up nodes.
Reads	ALL	1	All reads fail All reads fail	Reads succeed for some objects and fail for others	Reads succeed for some objects and fail for others

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	3 Nodes in Cluster	4 Nodes in Cluster	5 or More Nodes In Cluster
			With 1 node down and 2 nodes up, the CL's requirement of 3 data replica reads and 3 metadata replica reads cannot be met for any objects.	Reads succeed for some objects and fail for others With 1 node down and 3 nodes up, whether the CL's requirement of 3 data replica reads and 3 metadata replica reads can be met for a given object depends on where the object's replicas are located. The requirement will be met for objects for which all 3 data replicas and all 3 metadata replicas are on the up nodes, and not for objects for which fewer than 3 data replicas and 3 metadata replicas are on the up nodes.	Reads succeed for some objects and fail for others With 1 node down and 4 or more nodes up, whether the CL's requirement of 3 data replica reads and 3 metadata replica reads can be met for a given object depends on where the object's replicas are located. The requirement will be met for objects for which all 3 data replicas and all 3 metadata replicas are on the up nodes, and not for objects for which fewer than 3 data replicas and 3 metadata replicas are on the up nodes.
		2	All reads fail All reads fail With 2 nodes down and 1 node up, the CL's requirement of 3 data replica reads and 3 metadata replica reads cannot be met for any objects.	All reads fail All reads fail With 2 nodes down and 2 nodes up, the CL's requirement of 3 data replica read and 3 metadata replica reads cannot be met for any objects.	Reads succeed for some objects and fail for others Reads succeed for some objects and fail for others With 2 nodes down and 3 or more nodes up, whether the CL's requirement of 3 data replica reads and 3 metadata replica reads can be met for a given object depends on where the object's replicas are located. The requirement will be met for objects for which all 3 data replicas and all 3 metadata replicas are on the up nodes, and not for objects for which fewer than 3 data replicas and 3 metadata replicas are

S3 Oper- ation Type	Configured Consistency Level (CL)	Number of Nodes Down	3 Nodes in Cluster	4 Nodes in Cluster	5 or More Nodes In Cluster
					on the up nodes.

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	3 Nodes in Cluster	4 Nodes in Cluster	5 or More Nodes In Cluster
	QUORUM (default)	1	All reads succeed All reads succeed With 1 node down and 2 nodes up, the CL's requirement of 2 data replica reads and 2 metadata replica reads can be met for all objects.	All reads succeed All reads succeed With 1 node down and 3 nodes up, the CL's requirement of 2 data replica reads and 2 metadata replica reads can be met for all objects -- since even objects for which the down node is a replica location will still have 2 replica locations among the up nodes.	All reads succeed All reads succeed With 1 node down and 4 or more nodes up, the CL's requirement of 2 data replica reads and 2 metadata replica reads can be met for all objects -- since even objects for which the down node is a replica location will still have 2 replica locations among the up nodes.
		2	All reads fail All reads fail With 2 nodes down and 1 node up, the CL's requirement of 2 data replica reads and 2 metadata replica reads cannot be met for any objects.	Reads succeed for some objects and fail for others Reads succeed for some objects and fail for others With 2 nodes down and 2 nodes up, whether the CL's requirement of 2 data replica reads and 2 metadata replica reads can be met for a given object depends on where the object's replicas are located. The requirement will be met for objects for which 2 data replicas and 2 metadata replicas are on the up nodes, and not for objects for which fewer than 2 data replicas and 2 metadata replicas are on the up nodes.	Reads succeed for some objects and fail for others Reads succeed for some objects and fail for others With 2 nodes down and 3 or more nodes up, whether the CL's requirement of 2 data replica reads and 2 metadata replica reads can be met for a given object depends on where the object's replicas are located. The requirement will be met for objects for which 2 data replicas and 2 metadata replicas are on the up nodes, and not for objects for which fewer than 2 data replicas and 2 metadata replicas are on the up nodes.
	ONE	1 or 2	All reads succeed All reads succeed	All reads succeed All reads succeed With 1 or 2 nodes down and 2 or more nodes up,	All reads succeed All reads succeed With 1 or 2 nodes down and 3 or more nodes up,

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	3 Nodes in Cluster	4 Nodes in Cluster	5 or More Nodes in Cluster
			With 1 or 2 nodes down and 2 or 1 nodes up, the CL's requirement of 1 data replica read and 1 metadata replica read can be met for all objects.	the CL's requirement of 1 data replica read and 1 metadata replica read can be met for all objects -- since even objects for which 2 replica locations are down will still have 1 replica location among the up nodes.	the CL's requirement of 1 data replica read and 1 metadata replica read can be met for all objects -- since even objects for which 2 replica locations are down will still have 1 replica location among the up nodes.

Single DC, $k+2$ Erasure Coding

By default HyperStore supports several $k+2$ erasure coding schemes: $4+2$, $6+2$, and $8+2$. With a storage policy based on $k+2$ erasure coding, the system's ability to support writes and reads when 1 or 2 nodes are down depends on your consistency level configuration and on whether you have $k+2$, $k+3$, or $k+4$ or more nodes in the cluster. For example with a $4+2$ policy ($k = 4$), write and read resilience depends in part on whether you have 6, 7, or 8 or more nodes in the cluster; and with a $6+2$ policy ($k = 6$), resilience depends in part on whether you have 8, 9, or 10 or more nodes in the cluster.

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	$k+2$ Nodes in Cluster	$k+3$ Nodes in Cluster	$k+4$ or More Nodes in Cluster
Writes	ALL	1	All writes fail All writes fail With 1 node down and $k+1$ nodes up, the CL's requirement of $k+2$ data fragment writes and 3 metadata replica writes cannot be met for any objects.	Writes succeed for some objects and fail for others Writes succeed for some objects and fail for others With 1 node down and $k+2$ nodes up, whether the CL's requirement of $k+2$ data fragment writes and 3 metadata replica writes can be met for a given object depends on where the object's data and metadata write endpoints are located (based on <i>buck- etname/objectname</i> hash	Writes succeed for some objects and fail for others Writes succeed for some objects and fail for others With 1 node down and $k+3$ nodes up, whether the CL's requirement of $k+2$ data fragment writes and 3 metadata replica writes can be met for a given object depends on where the object's data and metadata write endpoints are located (based on <i>buck- etname/objectname</i> hash

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	$k+2$ Nodes in Cluster	$k+3$ Nodes in Cluster	$k+4$ or More Nodes in Cluster
				value). The requirement will be met for objects for which all write endpoints are on the up nodes, and not for objects for which a write endpoint is on the down node.	value). The requirement will be met for objects for which all write endpoints are on the up nodes, and not for objects for which a write endpoint is on the down node.
		2	All writes fail All writes fail With 2 nodes down and k nodes up, the CL's requirement of $k+2$ data fragment writes and 3 metadata replica writes cannot be met for any objects.	All writes fail All writes fail With 2 nodes down and $k+1$ nodes up, the CL's requirement of $k+2$ data fragment writes and 3 metadata replica writes cannot be met for any objects.	Writes succeed for some objects and fail for others Writes succeed for some objects and fail for others With 2 nodes down and $k+2$ nodes up, whether the CL's requirement of $k+2$ data fragment writes and 3 metadata replica writes can be met for a given object depends on where the object's data and metadata write endpoints are located (based on <i>buck- etname/objectname</i> hash value). The requirement will be met for objects for which all write endpoints are on the up nodes, and not for objects for which a write endpoint is on the down node.
	QUORUM (default)	1	All writes succeed All writes succeed With 1 node down and $k+1$ nodes up, the CL's requirement of $k+1$ data fragment writes and 2 metadata rep-	All writes succeed All writes succeed With 1 node down and $k+2$ nodes up, the CL's requirement of $k+1$ data fragment writes and 2 metadata replica writes can be met for all objects -- since even objects for which the down node is 1 of the $k+2$ data fragment write endpoints and 1 of	All writes succeed All writes succeed With 1 node down and $k+3$ nodes up, the CL's requirement of $k+1$ data fragment writes and 2 metadata replica writes can be met for all objects -- since even objects for which the down node is 1 of the $k+2$ data fragment

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	$k+2$ Nodes in Cluster	$k+3$ Nodes in Cluster	$k+4$ or More Nodes in Cluster
			lica writes can be met for all objects -- since even objects for which the down node is 1 of the $k+2$ data fragment write endpoints and 1 of the 3 metadata write endpoints will still have enough data and metadata write endpoints on the up nodes to meet the requirement.	the 3 metadata write endpoints will still have enough data and metadata write endpoints on the up nodes to meet the requirement.	the 3 metadata write endpoints will still have enough data and metadata write endpoints on the up nodes to meet the requirement.
		2	All writes fail All writes fail With 2 nodes down and k nodes up, the CL's requirement of $k+1$ data fragment writes and 2 metadata replica writes cannot be met for any objects.	Writes succeed for some objects and fail for others Writes succeed for some objects and fail for others With 2 nodes down and $k+1$ nodes up, whether the CL's requirement of $k+1$ data fragment writes and 2 metadata replica writes can be met for a given object depends on where the object's data and metadata write endpoints are located (based on <i>buck- etname/objectname</i> hash value). The requirement will be met for objects for which $k+1$ data write endpoints and at least 2 metadata write endpoints are on the up nodes, and not for objects for which fewer endpoints are on the up nodes.	Writes succeed for some objects and fail for others Writes succeed for some objects and fail for others With 2 nodes down and $k+2$ nodes up, whether the CL's requirement of $k+1$ data fragment writes and 2 metadata replica writes can be met for a given object depends on where the object's data and metadata write endpoints are located (based on <i>buck- etname/objectname</i> hash value). The requirement will be met for objects for which $k+1$ data write endpoints and at least 2 metadata write endpoints are on the up nodes, and not for objects for which fewer endpoints are on the up nodes.

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	$k+2$ Nodes in Cluster	$k+3$ Nodes in Cluster	$k+4$ or More Nodes in Cluster
Reads	ALL	1 or 2	Reads succeed for some objects and fail for others Reads succeed for some objects and fail for others With 1 or 2 nodes down and $k+1$ or k nodes up, whether the CL's requirement of k data fragment reads and 3 metadata replica reads can be met for a given object depends on where the object's metadata replicas are located. With 1 or 2 nodes down there will still be k or more data fragments on the up nodes, so the data fragment part of the CL requirement will be met for all objects. But the metadata part of the requirement will be met for only objects for which all 3 metadata replicas are on the	Reads succeed for some objects and fail for others Reads succeed for some objects and fail for others With 1 or 2 nodes down and $k+2$ or $k+1$ nodes up, whether the CL's requirement of k data fragment reads and 3 metadata replica reads can be met for a given object depends on where the object's metadata replicas are located. With 1 or 2 nodes down there will still be k or more data fragments on the up nodes, so the data fragment part of the CL requirement will be met for all objects. But the metadata part of the requirement will be met for only objects for which all 3 metadata replicas are on the up nodes, and not for objects for which fewer than 3 metadata replicas are on the up nodes.	Reads succeed for some objects and fail for others Reads succeed for some objects and fail for others With 1 or 2 nodes down and $k+3$ or $k+2$ nodes up, whether the CL's requirement of k data fragment reads and 3 metadata replica reads can be met for a given object depends on where the object's metadata replicas are located. With 1 or 2 nodes down there will still be k or more data fragments on the up nodes, so the data fragment part of the CL requirement will be met for all objects. But the metadata part of the requirement will be met for only objects for which all 3 metadata replicas are on the up nodes, and not for objects for which fewer than 3 metadata replicas are on the up nodes.

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	$k+2$ Nodes in Cluster	$k+3$ Nodes in Cluster	$k+4$ or More Nodes in Cluster
			up nodes, and not for objects for which fewer than 3 metadata replicas are on the up nodes.		
	QUORUM (default)	1	All reads succeed All reads succeed With 1 node down and $k+1$ nodes up, the CL's requirement of k data fragment reads and 2 metadata replica reads can be met for all objects -- since even objects for which the down node holds 1 of the $k+2$ data fragments and 1 of the 3 metadata replicas will still have enough data fragments and metadata replicas on the up nodes to meet the requirement.	All reads succeed All reads succeed With 1 node down and $k+2$ nodes up, the CL's requirement of k data fragment reads and 2 metadata replica reads can be met for all objects -- since even objects for which the down node holds 1 of the $k+2$ data fragments and 1 of the 3 metadata replicas will still have enough data fragments and metadata replicas on the up nodes to meet the requirement.	All reads succeed All reads succeed With 1 node down and $k+3$ nodes up, the CL's requirement of k data fragment reads and 2 metadata replica reads can be met for all objects -- since even objects for which the down node holds 1 of the $k+2$ data fragments and 1 of the 3 metadata replicas will still have enough data fragments and metadata replicas on the up nodes to meet the requirement.
		2	Reads succeed for some objects and fail for others Reads succeed for some objects and fail for others	Reads succeed for some objects and fail for others Reads succeed for some objects and fail for others With 2 nodes down and	Reads succeed for some objects and fail for others Reads succeed for some objects and fail for others With 2 nodes down and

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	$k+2$ Nodes in Cluster	$k+3$ Nodes in Cluster	$k+4$ or More Nodes in Cluster
			ers With 2 nodes down and k nodes up, whether the CL's requirement of k data fragment reads and 2 metadata replica reads can be met for a given object depends on where the object's metadata replicas are located. With 2 nodes down there will still be k data fragments on the up nodes, so the data fragment part of the CL requirement will be met for all objects. But the metadata part of the requirement will be met only for objects for which 2 of the 3 metadata replicas are on the up nodes, and not for objects for which fewer than 2 metadata replicas are on the up nodes.	$k+1$ nodes up, whether the CL's requirement of k data fragment reads and 2 metadata replica reads can be met for a given object depends on where the object's metadata replicas are located. With 2 nodes down there will still be k data fragments on the up nodes, so the data fragment part of the CL requirement will be met for all objects. But the metadata part of the requirement will be met only for objects for which 2 of the 3 metadata replicas are on the up nodes, and not for objects for which fewer than 2 metadata replicas are on the up nodes.	$k+2$ nodes up, whether the CL's requirement of k data fragment reads and 2 metadata replica reads can be met for a given object depends on where the object's metadata replicas are located. With 2 nodes down there will still be k data fragments on the up nodes, so the data fragment part of the CL requirement will be met for all objects. But the metadata part of the requirement will be met only for objects for which 2 of the 3 metadata replicas are on the up nodes, and not for objects for which fewer than 2 metadata replicas are on the up nodes.

Single DC, $k+3$ Erasure Coding

By default HyperStore supports one $k+3$ erasure coding scheme: $9+3$. With a storage policy based on $k+3$ erasure coding, the system's ability to support writes and reads when 1 or 2 nodes are down depends on your consistency level configuration and on whether you have $k+3$, $k+4$, or $k+5$ or more nodes in the cluster. For example with a $9+3$ policy ($k = 9$), write and read resilience depends in part on whether you have 12, 13, or 14 or more nodes in the cluster.

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	$k+3$ Nodes in Cluster	$k+4$ Nodes in Cluster	$k+5$ or More Nodes in Cluster
Writes	ALL	1	All writes fail All writes fail With 1 node down and $k+2$ nodes up, the CL's requirement of $k+3$ data fragment writes and 5 metadata replica writes cannot be met for any objects.	Writes succeed for some objects and fail for others Writes succeed for some objects and fail for others With 1 node down and $k+3$ nodes up, whether the CL's requirement of $k+3$ data fragment writes and 5 metadata replica writes can be met for a given object depends on where the object's data and metadata write endpoints are located (based on <i>buck- etname/objectname</i> hash value). The requirement will be met for objects for which all write endpoints are on the up nodes, and not for objects for which a write endpoint is on the down node.	Writes succeed for some objects and fail for others Writes succeed for some objects and fail for others With 1 node down and $k+4$ nodes up, whether the CL's requirement of $k+3$ data fragment writes and 5 metadata replica writes can be met for a given object depends on where the object's data and metadata write endpoints are located (based on <i>buck- etname/objectname</i> hash value). The requirement will be met for objects for which all write endpoints are on the up nodes, and not for objects for which a write endpoint is on the down node.
		2	All writes fail All writes fail With 2 nodes down and $k+1$ nodes up, the CL's requirement of $k+3$ data fragment writes and 3 metadata replica writes cannot be met for any objects.	All writes fail All writes fail With 2 nodes down and $k+2$ nodes up, the CL's requirement of $k+3$ data fragment writes and 3 metadata replica writes cannot be met for any objects.	Writes succeed for some objects and fail for others Writes succeed for some objects and fail for others With 2 nodes down and $k+3$ nodes up, whether the CL's requirement of $k+3$ data fragment writes and 3 metadata replica writes can be met for a given object depends on where the object's data and metadata write endpoints are located (based

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	$k+3$ Nodes in Cluster	$k+4$ Nodes in Cluster	$k+5$ or More Nodes in Cluster
					on <i>buck- etname/objectname</i> hash value). The requirement will be met for objects for which all write endpoints are on the up nodes, and not for objects for which a write endpoint is on the down node.
	QUORUM (default)	1 or 2	All writes succeed All writes succeed With 1 or 2 nodes down and $k+2$ or $k+1$ nodes up, the CL's requirement of $k+1$ data fragment writes and 3 metadata replica writes can be met for all objects -- since even objects for which the down nodes are 2 of the $k+3$ data fragment write endpoints and 2 of the 5 metadata write endpoints will still have enough data and metadata write endpoints on the up nodes to meet the requirement.	All writes succeed All writes succeed With 1 or 2 nodes down and $k+3$ or $k+2$ nodes up, the CL's requirement of $k+1$ data fragment writes and 3 metadata replica writes can be met for all objects -- since even objects for which the down nodes are 2 of the $k+3$ data fragment write endpoints and 2 of the 5 metadata write endpoints will still have enough data and metadata write endpoints on the up nodes to meet the requirement.	All writes succeed All writes succeed With 1 or 2 nodes down and $k+4$ or $k+3$ nodes up, the CL's requirement of $k+1$ data fragment writes and 3 metadata replica writes can be met for all objects -- since even objects for which the down nodes are 2 of the $k+3$ data fragment write endpoints and 2 of the 5 metadata write endpoints will still have enough data and metadata write endpoints on the up nodes to meet the requirement.
Reads	ALL	1 or 2	Reads succeed for some objects	Reads succeed for some objects and fail for others	Reads succeed for some objects and fail for others

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	$k+3$ Nodes in Cluster	$k+4$ Nodes in Cluster	$k+5$ or More Nodes in Cluster
			and fail for others Reads succeed for some objects and fail for others With 1 or 2 nodes down and $k+2$ or $k+1$ nodes up, whether the CL's requirement of k data fragment reads and 5 metadata replica reads can be met for a given object depends on where the object's metadata replicas are located. With 1 or 2 nodes down there will still be more than k data fragments on the up nodes, so the data fragment part of the CL requirement will be met for all objects. But the metadata part of the requirement will be met for only objects for which all 5 metadata replicas are on the up nodes, and not for objects	Reads succeed for some objects and fail for others With 1 or 2 nodes down and $k+3$ or $k+2$ nodes up, whether the CL's requirement of k data fragment reads and 5 metadata replica reads can be met for a given object depends on where the object's metadata replicas are located. With 1 or 2 nodes down there will still be more than k data fragments on the up nodes, so the data fragment part of the CL requirement will be met for all objects. But the metadata part of the requirement will be met for only objects for which all 5 metadata replicas are on the up nodes, and not for objects for which fewer than 5 metadata replicas are on the up nodes.	Reads succeed for some objects and fail for others With 1 or 2 nodes down and $k+4$ or $k+3$ nodes up, whether the CL's requirement of k data fragment reads and 5 metadata replica reads can be met for a given object depends on where the object's metadata replicas are located. With 1 or 2 nodes down there will still be more than k data fragments on the up nodes, so the data fragment part of the CL requirement will be met for all objects. But the metadata part of the requirement will be met for only objects for which all 5 metadata replicas are on the up nodes, and not for objects for which fewer than 5 metadata replicas are on the up nodes.

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	$k+3$ Nodes in Cluster	$k+4$ Nodes in Cluster	$k+5$ or More Nodes in Cluster
			for which fewer than 5 metadata replicas are on the up nodes.		
	QUORUM (default)	1 or 2	All reads succeed All reads succeed With 1 or 2 nodes down and $k+2$ or $k+1$ nodes up, the CL's requirement of k data fragment reads and 3 metadata replica reads can be met for all objects -- since even objects for which the down nodes hold 2 of the $k+3$ data fragments and 2 of the 5 metadata replicas will still have enough data fragments and metadata replicas on the up nodes to meet the requirement.	All reads succeed All reads succeed With 1 or 2 nodes down and $k+3$ or $k+2$ nodes up, the CL's requirement of k data fragment reads and 3 metadata replica reads can be met for all objects -- since even objects for which the down nodes hold 2 of the $k+3$ data fragments and 2 of the 5 metadata replicas will still have enough data fragments and metadata replicas on the up nodes to meet the requirement.	All reads succeed All reads succeed With 1 or 2 nodes down and $k+4$ or $k+3$ nodes up, the CL's requirement of k data fragment reads and 3 metadata replica reads can be met for all objects -- since even objects for which the down nodes hold 2 of the $k+3$ data fragments and 2 of the 5 metadata replicas will still have enough data fragments and metadata replicas on the up nodes to meet the requirement.

Single DC, $k+4$ Erasure Coding

By default HyperStore supports one $k+4$ erasure coding scheme: $12+4$. With a storage policy based on $k+4$ erasure coding, the system's ability to support writes and reads when 1 or 2 nodes are down depends on your consistency level configuration and on whether you have $k+4$, $k+5$, or $k+6$ or more nodes in the cluster. For example with a $12+4$ policy ($k = 12$), write and read resilience depends in part on whether you have 16, 17, or

18 or more nodes in the cluster.

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	$k+4$ Nodes in Cluster	$k+5$ Nodes in Cluster	$k+6$ or More Nodes in Cluster
Writes	ALL	1	All writes fail All writes fail With 1 node down and $k+3$ nodes up, the CL's requirement of $k+4$ data fragment writes and 7 metadata replica writes cannot be met for any objects.	Writes succeed for some objects and fail for others Writes succeed for some objects and fail for others With 1 node down and $k+4$ nodes up, whether the CL's requirement of $k+4$ data fragment writes and 7 metadata replica writes can be met for a given object depends on where the object's data and metadata write endpoints are located (based on <i>buck-ename/objectname</i> hash value). The requirement will be met for objects for which all write endpoints are on the up nodes, and not for objects for which a write endpoint is on the down node.	Writes succeed for some objects and fail for others Writes succeed for some objects and fail for others With 1 node down and $k+5$ nodes up, whether the CL's requirement of $k+4$ data fragment writes and 7 metadata replica writes can be met for a given object depends on where the object's data and metadata write endpoints are located (based on <i>buck-ename/objectname</i> hash value). The requirement will be met for objects for which all write endpoints are on the up nodes, and not for objects for which a write endpoint is on the down node.
		2	All writes fail All writes fail With 2 nodes down and $k+2$ nodes up, the CL's requirement of $k+4$ data fragment writes and 7 metadata replica writes cannot be met for any objects.	All writes fail All writes fail With 2 nodes down and $k+3$ nodes up, the CL's requirement of $k+4$ data fragment writes and 7 metadata replica writes cannot be met for any objects.	Writes succeed for some objects and fail for others Writes succeed for some objects and fail for others With 2 nodes down and $k+4$ nodes up, whether the CL's requirement of $k+4$ data fragment writes and 7 metadata replica writes can be met for a given object depends on where the object's data and metadata write endpoints are located (based on <i>buck-ename/objectname</i> hash value). The requirement will be met for objects for

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	$k+4$ Nodes in Cluster	$k+5$ Nodes in Cluster	$k+6$ or More Nodes in Cluster
					which all write endpoints are on the up nodes, and not for objects for which a write endpoint is on the down node.
	QUORUM (default)	1 or 2	All writes succeed All writes succeed With 1 or 2 nodes down and $k+3$ or $k+2$ nodes up, the CL's requirement of $k+1$ data fragment writes and 4 metadata replica writes can be met for all objects -- since even objects for which the down nodes are 2 of the $k+4$ data fragment write endpoints and 2 of the 7 metadata write endpoints will still have enough data and metadata write endpoints on the up nodes to meet the requirement.	All writes succeed All writes succeed With 1 or 2 nodes down and $k+4$ or $k+3$ nodes up, the CL's requirement of $k+1$ data fragment writes and 4 metadata replica writes can be met for all objects -- since even objects for which the down nodes are 2 of the $k+4$ data fragment write endpoints and 2 of the 7 metadata write endpoints will still have enough data and metadata write endpoints on the up nodes to meet the requirement.	All writes succeed All writes succeed With 1 or 2 nodes down and $k+5$ or $k+4$ nodes up, the CL's requirement of $k+1$ data fragment writes and 4 metadata replica writes can be met for all objects -- since even objects for which the down nodes are 2 of the $k+4$ data fragment write endpoints and 2 of the 7 metadata write endpoints will still have enough data and metadata write endpoints on the up nodes to meet the requirement.
Reads	ALL	1 or 2	Reads succeed for some objects and fail for others Reads succeed for some objects	Reads succeed for some objects and fail for others Reads succeed for some objects and fail for others	Reads succeed for some objects and fail for others Reads succeed for some objects and fail for others

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	$k+4$ Nodes in Cluster	$k+5$ Nodes in Cluster	$k+6$ or More Nodes in Cluster
			and fail for others With 1 or 2 nodes down and $k+3$ or $k+2$ nodes up, whether the CL's requirement of k data fragment reads and 7 metadata replica reads can be met for a given object depends on where the object's metadata replicas are located. With 1 or 2 nodes down there will still be more than k data fragments on the up nodes, so the data fragment part of the CL requirement will be met for all objects. But the metadata part of the requirement will be met for only objects for which all 7 metadata replicas are on the up nodes, and not for objects for which fewer than 7 metadata replicas are on the up nodes.	With 1 or 2 nodes down and $k+4$ or $k+3$ nodes up, whether the CL's requirement of k data fragment reads and 7 metadata replica reads can be met for a given object depends on where the object's metadata replicas are located. With 1 or 2 nodes down there will still be more than k data fragments on the up nodes, so the data fragment part of the CL requirement will be met for all objects. But the metadata part of the requirement will be met for only objects for which all 7 metadata replicas are on the up nodes, and not for objects for which fewer than 7 metadata replicas are on the up nodes.	With 1 or 2 nodes down and $k+5$ or $k+4$ nodes up, whether the CL's requirement of k data fragment reads and 7 metadata replica reads can be met for a given object depends on where the object's metadata replicas are located. With 1 or 2 nodes down there will still be more than k data fragments on the up nodes, so the data fragment part of the CL requirement will be met for all objects. But the metadata part of the requirement will be met for only objects for which all 7 metadata replicas are on the up nodes, and not for objects for which fewer than 7 metadata replicas are on the up nodes.

S3 Operation Type	Configured Consistency Level (CL)	Number of Nodes Down	$k+4$ Nodes in Cluster	$k+5$ Nodes in Cluster	$k+6$ or More Nodes in Cluster
	QUORUM (default)	1 or 2	All reads succeed All reads succeed With 1 or 2 nodes down and $k+3$ or $k+2$ nodes up, the CL's requirement of k data fragment reads and 4 metadata replica reads can be met for all objects -- since even objects for which the down nodes hold 2 of the $k+4$ data fragments and 2 of the 7 metadata replicas will still have enough data fragments and metadata replicas on the up nodes to meet the requirement.	All reads succeed All reads succeed With 1 or 2 nodes down and $k+4$ or $k+3$ nodes up, the CL's requirement of k data fragment reads and 4 metadata replica reads can be met for all objects -- since even objects for which the down nodes hold 2 of the $k+4$ data fragments and 2 of the 7 metadata replicas will still have enough data fragments and metadata replicas on the up nodes to meet the requirement.	All reads succeed All reads succeed With 1 or 2 nodes down and $k+5$ or $k+4$ nodes up, the CL's requirement of k data fragment reads and 4 metadata replica reads can be met for all objects -- since even objects for which the down nodes hold 2 of the $k+4$ data fragments and 2 of the 7 metadata replicas will still have enough data fragments and metadata replicas on the up nodes to meet the requirement.

Additional Considerations for S3 Availability

S3 Availability for Large Objects

For uploading larger objects, S3 client applications typically use Multipart Upload -- which breaks the object data into contiguous parts and uploads each part separately. Amazon recommends that client applications use Multipart Upload for objects 100MB and larger. Within HyperStore, each part is assigned its own hash value and is separately replicated or erasure coded within the cluster.

Also, for each object larger than 10MB -- or for Multipart Uploads, for each object part larger than 10MB -- the HyperStore system breaks the object or part into multiple "chunks" of 10MB or smaller. Each chunk is assigned its own hash value and is separately replicated or erasure coded within the cluster.

For S3 write and read availability for large objects, within HyperStore the write or read of **each part and/or chunk** must satisfy the object data consistency level requirement in order for the S3 object upload or object read operation as a whole to succeed. (The metadata requirements are not impacted by object size since the metadata records are for the object as a whole and not for each part or chunk.)

In terms of the Result categories in the tables above, the consequences of an object being large are as follows:

- **"All writes/reads succeed"** -- No effect. Just as writes or reads of all individual small objects succeed, so too do writes or reads of all large object parts and chunks.
- **"All writes/reads fail"** -- No effect. Large objects fail just as small ones do.
- **"Writes/Reads succeed for some objects and fail for others."** -- Here, the object data consistency criteria that determine which objects succeed (as displayed when you click the Result text in the tables) must be met by **each part and/or chunk** in order for the S3 object upload or object read operation as a whole to succeed. Consequently, in these scenarios, the write or read of a large object with multiple parts and/or chunks has a greater chance of failing than the write or read of a small object.

Writes of Object Metadata Per Bucket

For an S3 write operation to succeed, your configured write consistency requirement must be met for the object data and also for the object metadata. The system actually writes two types of object metadata record -- a record specific to the object and a bucket-level record that gets updated with metadata for each object in the bucket (the bucket-level record is used for listing the contents of a bucket, among other things).

For a given object, the per-object metadata record has the same key format as the object data (*bucketname/objectname*) and therefore is assigned the same hash token as the object data and will be written to the same endpoint nodes as the object data -- or a subset of those nodes, in the case of erasure coding storage policies. By contrast, the per-bucket object metadata record has a different key format and therefore a different hash token, and so may be written to different endpoint nodes than the object data. The per-bucket object metadata record is replicated to the same degree as the per-object metadata record -- for example, three times in a 3X replication storage policy -- and must meet the same configured write consistency requirements.

To limit the complexity of the tables above, in the Result descriptions the references to object metadata are referring only to the per-object metadata record. In terms of the Result categories in the tables, the consequences of the system's need to also write per-bucket object metadata -- potentially to different endpoint nodes than the object data and per-object metadata -- are as follows:

- **"All writes succeed"** -- No effect. In these scenarios writes succeed for the per-bucket object metadata also.
- **"All writes fail"** -- No effect. In these scenarios S3 writes fail regardless of the per-bucket object metadata considerations.
- **"Writes succeed for some objects and fail for others."** -- In these scenarios, writes succeed for objects for which the consistency requirement can be met for object data, per-object metadata, **and** per-bucket object metadata. Writes fail for objects for which the consistency requirement cannot be met for either the object data, or the per-object metadata, or the per-bucket metadata. In such down node scenarios where writes succeed for some objects and fail for others, whether the write of a given object succeeds or fails is determined not only by the hash token assigned to the object's data and per-object metadata record but also by the hash token assigned to the per-bucket object metadata record.

Note Per-bucket object metadata is not used for S3 object reads and has **no impact on any of the read availability scenarios** in the tables above. Only the per-object metadata record is relevant to object reads.

Redis DB Access

The HyperStore system's S3 Service needs information from the Redis Credentials database and the Redis QoS database in order to process S3 write and read requests. In most cases -- particularly in larger clusters -- 1 or 2 nodes being down in your system will not impact the availability of these databases (which are implemented across multiple nodes in master-slave relationships). If problems within your system were to lead to either of these databases being completely offline -- such as if all the nodes running Redis Credentials are down, or all the nodes running Redis QoS are down -- the S3 layer can use cached Redis data for a while. But if the cached data expires and a Redis database is still completely offline, then S3 requests will start to fail.

5.4. Security and Privacy Features

5.4.1. HTTPS (SSL/TLS)

5.4.1.1. HTTPS Feature Overview

By default the following HyperStore services support client access via HTTPS, with each service using its own self-signed TLS/SSL certificate that is automatically generated during HyperStore installation:

- Cloudian Management Console (CMC) Service
- Admin Service
- IAM Service
- S3 Service

Note HTTPS for the S3 Service is enabled by default only in fresh installations of HyperStore version 7.4 or newer. For systems that were originally installed as a version older than 7.4, and then subsequently upgraded, HTTPS for the S3 Service is disabled by default (the upgrade does not automatically enable it). If your original HyperStore installation was older than version 7.4 and you have not yet enabled HTTPS for the S3 Service, you can enable it by [generating a self-signed certificate](#) for the S3 Service or [importing a CA-signed certified](#) for the S3 Service. Either of those operations automatically enables HTTPS for the S3 Service if it is not already enabled.

For the SQS Service, HTTPS is disabled by default but you can enable it with the configuration setting `"sqs.tls.enabled / sqs.tls.required"` (page 846)

HyperStore provides you tools that simplify several SSL certificate management tasks that you may wish to perform for the CMC, Admin, IAM, S3, and/or SQS services:

- Generate a new self-signed certificate (see ["Generating a New Self-Signed Certificate for a Service"](#) (page 178))

- Generate a Certificate Signing Request (CSR) to submit to a Certificate Authority (CA) (see ["Generating a Certificate Signing Request \(CSR\)"](#) (page 180))
- Import a CA-signed certificate and associated intermediate and root certificates (see ["Importing a CA-Signed Certificate"](#) (page 182))

You may also wish to block regular HTTP access to these HyperStore services, so that only HTTPS access is allowed. For information on this task see ["Disabling Regular HTTP Access to HyperStore Services"](#) (page 185).

Because HyperStore services sometimes make outbound HTTPS connections -- when implementing cross-region replication or auto-tiering, for example -- you may also wish to import one or more root Certificate Authorities' certificates to the truststore used by HyperStore services, so that HyperStore services making outbound HTTPS connections can trust certificates signed by those CAs. This is necessary only for private root CAs, since the major public CAs are already in the truststore by default.

Adding a private root CA's certificate to HyperStore's truststore is also necessary if you import a CA-signed certificate to the S3 Service, the Admin Service, or the IAM service, and the root CA in the certificate chain is a private CA. This is because the CMC acts as a client to those services.

For information on adding a root CA's certificate to the truststore, see ["Adding a Private Root CA's Certificate to HyperStore's Truststore"](#) (page 185).

Note

* HyperStore HTTPS listeners support **TLS v1.2 and TLS v1.3** and will not accept client connections that use TLS versions older than 1.2.

* The Admin Service, along with supporting HTTPS, also supports HTTP(S) Basic Authentication. For more information -- including how to find or change the default password for authentication -- see the Introduction section in the *Cloudian HyperStore Admin API Reference*.

5.4.1.2. Generating a New Self-Signed Certificate for a Service

Note This task is supported by the HyperStore installer, and the procedure below describes how to use the installer to perform this task. If you prefer, you can use the HyperStore command line tool *hsctl* to perform this task rather than the installer. Using *hsctl* for this task provides you additional options not available through the installer: the option to generate self-signed certificates for all HTTPS-enabled services in one operation; the option to assign a non-default lifespan to the certificate(s) (the default lifespan is one year); and for the S3 Service in a multi-region deployment, the option to generate different certificates for each region. However, if you use *hsctl* to generate a self-signed certificate, afterwards you still must use the installer to push the change out to the cluster and restart the affected service (Steps 5 and 6 in the procedure below). For more information about using *hsctl* to generate self-signed certificates, log into the Config Controller node and then on the terminal command line enter *hsctl cert self-signed --help*

Follow the steps below to use the HyperStore installer to generate a new self-signed certificate for an HTTPS-enabled HyperStore service. This operation will **overwrite** the certificate that the service is currently using.

1. Log into the Config Controller node and change into */opt/cloudian-staging/8.2* (the installation staging directory). Then launch the installer:

```
# ./cloudianInstall.sh
```

If you are using the HyperStore Shell

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Config Controller node you can launch the installer with this command:

```
$ hspkg install
```

Once launched, the installer's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

- At the installer's main menu enter **4** for "Advanced Configuration Options". Then in the Advanced Configuration Options menu enter **e** for "Configure SSL for Admin, CMC, S3, IAM/STS, and SQS services". This takes you to the "SSL Certificate Management" menu:

```
SSL certificate management
-----

a ) Configure Admin service
b ) Configure CMC service
c ) Configure S3 service
d ) Configure IAM/STS service
e ) Configure SQS service
x ) Return to previous menu

Choice: █
```

- From the SSL Certificate Management menu select the service for which you want to generate a new self-signed certificate. This takes you to the "<Service Name> SSL Certificate Management" menu. The example below is for the S3 Service, but regardless of which service you are working with you will see the same menu options.

```
S3 service SSL certificate management
-----

a ) Generate a new Self-Signed Certificate
b ) Import CA-signed certificates (in PEM or PKCS12 format)
x ) Return to previous menu

Choice: █
```

- Enter **a** for "Generate a new Self-Signed Certificate". Then at the prompt, confirm that you want to generate a new self-signed certificate for this service. The installer will then generate the certificate.
- Navigate back to the installer's main menu, and then enter **2** for "Cluster Management". Then from the Cluster Management menu enter **b** for "Apply Configuration to Cluster", and follow the prompts to push to the cluster.

6. Exit the installer. On the Config Controller node, [use the `hsctl` command line tool to restart the service for which you generated a new self-signed certificate.](#)

5.4.1.3. Generating a Certificate Signing Request (CSR)

Note This task is not supported by the HyperStore installer. You must use the HyperStore command line tool `hsctl` to perform this task. The `hsctl` tool can be run by the `root` user or by a [HyperStore Shell](#) user who has the Trusted role.

To generate a certificate signing request (CSR) for one or more HyperStore services: Log into the Config Controller node, change into a working directory, and run the `hsctl` command described below. This will generate one or more CSR / private key pairs in the working directory in which you run the command.

```
hsctl cert req [--combined] [--service=enum] [--domains=string] [--country="string"]
[--state="string"] [--locality="string"] [--organization="string"] [--organization-unit="string"]
[--email="string"]
```

By default the `hsctl cert req` command:

- Generates a separate CSR and corresponding private key for each of the HTTPS-enabled HyperStore services -- the S3 service, the CMC service, the Admin service, and the IAM service. If you have multiple service regions in your system, then for the S3 service a separate CSR (and private key) will be generated for each service region.
- Uses the service endpoints configured in the HyperStore system, including a wildcarded endpoint for the S3 service in each region.
- Uses Cloudian and generic organization identifiers rather than identifiers for your organization.

You can modify this behavior by using any of the command options described below:

--combined

By default, separate CSRs will be generated for each of the HTTPS-enabled HyperStore services. If you prefer to generate a **single combined CSR** for all of the HTTPS-enabled services, use the *--combined* option (and omit the *--service* option described below).

Note This is a valid option only if the CA that you are using supports the use of the subject alternative name (SAN) field. A combined CSR for HyperStore services will list multiple service endpoints in the SAN field.

--service

Use the *--service* option if you want to generate a CSR for just one HyperStore service. Supported values are:

- s3
- cmc
- admin
- iam

Note The *--service* value must be lower case.

If you do not use the `--service` option, a CSR will be generated for each of the services listed above. (Or, if you omit the `--service` option and use the `--combined` option, a single combined CSR will be generated for all of the services listed above.)

`--domains`

By default, CSRs for HyperStore services will be created using the service domains that are configured in the system. Use the `--domains` option if you want to explicitly specify the domain(s) to use in the CSR for a service.

You can check to see the default domains that will be used in CSRs by running the command `hsctl cert list-domains`. For example:

```
[root]# hsctl cert list-domains
cmc:
cmc.mycloudianhyperstore.com

iam:
iam.mycloudianhyperstore.com

admin:
s3-admin.mycloudianhyperstore.com

s3-region1:
s3-region1.mycloudianhyperstore.com
*.s3-region1.mycloudianhyperstore.com
```

Note that by default, the CSR for the S3 service will place all endpoints in the Subject Alternative Name (SAN) field and will include wildcarded versions of the explicit endpoints. In the example above, the wildcarded endpoint is `*.s3-region1.mycloudianhyperstore.com`. The wildcarded S3 endpoint is needed if your S3 service is going to support Virtual Host Style Access by S3 clients -- an access method for which the bucket name is used as a sub-domain of the S3 endpoint. This is the S3 access method that AWS recommends.

The most likely use case for the `--domains` option is if you intend to submit the S3 service CSR to a CA that does not support SAN and/or does not support wildcards in endpoints. In that case you could generate a CSR for the S3 service like this, for example:

```
hsctl cert req --service=s3 --domains=s3-region1.mycloudianhyperstore.com
```

This command would generate an S3 service CSR in which there is only one endpoint -- `s3-region1.mycloudianhyperstore.com` -- and no wildcarding.

Note that the domain that you specify **must** match one of the domains configured for the service in the system (one of the domains returned by `hsctl cert list-domains`).

IMPORTANT ! If the certificate for your S3 service does not include a wildcarded endpoint, then S3 client applications using your service will have to use Path-Based Access (in which the bucket name is part of the URI path) rather than Virtual Host Style Access.

Note If you wish you can use the `--domains` option to specify multiple domains with comma separation (`--domains=<domain1>,<domain2>,...`). You might use this approach for example if your CA supports SAN but not wildcarding. If you list multiple domains, the first-listed domain will be placed in the Common Name (CN) field of the CSR and any additional domains will be placed in the SAN field.

`--country, --state, etc`

Use these options to identify your organization.

- Quote-enclose each value
- Because the values are quote-enclosed, you can use spaces within the values. For example, `--locality="San Francisco"`.
- For `--country` use a 2-character ISO format country code.
- If you omit any of these identifiers, default values will be used as follows: `--country="US" --state="California" --locality="San Mateo" --organization="Cloudian, Inc" --organization-unit="Self-Signed Certificates Unit" --email="example@example.com"`. For example if you specify most of the identifiers but omit the `--organization-unit`, then within the CSR that's generated the organization unit will be indicated as "Self-Signed Certificates Unit".

Example (CSRs for each service):

```
[root]# hscctl cert req --country="US" --state="MA" --locality="Foxborough"
--organization="Patriots" --email="bill@gillettestadium.com"
Generating certificate signing request (CSR) and private key for 'cmc'
Writing CSR to file: './cmc.csr.pem'
Writing key to file: './cmc.key.pem'
Generating certificate signing request (CSR) and private key for 'iam'
Writing CSR to file: './iam.csr.pem'
Writing key to file: './iam.key.pem'
Generating certificate signing request (CSR) and private key for 'admin'
Writing CSR to file: './admin.csr.pem'
Writing key to file: './admin.key.pem'
Generating certificate signing request (CSR) and private key for 's3-region1'
Writing CSR to file: './s3-region1.csr.pem'
Writing key to file: './s3-region1.key.pem'
```

Example (combined CSR for all services):

```
[root]# hscctl cert req --combined --country="US" --state="MA" --locality="Foxborough"
--organization="Patriots" --email="bill@gillettestadium.com"
Generating certificate signing request (CSR) and private key for 'combined'
Writing CSR to file: './combined.csr.pem'
Writing key to file: './combined.key.pem'
```

Example (CSR for one specified service):

```
[root]# hscctl cert req --service=cmc --country="US" --state="MA" --locality="Foxborough"
--organization="Patriots" --email="bill@gillettestadium.com"
Generating certificate signing request (CSR) and private key for 'cmc'
Writing CSR to file: './cmc.csr.pem'
Writing key to file: './cmc.key.pem'
```

5.4.1.4. Importing a CA-Signed Certificate

Note This task is supported by the HyperStore installer, and the procedure below describes how to use the installer to perform this task. If you prefer, you can use the HyperStore command line tool `hscctl` to perform this task rather than the installer. Using `hscctl` for this task provides you additional options not available through the installer: the option to import CA-signed certificates for all HTTPS-enabled services in one operation (or a wildcard certificate to be applied to all services); and for the S3 Service in a multi-region deployment, the option to import different CA-signed certificates for each region. However,

if you use `hscctl` to import a CA-signed certificate, afterwards you still must use the installer to push the change out to the cluster and restart the affected service (Steps 6 and 7 in the procedure below). For more information about using `hscctl` to import CA-signed certificates, log into the Config Controller node and then on the terminal command line enter `hscctl cert import --help`

Follow the steps below to use the HyperStore installer to import a certificate signed by a public or private CA, for use by an HTTPS-enabled HyperStore service.

1. Log into the Config Controller node, and in either `/opt/cloudian-staging/8.2` (the installation staging directory) or a working directory, place the file(s) from which you are going to import a CA-signed certificate:
 - The file(s) must be in PEM format or PKCS12 format (with filename extension `.p12` or `.pfx`). No other formats are supported.
 - The entire trust chain must be present in the file(s): the final CA-signed certificate, any intermediate certificates, and the CA root certificate. These may all be present in a single file, or may be divided into multiple files.
 - The private key from which the CSR was generated must be present, either as a dedicated key file (such as a `.key.pem` file) or within one of the other files.
 - If you are using the [HyperStore Shell \(HSH\)](#), you will not be able to save files into `/opt/cloudian-staging/8.2`. Instead, save the files into a working directory under your home directory.
2. Still logged into the Config Controller node, change into `/opt/cloudian-staging/8.2`. Then launch the installer:

```
# ./cloudianInstall.sh
```

If you are using the HyperStore Shell

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Config Controller you can launch the installer with this command:

```
$ hspkg install
```

Once launched, the installer's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

3. At the installer's main menu enter **4** for "Advanced Configuration Options". Then in the Advanced Configuration Options menu enter **e** for "Configure SSL for Admin, CMC, S3, IAM/STS, and SQS services". This takes you to the "SSL Certificate Management" menu:

```
SSL certificate management
-----

a ) Configure Admin service
b ) Configure CMC service
c ) Configure S3 service
d ) Configure IAM/STS service
e ) Configure SQS service
x ) Return to previous menu
```

```
Choice: █
```

4. From the SSL Certificate Management menu select the service for which you want to import a CA-signed certificate. This takes you to the "<Service Name> SSL Certificate Management" menu. The example below is for the S3 Service, but regardless of which service you are working with you will see the same menu options.

```
S3 service SSL certificate management
-----

a ) Generate a new Self-Signed Certificate
b ) Import CA-signed certificates (in PEM or PKCS12 format)
x ) Return to previous menu
```

```
Choice: █
```

5. Enter **b** for "Import CA-Signed Certificates (in PEM or PKSC12 format)". You will then be prompted to enter filenames:
- If the file(s) are in any directory other than `/opt/cloudian-staging/8.2`, include the full path to the file(s).
 - At each successive "Enter filename" prompt, type one filename (including path if applicable) then press Enter.
 - When there are no more files to enter, at the "Enter filename" prompt simply press Enter without typing a filename. This will trigger the import of the CA-signed certificate file(s).
 - If any of the files that you specified are password-protected you will be prompted to supply the password.
6. After the import operation completes, navigate back to the installer's main menu, and then enter **2** for "Cluster Management". Then from the Cluster Management menu enter **b** for "Apply Configuration to Cluster", and follow the prompts to push to the cluster.
7. Exit the installer. On the Config Controller node, [use the `hsctl` command line tool to restart the service for which you imported a CA-signed certificate](#).

5.4.1.5. Adding a Private Root CA's Certificate to HyperStore's Truststore

Note This task is not supported by the HyperStore installer. You must use the HyperStore command line tool `hsctl` to perform this task. The `hsctl` tool can be run by the `root` user or by a [HyperStore Shell](#) user who has the Trusted role.

In some contexts HyperStore services may act as clients to other services that present SSL certificates. Examples include, potentially:

- When the S3 Service implements cross-region replication, or auto-tiering.
- When the CMC connects to an LDAP server (to authenticate users logging into the CMC, if you have enabled LDAP authentication for one or more user groups)
- When the CMC connects to the Admin Service
- When the CMC connects to the S3 Service (applicable only if you've set [s3.tls.required](#) to `true` so that the CMC connects to the S3 Service's HTTPS port; it defaults to `false` so that the HTTP port is used)
- When the CMC connects to the IAM Service (applicable only if you've set [iam.tls.required](#) to `true` so that the CMC connects to the IAM Service's HTTPS port; it defaults to `false` so that the HTTP port is used)

By default HyperStore's truststore includes all the major public root CAs. If HyperStore services are connecting to services that are using a private CA as their root CA, you can add that CA's certificate to HyperStore's truststore so that HyperStore services trust certificates signed by that CA.

To add a private root CA's certificate to HyperStore's truststore:

1. Log into the Config Controller node, and copy the private root CA's certificate file into a working directory.
2. Change into the working directory and run this `hsctl` command:

```
# hsctl cert ca trust add <CA certificate filename>
```

This operation will import any PEM formatted private CA root certificates found inside the input file.

3. Run this command to apply the change to the cluster:

```
# hsctl config apply pki.trust
```

4. On the Config Controller node, [use the `hsctl` command line tool to restart the services that you want to trust the private root CA that you added](#). Typically this would be the S3 Service (such as in the case of cross-region replication) and/or the CMC Service (such as if the CMC is accessing an LDAP server when authenticating users).

5.4.1.6. Disabling Regular HTTP Access to HyperStore Services

For security purposes you may wish to disable regular (non-secure) HTTP access to HyperStore public services so that access to the services is exclusively through HTTPS. For best protection you can block regular HTTP access in **both** of these ways:

- Configure a **firewall** to block S3 HTTP access (port 80), CMC HTTP access (port 8888), IAM HTTP access (port 16080), and Admin Service HTTP access (port 18081). If you are using the HyperStore firewall, see ["Customizing the HyperStore Firewall"](#) (page 853) for instructions. (If you have not yet enabled the HyperStore firewall, and want to do so, see ["Enabling or Disabling the HyperStore](#)

[Firewall" \(page 851\)\)](#).

- Set the **HyperStore system configuration** so that the services require clients to use HTTPS connections:
 - For the Admin Service, set [admin.tls.required](#) to *true* if it is not already set to *true* (it defaults to *true* if your original HyperStore install was version 6.0.2 or newer, and defaults to *false* if your original HyperStore install was older than 6.0.2)
 - For the CMC, set [cmc.tls.required](#) to *true* if it is not already set to *true* (it defaults to *true*)
 - For the IAM Service, set [iam.tls.required](#) to *true* (it defaults to *false*)
 - For the S3 Service, set [s3.tls.required](#) to *true* (it defaults to *false*)
 - For the SQS Service, set [sqs.tls.required](#) to *true* (it defaults to *false*)

5.4.2. Server-Side Encryption

5.4.2.1. Server-Side Encryption Feature Overview

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Server-Side Encryption Configuration Changes Do Not Apply Retroactively" \(page 187\)](#)
- ["Server-Side Encryption and Auto-Tiering" \(page 188\)](#)
- ["Server-Side Encryption and Cross-Region Replication" \(page 188\)](#)

The HyperStore system supports server-side encryption (SSE) of data at rest. Several different methods of server-side encryption are supported:

- Encryption using **HyperStore system-generated encryption keys** (regular SSE)
- Encryption using encryption keys managed in part by a **Key Management System that supports the Key Management Interoperability Protocol** (SSE-KMIP) (popular KMIP compliant systems include Fortanix, HashiCorp, and Thales)
- Encryption using encryption keys managed by the **Amazon Web Services Key Management Service** (AWS KMS)
- Encryption using **customer-provided encryption keys** (SSE-C)

The selection of whether to use server-side encryption, and which method to use, can be made at the **storage policy level** (so that a default encryption method is applied to all objects uploaded to any bucket that uses the storage policy), at the **bucket level** (so that a default encryption method is applied to all objects uploaded to the bucket), or at the **object level** (as specified by headers in the object upload request). However not all encryption types can be configured at every level: for example, SSE-KMIP can only be configured at the storage policy and bucket levels and SSE-C can only be configured at the object level.

The table below summarizes the server-side encryption options available at the levels of storage policy default encryption, bucket-specific default encryption, and object-specific encryption. The table also indicates whether those options are supported through the CMC or through HyperStore's S3 Service (S3 API) by using a third party S3 client (a client other than the CMC). Note that some options are supported only through the CMC (because they involve more than just S3 calls) and some options are supported only by using a third party S3 client to make calls to the HyperStore S3 API (because the CMC is a limited S3 client that does not support all S3 API calls and options).

Storage Policy Default	Bucket-Specific Default	Object-Specific
• SSE (CMC) • SSE-KMIP (CMC)	• SSE (S3 API) • AWS KMS (S3 API) • SSE-KMIP (CMC)	• SSE (CMC or S3 API) • AWS KMS (S3 API) • SSE-C (S3 API)

When the system is processing an object upload, the precedence ordering among these different configuration levels is usually that the more granular setting takes precedence over higher-level defaults. Put differently:

1. If a server-side encryption method is specified in the object upload request, the system uses that method. If not, then...
2. If a default server-side encryption method is specified in the configuration of the bucket to which the object is being uploaded, the system uses that method. If not, then...
3. If a default server-side encryption method is specified in the configuration of the storage policy used by the bucket to which the object is being uploaded, the system uses that method.

The exception to this typical precedence ordering is when SSE-KMIP is used. In particular:

- If SSE-KMIP is set as the storage policy default then all buckets that use that storage policy must use SSE-KMIP. Setting a different type of encryption at the bucket level is not allowed.
- If SSE-KMIP is set as the storage policy default or as the bucket default, then for an object specifying regular SSE in the upload request, SSE-KMIP will be used to encrypt the object rather than regular SSE. (By contrast an object level request for SSE-C does take precedence over a higher-level default of SSE-KMIP -- and so does an object level request for AWS KMS although this is an unlikely scenario since both AWS KMS and SSE-KMIP require some set-up by the system administrator and you would not typically set up both.)

For more information about using the supported server-side encryption methods, see:

- ["Using Regular SSE" \(page 188\)](#)
- ["Using SSE-KMIP" \(page 191\)](#)
- ["Using AWS KMS" \(page 195\)](#)
- ["Using SSE-C" \(page 198\)](#)

Note Regardless of the type of server-side encryption used, encryption is applied at the HyperStore "coordinator node" (the node that happens to receive the incoming S3 PUT request) before the object data is transmitted to the HyperStore endpoint nodes where the object data will be stored.

Server-Side Encryption Configuration Changes Do Not Apply Retroactively

Server-side encryption does not apply retroactively to objects that have already been uploaded to the system. For example, if you modify a bucket's configuration so that it includes server-side encryption, this will apply only to objects uploaded from that time forward -- not to objects that had been uploaded to the bucket previously. The same is true of adding server-side encryption to a storage policy's configuration: from that time forward, objects that get uploaded into buckets that use that storage policy will be encrypted, but objects that had already been uploaded previously will not be encrypted.

Conversely, if a bucket configuration or storage policy configuration uses server-side encryption and then you subsequently disable encryption for that bucket or storage policy, then from that time forward newly uploaded objects will not be encrypted -- but objects that had already been uploaded and encrypted prior to the configuration change will remain encrypted.

Server-Side Encryption and Auto-Tiering

How the HyperStore system handles auto-tiering of server-side encrypted objects depends on the encryption method used and the tiering destination.

Object's Encryption in Source HyperStore	Tiering to Amazon, Google, HyperStore, or Other Fully S3-Compliant Destinations	Tiering to Azure or Spectra BlackPearl
Regular SSE or SSE-KMIP	HyperStore decrypts the object, then tiers it to the destination system and includes an <code>x-amz-server-side-encryption: AES256</code> header in the upload request so that server-side encryption is implemented at the destination system. If the destination is a remote HyperStore system, the encryption applied there will be either regular SSE or SSE-KMIP depending on how the destination bucket and its storage policy are configured.	HyperStore decrypts the object, then tiers it to the destination system in decrypted form. These destinations do not support a server-side encryption request header in object uploads.
AWS KMS	HyperStore decrypts the object, then tiers it to the destination system and includes an <code>x-amz-server-side-encryption: aws:kms</code> header in the upload request so that AWS KMS based server-side encryption is implemented at the destination system. If the destination is a remote HyperStore system, AWS KMS based encryption will work only if the system is configured for AWS KMS integration.	
SSE-C	HyperStore tiers the encrypted object to the destination system, where it remains encrypted. To retrieve such a tiered object, client applications must first Restore the object into HyperStore, then GET the object (and include the SSE-C headers in the GET request). Streaming GETs of such tiered objects directly from the destination are not supported.	

Server-Side Encryption and Cross-Region Replication

When cross-region replication is configured for a source bucket:

- Objects encrypted by the regular SSE method **are** replicated to the destination bucket.
- Objects encrypted by the AWS KMS or SSE-C methods **are not** replicated to the destination bucket.

Note Cross-region replication is not supported for a source bucket that has SSE-KMIP configured. This applies to buckets that use a storage policy that has been configured for SSE-KMIP as well as buckets for which SSE-KMIP has been configured at the bucket level.

5.4.2.2. Using Regular SSE

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Preparing the Regular SSE Feature" \(page 189\)](#)

- ["Configuring Regular SSE as the Default for a Storage Policy \(CMC Only\)" \(page 189\)](#)
- ["Configuring Regular SSE as the Default for a Bucket \(S3 API Only\)" \(page 189\)](#)
- ["Requesting Regular SSE for Specific Objects \(CMC or S3 API\)" \(page 190\)](#)

With **regular server-side encryption (regular SSE)**, HyperStore:

- Maintains a master encryption key, obfuscated in the source code
- Generates a unique per-object encryption key for each new object and uses the per-object key to encrypt the object
- Uses the master key to encrypt each per-object key and stores the encrypted per-object key within the object's metadata

Regular server-side encryption (SSE) can be configured as the default encryption type for a storage policy, as the default encryption type for a particular bucket, or as the encryption type for particular objects. For information about the precedence ordering among these configuration levels see ["Server-Side Encryption" \(page 186\)](#).

Preparing the Regular SSE Feature

With regular SSE, the HyperStore system generates the per-object encryption keys using AES-128 by default. While AES-128 will work for regular SSE, and may be acceptable in a testing or evaluation environment, for greater security Cloudian recommends using AES-256.

To enable AES-256 in your HyperStore system, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set s3.sse.aes256=true
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

Note Amazon uses AES-256 for its regular SSE implementation, and AES-256 is called for in Amazon's SSE specification.

Configuring Regular SSE as the Default for a Storage Policy (CMC Only)

When you use the CMC to create or edit storage policies, one of the configurable policy attributes is server-side encryption. This sets a default server-side encryption type for all objects uploaded to all buckets that use the storage policy. To configure a storage policy to use regular SSE by default, in the "Server-Side Encryption" field of the storage policy configuration interface select "SSE".

For more information on storage policy creation and configuration see:

- ["Add a Storage Policy" \(page 513\)](#)
- ["Edit a Storage Policy" \(page 542\)](#)

Configuring Regular SSE as the Default for a Bucket (S3 API Only)

HyperStore's S3 API supports setting a default server-side encryption type for a bucket. This sets a default server-side encryption type for all objects uploaded to the bucket. The S3 API operations that set and manage a bucket default server-side encryption method are:

- PutBucketEncryption
- GetBucketEncryption

- DeleteBucketEncryption

To configure a bucket to apply regular SSE to uploaded objects by default:

- In the *PutBucketEncryption* request body set the *SSEAlgorithm* element to *AES256*.
- In the bucket's storage policy the server-side encryption configuration must be set to either "None" or "SSE". If in the *PutBucketEncryption* request the *SSEAlgorithm* element is set to *AES256* and the bucket's storage policy is configured to use "KMIP" for server-side encryption, then [SSE-KIMP](#) will be used as the bucket's default server-side encryption method rather than regular SSE.

For details of HyperStore's support for these S3 operations, see the S3 section of the *Cloudian HyperStore AWS APIs Support Reference*.

Note that:

- If you have not [enabled AES-256](#) in your HyperStore system, the system will use AES-128 to generate per-object encryption keys even though the *SSEAlgorithm* element specifies *AES256*.
- The CMC does not support setting a bucket default server-side encryption method.
- If a bucket's storage policy is configured to use regular SSE by default, there is no need to explicitly configure regular SSE as the bucket default encryption method. The bucket will inherit the SSE configuration from the storage policy.

Requesting Regular SSE for Specific Objects (CMC or S3 API)

Regular SSE can be requested for specific objects as those objects are uploaded to a bucket in the HyperStore system. This can be done either through the CMC or through a third party S3 client application that calls the HyperStore S3 API.

Note that:

- If regular SSE is configured as the default server-side encryption type for the bucket into which an object is being uploaded, or as the default server-side encryption type for the storage policy used by that bucket, there is no need to explicitly request regular SSE as the encryption type for that object.
- If [SSE-KMIP](#) is set as the storage policy default or as the bucket default, then for an object specifying regular SSE in the upload request, SSE-KMIP will be used to encrypt the object rather than regular SSE.

Object-Level SSE Through the CMC

In the CMC's **Objects** page, where a user can upload objects into the HyperStore system, the interface displays a "Store Encrypted" checkbox. If the user selects this checkbox, the HyperStore system applies regular server-side encryption to the uploaded object(s).

Object-Level SSE Through the S3 API

HyperStore's S3 API supports regular server-side encryption by the inclusion of the *x-amz-server-side-encryption: AES256* request header in any of these operations on objects:

- PutObject
- CreateMultipartUpload
- UploadPart
- POST Object
- CopyObject

For details of HyperStore's support for these S3 operations, see the S3 section of the *Cloudian HyperStore AWS APIs Support Reference*.

Note If you have not [enabled AES-256](#) in your HyperStore system, the system will use AES-128 encryption even though the `x-amz-server-side-encryption` request header specifies `AES256`.

5.4.2.3. Using SSE-KMIP

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Preparing the SSE-KMIP Feature" \(page 192\)](#)
- ["Configuring SSE-KMIP as the Default for a Storage Policy \(CMC Only\)" \(page 193\)](#)
- ["Configuring SSE-KMIP as the Default for a Bucket \(CMC Only\)" \(page 194\)](#)
- ["Re-Keying Buckets" \(page 194\)](#)
- ["Bucket Deletion" \(page 195\)](#)

With **SSE-KMIP** -- server-side encryption utilizing a Key Management System (KMS) that supports the Key Management Interoperability Protocol (KMIP) -- HyperStore triggers in the KMS the creation of one "master key" for each bucket that uses SSE-KMIP. Each bucket's master key is stored exclusively on the KMS, while in HyperStore a corresponding master key ID is stored in the bucket's metadata. When an object is uploaded to a bucket that uses SSE-KMIP, HyperStore generates a unique plain text data key and uses that data key to encrypt the object data. HyperStore then submits the plain text data key and the bucket's master key ID to the KMS, which uses the bucket master key to encrypt the data key and returns the encrypted data key back to HyperStore. HyperStore stores the encrypted data key in the object's metadata and discards the plain text data key.

Subsequently when a client downloads the object, HyperStore submits the encrypted data key and the bucket's master key ID to the KMS, which returns back the decrypted data key. HyperStore uses this plain text data key to decrypt the object and then deletes the plain text data key from memory.

SSE-KMIP can be configured at the storage policy level or at the per-bucket level. It cannot be configured at the per-object level. For information about the precedence ordering among these levels see ["Server-Side Encryption" \(page 186\)](#).

Note that:

- In communicating with a KMS, HyperStore supports **KMIP version 2.1 and earlier**.
- The process described above for encrypting and decrypting per-object data keys is used when the KMS supports the KMIP *encrypt* and *decrypt* commands, as is typically the case. In the atypical case when the KMS does not support these commands (an example is older versions of HashiCorp Vault -- versions 1.10 and earlier), HyperStore will retrieve the bucket's master key from the KMS each time an object is uploaded or downloaded. HyperStore will use the bucket master key to perform the encryption or decryption of the object's data key and then delete the bucket master key from memory.
- When a new bucket is created that uses a storage policy that is configured for SSE-KMIP, HyperStore automatically has the KMS create a master key for the new bucket. However, **if you plan to add SSE-KMIP to an existing storage policy's configuration** and there are already existing buckets that use that storage policy, then **before** you enable SSE-KMIP on the storage policy you must manually initiate a HyperStore operation that will direct the KMS to create master keys for those buckets. Details are in the ["Preparing the SSE-KMIP Feature" \(page 192\)](#) section below.

- Although for simplicity the terms "encrypt" and "decrypt" are used throughout the SSE-KMIP introduction above, technically what is done in regard to the per-object data keys is NIST-compliant "wrapping" and "unwrapping".

Preparing the SSE-KMIP Feature

This section describes the following SSE-KMIP preparation topics:

- ["Enabling AES-256 for Object Data Encryption"](#) (page 192)
- ["Creating a KMIP Profile for Each Target KMS"](#) (page 192)
- ["Defining a Command-Restricted Role for HyperStore on Each Target KMS"](#) (page 193)

Enabling AES-256 for Object Data Encryption

With SSE-KMIP, in the process whereby HyperStore generates a unique plain text data key and uses that data key to encrypt the object data, HyperStore by default uses AES-128. While AES-128 may be acceptable in a testing or evaluation environment, for greater security Cloudian recommends using AES-256.

To enable AES-256 in your HyperStore system, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.sse.aes256=true
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

Note When the KMS encrypts the object data keys supplied by HyperStore, the KMS will use AES-256 regardless of your HyperStore configuration.

Creating a KMIP Profile for Each Target KMS

A KMIP profile identifies a specific KMS that supports KMIP, and encapsulates the information needed for HyperStore to connect to that KMS. A KMIP profile must be created in the CMC for each KMS that HyperStore will utilize for server-side encryption. Then, a KMIP profile can be referenced by name when KMIP-based server-side encryption is configured for a storage policy or for an individual bucket.

You can control which of the following CMC user types are allowed to create KMIP profiles:

- System administrators
- Group administrators
- Regular users

By default, only system administrators can create KMIP profiles. To change this -- for example in a Managed Service Provider context where you want to allow each user group to specify and connect to a KMS of their choosing -- use the system configuration setting **"s3.kmip.manageProfile.grantees"** (page 822).

The usage scope of a KMIP profile varies depending on which type of CMC login role created the profile:

- KMIP profiles created by a **system administrator** are available for the purpose of setting KMIP encryption as a storage policy attribute, and are also available to all users in all user groups for the purpose of setting KMIP encryption for buckets for which KMIP is not already configured as part of the buckets' storage policy.

- KMIP profiles created by a **group administrator** are available to all users in that group, for the purpose of setting KMIP encryption for buckets for which KMIP is not already configured as part of the buckets' storage policy.
- KMIP profiles created by a **regular user** are available only to that user, for the purpose of setting KMIP encryption for buckets for which KMIP is not already configured as part of the buckets' storage policy.

As a system administrator you can create one or more KMIP profiles using the CMC's ["KMIP Profiles"](#) (page 545) page.

If you allow group administrators and/or regular users to also create KMIP profiles, the CMC will display for them a KMIP profile creation page at **Server-Side Encryption -> KMIP Profiles**.

Note If you had been using KMIP while running HyperStore versions older than 8.2 -- back when only one KMS was allowed -- that existing KMS connection information is automatically encapsulated into a KMIP "profile" by the HyperStore 8.2 upgrade process, and is given the KMIP profile name "default".

Defining a Command-Restricted Role for HyperStore on Each Target KMS

This is optional but recommended. As a security best practice, the KMS administrator should be asked to define a role for HyperStore that grants HyperStore permission to perform only the following KMIP operations (these are the only KMIP operations that HyperStore requires):

- DiscoverVersions
- Query
- Create
- Activate
- Get
- GetAttributes
- Locate
- Revoke
- Rekey
- Encrypt
- Decrypt

Configuring SSE-KMIP as the Default for a Storage Policy (CMC Only)

When you use the CMC to create or edit storage policies, one of the configurable policy attributes is server-side encryption. This sets a default server-side encryption type for all objects uploaded to all buckets that use the storage policy. This applies only to new buckets that are created **after** you configure server-side encryption as an attribute of the storage policy.

To configure a storage policy to use SSE-KMIP by default, in the "Server-Side Encryption" field of the storage policy configuration interface select "KMIP". When you do so, an additional field will display where you must select a [KMIP profile](#) (from a list of KMIP profiles that you have created). The KMIP profile indicates the KMS that HyperStore should connect to, and how to connect.

For more information on storage policy configuration see:

- ["Add a Storage Policy"](#) (page 513)
- ["Edit a Storage Policy"](#) (page 542)

For information about how server-side encryption configured at the storage policy level interacts with settings at the bucket level and object level, see ["Server-Side Encryption" \(page 186\)](#).

Note If you add KMIP encryption to an existing storage policy that is already being used by existing buckets, this in itself will not apply KMIP encryption to the existing buckets (instead it applies only from that time forward, to newly created buckets that use the storage policy). If you want to apply KMIP encryption to the existing buckets, this can be done on a per-bucket basis, as described in the next section below.

Configuring SSE-KMIP as the Default for a Bucket (CMC Only)

Bucket owners can set SSE-KMIP encryption as the default encryption behavior for a bucket by using the CMC's **Bucket Properties** interface (specifically the **KMIP** tab in that interface). Once this is done, objects subsequently uploaded to the bucket will be KMIP encrypted. It does not affect objects that were already in the bucket, if any.

Note that:

- Once a bucket has been configured for KMIP encryption, using a particular KMIP profile, **the KMIP profile for the bucket cannot be changed**. This includes the scenario where a bucket has inherited KMIP encryption from the storage policy that it uses. The bucket cannot be changed from using the KMIP profile associated with that storage policy to using a different KMIP profile.
- Once a bucket has been configured for KMIP encryption, KMIP encryption cannot be disabled for the bucket through the CMC. The only way to disabled KMIP encryption for a bucket is to use a third party S3 application to submit the S3 *DeleteBucketEncryption* call.
- As an administrator you may or may not be able to enable KMIP on particular buckets, on behalf of the bucket owners, depending on how you have configured the ["cmc.ui.admin.manageUsers.view.objectDataGrantees" \(page 739\)](#) setting.

Re-Keying Buckets

Once a bucket has been using SSE-KMIP, you may wish to periodically generate a new master key for the bucket. This is known as "re-keying" the bucket. The bucket's new master key -- which will be created by and stored on the KMS along with the bucket's older master key(s) -- will apply only to the encryption of HyperStore-supplied data keys for objects uploaded **after** the re-keying occurs (and decryption of those data keys if those objects are subsequently downloaded). The older bucket master key(s) continue to be used for decrypting data keys for objects that existed in the bucket prior to the re-keying.

IMPORTANT ! Some KMS products -- for example Fortanix -- allow you to rotate keys (re-key) on the KMS console. **Do not rotate HyperStore bucket keys directly on the KMS**. This will result in the KMS and HyperStore being out of sync in regard to which master key a bucket is currently using. Instead, follow the procedure below to re-key a bucket.

You can re-key a bucket or buckets **on demand** by logging into any HyperStore node and running whichever of the following commands best suits your purpose.

Re-key a specified bucket:

```
curl -d "" 'http://localhost/.system/rekey?bucketName=<bucketname>'
```

Re-key all buckets that use a specified storage policy:

```
curl -d "" 'http://localhost/.system/rekey?storagePolicy=<storagePolicyId>'
```

where `<storagePolicyId>` is the system-generated unique ID for the storage policy (which you can get from the CMC's [Storage Policies](#) page or from the Admin API call `GET /bppolicy/listpolicy`).

Re-key all buckets that use KMIP encryption:

```
curl -d "" 'http://localhost/.system/rekey'
```

Note that in a multi-region HyperStore system:

- The commands for re-keying a specified bucket or a specified storage policy must be run from a HyperStore node within the region where the bucket or storage policy is based.
- The command for re-keying all buckets that use KMIP encryption must be run from one node in each service region. For example if you have two regions "Region1" and "Region2", run the command from a node in Region1 and then run the command from a node in Region2.

If you wish you can configure a **cron job** to regularly re-key all buckets that use KMIP encryption. To do so, log into the Config Controller node and run these `hsctl` commands:

```
# hsctl config set jobs.system.kmipRekey.enabled=true
# hsctl config apply jobs
```

By default the cron job will run once per month (on the first day of each calendar month). To change the scheduling of the cron job, on the Config Controller node run the following commands:

```
# hsctl config set jobs.system.kmipRekey.schedule=<string>
# hsctl config apply jobs
```

The `<string>` is specified in standard crontab schedule formatting, enclosed in quotes; for example the default schedule is "0 0 1 * *".

For more information cron jobs see ["System cron Jobs"](#) (page 632)

Note Do not have two processes re-keying the same bucket at the same time. For example, do not manually run re-key commands at the same time as a re-key cron job is running; and be sure not to have two different system administrators running re-key commands for the same bucket at the same time. This could result in the KMS and HyperStore being out of sync in regard to which master key a bucket is currently using.

Bucket Deletion

When a bucket that has been using KMIP encryption is deleted in HyperStore, HyperStore sends to the KMS a command to revoke the bucket's master key. If the bucket has had multiple master keys as a result of re-keying, HyperStore directs the KMS to revoke all of the bucket's master keys.

If you wish to permanently destroy the revoked keys, it must be done on the KMS. Check your KMS documentation for information about destroying revoked keys.

5.4.2.4. Using AWS KMS

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Preparing the AWS KMS Encryption Feature"](#) (page 196)
- ["Configuring AWS KMS Encryption as the Default for a Bucket \(S3 API Only\)"](#) (page 197)

- ["Requesting AWS KMS Encryption for Specific Objects \(S3 API Only\)" \(page 197\)](#)
- ["Deleting a Bucket That Uses AWS KMS Encryption" \(page 198\)](#)

With **AWS KMS** encryption -- server-side encryption utilizing the Amazon Web Services Key Management Service -- HyperStore triggers in the AWS KMS the creation of one "customer master key" (CMK) per bucket. The CMK is stored exclusively in the AWS KMS. For each such CMK, HyperStore stores (in the Credentials DB) a CMK ID that allows HyperStore to tell AWS KMS which CMK to use for creating a unique encrypted "data key" for a given object (that is, the CMK for the bucket into which the object is being uploaded). AWS KMS returns to HyperStore both the encrypted data key and plain text version of the data key. After using the plain text data key to encrypt the object, HyperStore discards the plain text data key and stores the encrypted version of the data key within the object's metadata.

Subsequently when a client downloads the object, HyperStore submits the encrypted data key to the AWS KMS, which uses the CMK to decrypt the data key. AWS KMS returns the plain text data key to HyperStore, which uses it to decrypt the object and then deletes the plain text data key from memory.

AWS KMS based encryption can be configured as the default encryption type for a particular bucket or as the encryption type for particular objects. Setting the AWS KMS method as the default for a storage policy is not supported. For information about the precedence ordering between these configuration levels see ["Server-Side Encryption" \(page 186\)](#).

Note In compliance with Amazon's implementation, all S3 requests involving AWS KMS encryption must use SSL and Signature Version 4. For example HyperStore will reject object upload requests that specify AWS KMS encryption, or download requests for AWS KMS encrypted objects, if such requests use Signature Version 2.

Note For more information on the AWS KMS, in the AWS online documentation see [AWS Key Management Service \(KMS\)](#).

Preparing the AWS KMS Encryption Feature

To use the AWS KMS for HyperStore server-side encryption, you must have either or a combination of:

- **AWS account access credentials for each HyperStore user group** that will use the AWS KMS encryption feature. These group account credentials will be used by HyperStore to access the AWS KMS whenever a user within the group requests AWS KMS encryption for their bucket or for specific objects, or downloads AWS KMS encrypted objects.
- **Default AWS account access credentials** for your HyperStore service as a whole. These default AWS credentials will be used by HyperStore to access the AWS KMS on behalf of users who are in groups that do not have group account credentials for AWS.

To enable AWS KMS usage in your HyperStore system, complete the following system configuration steps:

1. On the Config Controller node, open the following file in a text editor:

```
/etc/cloudian/aws/credentials
```

2. Edit the file to specify the system default AWS access credentials, and (optionally) any group-specific AWS access credentials. Create a separate block for each group that has its own AWS access credentials, using the formatting shown in the example below. In the example there are credentials for just one group, "CloudianTest1". HyperStore will use the CloudianTest1 group's AWS credentials when

accessing the AWS KMS on behalf of users in that group. For users in any other group, HyperStore will use the system default AWS credentials (the "[default]" block) when accessing the AWS KMS.

```
[default]
aws_access_key_id = AKIAJKVELYABCCEIXXMA
aws_secret_access_key = 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

[CloudianTest1]
aws_access_key_id = ABCAJKVELY6YXCEIMAXX
aws_secret_access_key = abceikWvRR/7A8916x9vUDEhV+C+LIDmFCOE8MgC
```

Save your change and close the file.

3. Still on the Config Controller node, run this command to specify the AWS service region of the AWS KMS that you want HyperStore to use:

```
hsctl config set s3.sse.kms.region=<string>
```

For example:

```
hsctl config set s3.sse.kms.region=us-east-1
```

4. Still on the Config Controller node, run these commands to apply your changes:

```
hsctl config apply s3
hsctl service restart s3 --nodes=ALL
```

Configuring AWS KMS Encryption as the Default for a Bucket (S3 API Only)

HyperStore's S3 API supports setting a default server-side encryption type for a bucket. This sets a default server-side encryption type for all objects uploaded to the bucket. The S3 API operations that set and manage a bucket default server-side encryption method are:

- PutBucketEncryption
- GetBucketEncryption
- DeleteBucketEncryption

To configure a bucket for server-side encryption with AWS KMS, in the *PutBucketEncryption* request body set the *SSEAlgorithm* element to *aws:kms*.

For details of HyperStore's support for these S3 API operations, see the S3 section of the **Cloudian HyperStore AWS APIs Support Reference**.

Note The CMC does not support setting a bucket default server-side encryption method.

Requesting AWS KMS Encryption for Specific Objects (S3 API Only)

HyperStore's S3 API supports AWS KMS based server-side encryption by the inclusion of the *x-amz-server-side-encryption: aws:kms* request header in any of these operations on objects:

- PutObject
- CreateMultipartUpload
- UploadPart
- POST Object
- CopyObject

For details about HyperStore support of these operations see the S3 section in the *Cloudian HyperStore AWS APIs Support Reference*.

Note that:

- If AWS KMS is configured as the default server-side encryption type for the bucket into which an object is being uploaded there is no need to explicitly request AWS KMS as the encryption type for that object.
- The HyperStore S3 Service does not support the `x-amz-server-side-encryption-aws-kms-key-id` or `x-amz-server-side-encryption-context` request headers.
- The CMC does not support requesting AWS KMS based encryption for objects as they are uploaded. It **does** support downloading objects that have been encrypted by the AWS KMS method.

Deleting a Bucket That Uses AWS KMS Encryption

When you delete a HyperStore bucket that has used AWS KMS encryption -- either because AWS KMS encryption was the default encryption method for the bucket, or because certain objects within the bucket used AWS KMS encryption -- the bucket's "customer master key" (CMK) is scheduled for deletion from the remote AWS KMS system. The CMK deletion occurs 30 days after the deletion of the HyperStore bucket. During this 30 day period, if you do not wish the CMK to be deleted from the remote AWS KMS system you can execute a *CancelKeyDeletion* operation using the AWS Console or the AWS CLI.

5.4.2.5. Using SSE-C

Subjects covered in this section:

- *Introduction (immediately below)*
- *["Preparing the SSE-C Feature" \(page 198\)](#)*
- *["Requesting SSE-C for Specific Objects \(S3 API Only\)" \(page 199\)](#)*

When **server-side encryption with customer-provided encryption keys (SSE-C)** is used, the HyperStore system does not store the customer-provided encryption key itself but rather stores a hash of the key (for purposes of verifying the key if it's subsequently submitted in a GET Object request). The key hash is stored with the object metadata.

SSE-C based encryption can only be set at the per-object level. Setting SSE-C as the default encryption method for a bucket or a storage policy is not supported.

IMPORTANT ! When SSE-C is used, **the user is responsible for managing the encryption key**. If an object is uploaded to HyperStore system and encrypted with a user-provided key, the user will need to provide that same key when later requesting to download the object. **If the user loses the key, the encrypted object will not be downloadable**. This is consistent with Amazon's implementation of the SSE-C feature. For more information on Amazon's SSE-C feature see [Protecting Data Using Server-Side Encryption with Customer-Provided Encryption Keys \(SSE-C\)](#).

Preparing the SSE-C Feature

Before using SSE-C you must first:

- Enable AES-256 in your HyperStore system. Like Amazon S3, HyperStore's implementation of SSE-C requires AES-256 encryption of object data. To enable AES-256 in your HyperStore system, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set s3.sse.aes256=true
```

```
# hctl config apply s3
# hctl service restart s3 --nodes=ALL
```

- Set up HTTPS for your S3 Service. Like Amazon S3, HyperStore's implementation of SSE-C requires that the relevant S3 API requests be transmitted over HTTPS rather than regular HTTP. For instructions see ["Security and Privacy Features"](#) (page 177).

Note HyperStore supports a configuration for allowing a regular HTTP connection between a load balancer and your S3 servers for transmission of SSE-C requests over your internal network, while client applications use HTTPS in the requests that come into the load balancer. See the configuration setting ["s3.sse.tls.required"](#) (page 831).

Requesting SSE-C for Specific Objects (S3 API Only)

HyperStore's S3 API supports server-side encryption with user-provided keys by the inclusion of the *x-amz-server-side-encryption-customer-** request headers in any of these operations on objects:

- PutObject
- CreateMultipartUpload
- UploadPart
- POST Object
- CopyObject (supporting also the *x-amz-copy-source-server-side-encryption-customer-** request headers)
- GetObject
- HeadObject

For the full list of supported *x-amz-server-side-encryption-customer-** request headers for each of these operations, see the S3 section of the [Cloudian HyperStore AWS APIs Support Reference](#).

Note The CMC does not support requesting SSE-C for objects as they are uploaded, and it does not support downloading objects that have been encrypted by the SSE-C method.

5.4.3. Setting Up User Password and Credential Controls

Subjects covered in this section:

- ["CMC Passwords"](#) (page 199)
- ["CMC Multi-Factor Authentication"](#) (page 201)
- ["S3 Security Credentials"](#) (page 201)
- ["Allowing Administrator Access to User's Buckets Through the CMC"](#) (page 202)

5.4.3.1. CMC Passwords

System administrators can use the Cloudian Management Console (CMC) to monitor and administer the HyperStore system, and to provision and manage service users and user groups. Group administrators can use the CMC to provision and manage service users within their group, and to use the S3 storage service (creating and configuring buckets and uploading and downloading objects). Regular users can use the CMC to manage their own user accounts and to use the S3 storage service.

All types of users require a password to access the CMC. For the pre-configured system admin user named *admin*, the default password is *public* and you are required to change the password the first time you log into the CMC as the admin user (see [Accessing the Cloudian Management Console](#)). For all other users, when you create new users in the CMC -- additional system admin users, or group admins, or regular users -- you specify an initial CMC password for each user. Users can subsequently change their own password through the CMC's **"Security Credentials"** (page 572) page. And as a system admin you can change other users' passwords through the CMC's **"Manage Users"** (page 400) page.

Note The Admin API also supports provisioning users and creating user passwords. For more information see the "user" section of the *Cloudian HyperStore Admin API Reference*.

As a system administrator you can configure several types of requirements for CMC passwords, as shown in the table below. Note that some of these configurable requirements are disabled by default.

Password Restriction	Configuration Setting	Default
Minimum number of characters in password	"common.user.password.minLength" (page 757)	9
Number of days of password lifespan before the password expires and a new password must be created	"common.user.password.rotation.expiration" (page 758)	Requirement disabled
Maximum allowed character redundancy between old password and new password	"common.user.password.dupCharRatioLimit" (page 755)	Requirement disabled
Minimum number of new passwords before an old password can be reused	"common.user.password.uniqueGenerations" (page 758)	Requirement disabled
Minimum days between password changes	"common.user.password.rotation.gracePeriod" (page 758)	Requirement disabled
Whether or not to lock out a user who submits an incorrect password or passwords multiple times	"common.user.password.lock.enabled" (page 756)	Enabled if your original HyperStore version was 7.3 or newer Disabled if your original HyperStore version was older than 7.3
Time interval of password lock-out in number of seconds (if lock-out feature is enabled)	"common.user.password.lock.durationSec" (page 756)	1800
Number of failed attempts that triggers lock-out (if lock-out feature is enabled)	"common.user.password.lock.maxFailedAttempts" (page 757)	6

5.4.3.2. CMC Multi-Factor Authentication

The CMC supports multi-factor authentication (MFA), whereby a user is required to provide a time-based MFA code in order to log into the CMC (in addition to providing their user name and password). By default MFA is disabled for each user. Each user -- whether a system admin, a group admin, or a regular user -- has the option to enable MFA for their own CMC account. Users can enable MFA in the CMC's **Security Credentials** page (the same page in which they manage their CMC password).

For more information about how a user can enable MFA on their CMC account, see ["Security Credentials" \(page 572\)](#).

Optionally you can configure the system so that the CMC **requires** all users (including system administrators) to set up and use MFA for their CMC logins. To do so, set **"cmc.mfa.enforced"** (page 723) to *true* (by default it is *false*).

As a system administrator, you can disable MFA on the account of a user who currently has MFA enabled. You might need to do this if, for example, an MFA-enabled user is unable to log into the CMC because of a problem with their virtual MFA device (MFA application on their computer). If you have configured the system to require MFA for CMC logins, and you disable MFA for a particular user, the user will need to set up MFA again the next time they try to log into the CMC.

For more information about disabling MFA on a CMC user's account, see ["Disable MFA for a User" \(page 412\)](#).

5.4.3.3. S3 Security Credentials

To use the HyperStore S3 Service -- the service through which storage buckets are created and configured and objects are uploaded and downloaded -- users must have S3 security credentials (a paired "access key" and "secret key"). These credentials are distinct from the CMC password, and are used specifically to interact with the HyperStore storage layer through the S3 API. When you create a new user in the CMC, the system automatically creates S3 security credentials for the user. Users can subsequently view or manage their S3 security credentials through the CMC's **"Security Credentials" (page 572)** page. And as a system admin you view or manage other users' S3 security credentials through the CMC's **"Manage Users" (page 400)** page.

Note The Admin API also supports provisioning users and creating users' S3 security credentials. For more information see the "user" section of the *Cloudian HyperStore Admin API Reference*.

If a user employs the CMC as their S3 client application, the CMC automatically includes the user's S3 security credentials when the CMC submits calls to the S3 Service. If a user employs a third party S3 client application to access the HyperStore S3 Service, the user will need to supply their S3 security credentials to the third party S3 client application (so the client application can include the credentials when submitting calls to the HyperStore S3 Service). The user can retrieve their S3 security credentials from the CMC's **Security Credentials** page, so that they can then supply the credentials to the third party S3 client application.

Users are allowed to have multiple S3 credentials (multiple access key / secret key pairs). They can create and delete credentials -- and make credentials active or inactive -- in the CMC's **Security Credentials** page. By default the system allows users to have a maximum of five total S3 security credentials (including inactive credentials as well as active credentials). This restriction is configurable by the **"s3.user.maxCredentials"** (page 841) setting.

5.4.3.4. Allowing Administrator Access to User's Buckets Through the CMC

By default administrators cannot view or manage user's buckets and objects through the CMC. To change this and allow such access, use the configuration setting "[cmc.ui.admin.manageUsers.view.objectDataGrantees](#)" (page 739).

5.4.4. Enabling Secure Delete

IMPORTANT ! Using secure delete impacts system performance for delete operations. Consult with your Cloudbian representative if you are considering using secure delete.

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Enabling Secure Delete" \(page 203\)](#)
- ["Secure Delete Logging" \(page 203\)](#)

HyperStore supports a "secure delete" methodology for implementing object delete requests. **By default this feature is disabled.**

For background, note that object data is stored in an `ext4` file system on each HyperStore node, and the process of writing, reading, and deleting object data from the file system is managed by the HyperStore Service (for more information see ["HyperStore Service and the HSFS" \(page 66\)](#)). Also note that, depending on your storage policies, each object is either replicated or erasure coded, and the replicas or the erasure coded fragments are distributed across multiple nodes. Larger objects are broken into chunks first, before those chunks are then replicated or erasure coded.

When secure delete is enabled HyperStore implements the deletion of an object by first overwriting each byte of the object data three times, and then deleting the object. The three passes at overwriting each of the object's bytes are executed as:

- 1st pass: byte is overwritten as 00110101 (0x35)
- 2nd pass: byte is overwritten as 11001010 (0xCA)
- 3rd pass: byte is overwritten as 10010111 (0x97)

This overwriting occurs for **every byte of every replica or fragment of the object, on every node on which the object's data resides**. After the three overwriting passes complete, the object data is then deleted.

If you enable secure delete, then all deletes -- for any bucket, by any user -- are implemented as secure deletes. You cannot, for instance, apply this feature only to some buckets and not to others.

Note

* In the case of buckets that use versioning, to delete all versions of an object the S3 client application must explicitly delete each object version.

* Secure delete does not apply to object metadata stored in the Metadata DB (in Cassandra). When objects are deleted, the system deletes the corresponding object metadata in the normal way -- with no overwriting passes -- regardless of whether or not you have the secure delete feature turned on. Therefore you should limit any user-defined object metadata created by your S3 client applications to information that does not require secure delete.

5.4.4.1. Enabling Secure Delete

To enable the secure delete feature, set **"hyperstore.secureDelete"** (page 778) to *true* (by default it is *false*). (For instructions on changing this setting and applying your change see the setting description.)

5.4.4.2. Secure Delete Logging

Secure delete activity is logged in *cloudian-hyperstore-request-info.log* on each node. The activity is logged only after completion of the third and final overwrite pass. The log entry indicates SECURE-DELETE as the operation type, and a 200 status code in the log entry indicates that the secure delete was successful. The log entry also includes the object name and the corresponding *ext4* file name.

For more information about *cloudian-hyperstore-request-info.log* see **"HyperStore Service Logs"** (page 671).

5.4.5. FIPS Support

Federal Information Processing Standard (FIPS) [Publication 140-2](#) defines security requirements for cryptographic modules. HyperStore in some respects meets the FIPS 140-2 standard by default. For, HyperStore by default uses FIPS 140-2 prescribed cryptographic techniques for SHA-256 (used to hash request payloads for some types of S3 requests) and for HMAC (used for Signature version 4 validation of S3 requests).

However, if you need strict FIPS compliance you must enable full HyperStore FIPS compliance by running these *hsctl* commands on the Config Controller node:

```
# hsctl config set common.fips.enabled=true
# hsctl config set common.fips.tls.enabled=true
# hsctl config set common.tls.ciphers.profile=fips-140-2-compatible
# hsctl config apply all
# hsctl service restart all --nodes=ALL
```

This puts HyperStore into FIPS compliant mode for its role in [server-side encryption](#) implementation, and for TLS implementation (for the S3 Service and other user-facing services).

Note Setting *common.fips.enabled* to *true* will work only if the *ssh.daemon.disableConfigUpdate* setting is at its default value of *false*. You can confirm this with this command:

```
hsctl config get ssh.daemon.disableConfigUpdate
```

5.5. Auto-Tiering

5.5.1. Auto-Tiering Feature Overview

Subjects covered in this section:

- *Introduction (immediately below)*
- **"Auto-Tiering Versus Cross-Region Replication"** (page 205)
- **"Tiering Destination Accounts, Credentials, and Buckets"** (page 205)
- **"Bridge Mode (Proxy Tiering)"** (page 206)

- ["Option to Retain a Local Copy" \(page 207\)](#)
- ["Bi-Directional Auto-Tiering" \(page 207\)](#)
- ["Auto-Tiering Logging" \(page 207\)](#)
- ["Auto-Tiering Licensing" \(page 207\)](#)
- ["Auto-Tiering of Encrypted Objects" \(page 207\)](#)
- ["Auto-Tiering and User-Defined Object Metadata" \(page 208\)](#)
- ["How Auto-Tiering Impacts Usage Tracking, QoS, and Billing" \(page 208\)](#)
- ["Tiering Statistics for Spectra BlackPearl" \(page 208\)](#)

The HyperStore system supports an "auto-tiering" feature whereby objects can be automatically moved from local HyperStore storage to a remote storage system on a defined schedule. HyperStore supports auto-tiering from a local HyperStore bucket to any of several types of destinations systems:

- S3-compliant systems: AWS S3, AWS Glacier (Glacier Flexible Retrieval), AWS Deep Archive (Glacier Deep Archive), Google Storage Cloud, a HyperStore region or system, or a different S3-compliant system of your choosing
- Microsoft Azure
- Spectra Logic BlackPearl

Auto-tiering is configurable on a per-bucket basis. A bucket owner activating auto-tiering on a bucket can specify:

- The auto-tiering destination system
- Whether auto-tiering applies to all objects in the bucket, or only to objects that match against configurable filters for object name prefix, object size, and/or object tags

Note The HyperStore S3 API supports auto-tiering filtering by prefix, size, and/or tags, but the CMC currently only supports filtering by prefix and/or size. The CMC does not support filtering by object tags.

- The tiering schedule, which can be implemented in any of these ways:
 - Move objects X number of days after they're created
 - Move objects if they go X number of days without being accessed
 - Move objects on a fixed date — such as December 31, 2024
 - Move objects immediately after they're created ("Bridge Mode", also known as proxy tiering). Note that with Bridge Mode, filtering by prefix or tags is not supported. Only object size based filtering is supported for Bridge Mode.

Although there can only be one auto-tiering destination for a given HyperStore bucket, bucket owners have the option of configuring different auto-tiering schedules for different sets of objects within the bucket, based on the object name prefix (unless Bridge Mode is being used, which does not support prefix filtering).

For more information about configuring auto-tiering in the system and on individual buckets, see ["Preparing the Auto-Tiering Feature" \(page 209\)](#) and ["Setting Up Auto-Tiering for a Bucket" \(page 210\)](#).

Note Auto-tiering restrictions based on destination type:

- * Tiering to Azure, Google, or Spectra BlackPearl is not supported for source buckets that have versioning enabled or that have had versioning enabled in the past.
- * For tiering to Google to work, the destination bucket in Google must be configured for "fine-grained"

access, not "uniform" access.

* 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. To change this limit, consult with Cloudian Support.

5.5.1.1. Auto-Tiering Versus Cross-Region Replication

The key differences between auto-tiering and [cross-region replication](#) are:

- With cross-region replication each replicated object is stored in both the source bucket and the destination bucket. With auto-tiering, after an object is auto-tiered the object data is by default stored only in the destination bucket (although there is an option to retain a local copy for a configurable period of time).
- Cross-region replication replicates objects to the destination bucket immediately after they've been uploaded to the source bucket. With auto-tiering, typically the tiering of the object occurs on a user-defined schedule, after the objects have been in the source bucket for a period of time (although there is an option to tier immediately).
- Cross-region replication replicates nearly all object metadata and user-defined tags along with the object data, so that an object's full set of metadata resides both at the source and at the destination. With auto-tiering only basic metadata accompanies the tiered object, while the object's full metadata continues to be stored locally at the source.
- Cross-region replication requires that versioning be enabled on both the source bucket and the destination bucket. Auto-tiering does not have this versioning requirement.
- With cross-region replication the destination bucket is typically within the same HyperStore system as the source bucket (although replicating to an external system is supported). With auto-tiering the destination bucket is typically in an external system (although tiering to a bucket within the same HyperStore system is supported).

You cannot use cross-region replication and auto-tiering on the same source bucket.

5.5.1.2. Tiering Destination Accounts, Credentials, and Buckets

Auto-tiering to AWS or another destination system requires that there be an existing account that the HyperStore system can access at the destination system. There are two options for account access:

- Bucket owners can supply their own destination account credentials, on a per-bucket basis. In this way each bucket owner tiers to his or her own account at the destination system. This is the default method for auto-tiering.
- You can supply system default tiering credentials. This is the appropriate approach if all users will be tiering to the same account at the same tiering destination system. For more information see "[Preparing the Auto-Tiering Feature](#)" (page 209).

HyperStore encrypts supplied tiering account credentials and stores them in the Credentials DB, where they can be accessed by the system in order to implement auto-tiering operations.

Auto-tiering moves objects from a local HyperStore source bucket into a **tiering bucket** at the destination system. The source bucket owner when configuring auto-tiering can specify as the tiering bucket a bucket that already exists in the destination system, or the source bucket owner can have HyperStore create a tiering bucket in the destination system. If having HyperStore create a tiering bucket, the source bucket owner can

choose a tiering bucket name or have HyperStore automatically name the tiering bucket. When HyperStore automatically names the tiering bucket it uses this format:

<origin-bucket-name-truncated-to-34-characters>-<random-string>

The HyperStore system appends a 28-character random string to the origin bucket name to ensure that the resulting destination bucket name is unique within the destination system. If the origin bucket name exceeds 34 characters, in the destination bucket name the origin bucket name segment will be truncated to 34 characters.

After objects have been auto-tiered to the destination system they can be accessed directly through that system's interfaces (such as the AWS S3 Console), by persons having the applicable credentials. Auto-tiered objects can also be accessed indirectly through the local HyperStore system interfaces. Tiered object access is described in more detail in ["Accessing Auto-Tiered Objects" \(page 211\)](#).

Note In the case of auto-tiering to AWS Glacier or AWS Deep Archive, the HyperStore system creates a bucket in AWS S3 and configures that remote bucket for transitioning to AWS Glacier or AWS Deep Archive (or if the remote tiering bucket already exists, HyperStore configures that existing bucket for transitioning to AWS Glacier or AWS Deep Archive). Objects are then auto-tiered from the HyperStore source bucket to the remote bucket in AWS S3, where they are subject to AWS's automated mechanism for transitioning objects to AWS Glacier or AWS Deep Archive. An object's transitioning to AWS Glacier or AWS Deep Archive will occur within 24 hours of the object being tiered from HyperStore to AWS S3 (since AWS executes its tiering process once a day).

5.5.1.3. Bridge Mode (Proxy Tiering)

The HyperStore auto-tiering feature supports a "Bridge Mode" (also known as proxy tiering) whereby objects can be transitioned to a destination system immediately after they are uploaded to the HyperStore source bucket. As soon as such objects are successfully transitioned to the destination system, by default they are flagged for deletion from the local HyperStore system (the actual deletion will be executed afterwards, by a cron job that runs hourly). After deletion of the local copy of the objects, only object metadata remains (see ["How Auto-Tiering Impacts Usage Tracking, QoS, and Billing" \(page 208\)](#) for more information about this metadata).

With Bridge Mode, filtering by prefix or tags is not supported. The only filter type supported for Bridge Mode is object size. Using the object size filter you could, for example, have newly uploaded large objects get immediately tiered to a different system that uses object storage media best suited for large objects (such as QLC SSDs) while smaller objects remain on a local system.

When the system implements Bridge Mode tiering, whichever S3 node processes the upload of a given object into the source bucket also initiates the immediate transmission of the object to the destination system. In this way, just as the workload for processing numerous incoming object uploads from S3 client applications is distributed across all the nodes in the cluster, so too the workload associated with Bridge Mode tiering is distributed across the cluster. If the initial attempt to transmit an object to the destination system fails with a temporary error, the same node that performed the initial attempt will retry once every hour until either the object is successfully transmitted or a permanent error is encountered (an example permanent error would be if someone had deleted the tiering destination bucket). The local copy of the object will not be deleted until the object is successfully transmitted to the destination system, as indicated by the destination system returning a success status. All attempts are logged as described in ["Auto-Tiering Logging" \(page 207\)](#).

Users have the option of choosing Bridge Mode when they configure auto-tiering rules for a bucket.

Note Bridge mode is not supported for tiering to AWS Glacier or AWS Deep Archive or Spectra Black-Pearl.

5.5.1.4. Option to Retain a Local Copy

By default the system deletes the local copy of an object as soon as the object is successfully transitioned to the tiering destination. However, HyperStore supports an option to retain a local copy of objects that have been auto-tiered, for a specified period of time. Locally retained objects will continue to be protected by the storage policy associated with the bucket in which the objects reside (whether replication or erasure coding). After the specified retention period the system will automatically delete the local copy.

Users have the option of specifying a local retention period when they configure auto-tiering rules for a bucket. This option to retain a local copy is supported for Bridge Mode (Proxy) tiering as well as for regular, schedule-based tiering.

5.5.1.5. Bi-Directional Auto-Tiering

Bi-directional auto-tiering between two buckets within the same HyperStore system is prohibited. So, for example, the system does not allow having HyperStore bucket1 tier to HyperStore bucket2 while HyperStore bucket2 tiers to HyperStore bucket1.

5.5.1.6. Auto-Tiering Logging

As the HyperStore system auto-tiers objects from local storage to the tiering destination, it records information about the tiering transactions (including source bucket name, object name, and destination bucket name) in the tiering request log [cloudian-tiering-request-info.log](#). For regular auto-tiering that occurs on a defined schedule, the auto-tiering processing associated with a given bucket -- and the logging of that auto-tiering processing -- is spread out across all the nodes that participate in the storage policy used by that bucket. For example, in a HyperStore system with two data centers, if a bucket uses a storage policy that stores data only in one of the two data centers, the processing workload of that bucket's auto-tiering will be spread among the nodes in that one data center -- and so will the logging entries for that bucket's auto-tiering activity. By contrast, for a bucket that uses a storage policy that stores data in both data centers, the processing workload (and log entries) associated with that bucket's auto-tiering will be spread among all nodes in both data centers.

In the special case of "Bridge Mode" auto-tiering, whichever S3 node processes the upload of a given object into the source bucket also initiates the immediate auto-tiering of the object to the destination system, and the tiering request log entry for that is written locally on that node.

5.5.1.7. Auto-Tiering Licensing

Your HyperStore license may impose a limit on how much tiered data you can have stored in third party external systems (destinations other than an external HyperStore system). For details see ["Licensed Maximum Tiered Storage Usage"](#) (page 39).

5.5.1.8. Auto-Tiering of Encrypted Objects

For information about how auto-tiering works together with the server-side encryption feature, see ["Server-Side Encryption and Auto-Tiering"](#) (page 188). As detailed in that section, encrypted objects may or may not be eligible for auto-tiering depending on the encryption method and the particular tiering destination.

5.5.1.9. Auto-Tiering and User-Defined Object Metadata

For information about how auto-tiering impacts user-defined object metadata, see ["Cross-Region Replication and Auto-Tiering of Object Metadata and Tags" \(page 233\)](#).

5.5.1.10. How Auto-Tiering Impacts Usage Tracking, QoS, and Billing

When an object is auto-tiered to a destination system and deleted from local storage the size of the tiered object is subtracted from the bucket owner's HyperStore [usage count](#) for Storage Bytes. At the same time, a reference to the auto-tiered object is created and the size of this reference — 8KB, regardless of the transitioned object size — is added to the Storage Bytes count. Meanwhile, auto-tiering does not impact the Storage Objects count. An object counts toward the number of Storage Objects regardless of whether or not it is auto-tiered.

For example, if a 1MB object has been auto-tiered to AWS then that object would count as:

- 8KB toward the bucket owner's HyperStore count for Storage Bytes.
- 1 toward the bucket owner's HyperStore count for Storage Objects.

If an auto-tiered object is temporarily restored to HyperStore storage, then while the object is restored the object's size is added back to the Storage Bytes count and the 8KB for the reference is subtracted from the count. After the restore interval ends and the restored object instance is automatically deleted, the object size is once again subtracted from Storage Bytes and the 8KB for the reference is added back. For more on temporary restoration of tiered objects see ["Accessing Auto-Tiered Objects" \(page 211\)](#).

Auto-tiering does not impact HyperStore usage counts for Bytes-IN or Bytes-OUT.

Note If users select the [Retain Local Copy](#) option when configuring their buckets for auto-tiering, objects that are temporarily retained in HyperStore after they've been auto-tiered will continue to count toward the local Storage Bytes count until the local copy is deleted.

HyperStore's Quality of Service (QoS) and billing features make use of Storage Bytes and Storage Object counts. So the impact of auto-tiering on QoS and billing is as described above. For example in the case of the QoS restrictions applied to users, if a bucket owner's 1MB object has been auto-tiered to AWS, then that object counts as 8KB toward that user's QoS limit for Storage Bytes; and as 1 toward that user's QoS limit for Storage Objects.

For more information on the QoS and billing features, see ["Quality of Service \(QoS\) Feature Overview" \(page 246\)](#) and ["Billing Feature Overview" \(page 258\)](#).

5.5.1.11. Tiering Statistics for Spectra BlackPearl

If your HyperStore system is using auto-tiering to Spectra BlackPearl, you can run the following command on the HyperStore cron job master node to retrieve tiering statistics. This command is supported only for Spectra BlackPearl tiering.

```
# curl http://localhost:80/.system/stats/tiering/spectra
```

For example:

```
# curl http://localhost:80/.system/stats/tiering/spectra
{"totalInQueue":0,"totalTiered":51,"tieringFail":0,"restored":1,"restoreFail":0,
"bytesTiered":629145600,"bytesRestored":1073741824}
```

The returned statistics are:

- Number of objects in queue waiting to be tiered to Spectra BlackPearl
- Number of objects tiered to Spectra BlackPearl
- Number of objects for which tiering to Spectra BlackPearl failed
- Number of tiered objects restored to HyperStore from Spectra BlackPearl
- Number of objects for which restoring to HyperStore from Spectra BlackPearl failed
- Total number of bytes tiered to Spectra BlackPearl
- Total number of bytes restored to HyperStore from Spectra BlackPearl

Note If you're not sure which node is your cron job node, you can check this in the CMC's [Cluster Information](#) page.

5.5.2. Preparing the Auto-Tiering Feature

This section covers the following auto-tiering system set-up topics:

- ["Enable and Configure the Auto-Tiering Feature in the CMC"](#) (page 209)
- ["Configure Tiering Destinations"](#) (page 210)

5.5.2.1. Enable and Configure the Auto-Tiering Feature in the CMC

Note The configuration task described below is applicable only to using the auto-tiering feature through the CMC. It is not applicable to using a third party S3 client application to invoke the HyperStore S3 Service's auto-tiering feature.

By default auto-tiering functionality is **disabled** in the CMC, such that when CMC users are configuring bucket lifecycle properties they will not see an option for auto-tiering (they will only see an option for auto-expiration). If you want to enable the auto-tiering feature in the CMC -- so that CMC users can apply auto-tiering to their buckets -- do the following:

1. In the CMC's [Configuration Settings](#) page, open the **Auto-Tiering** panel.
2. Set "Enable Auto-Tiering" to Enabled.
3. Configure how you want auto-tiering options to display to CMC users as they configure their buckets for auto-tiering. The system supports three different approaches:
 - If you want all CMC users to auto-tier to a **single system-default tiering destination account for which you are providing the account security credentials**, set "Enable Per Bucket Credentials" to Disabled and then enter the "Default Tiering URL" and the account security credentials.
 - If you want CMC users to be able to choose from a **pre-configured list of tiering destinations** (AWS S3, AWS Glacier, AWS Deep Archive, Google, and Azure by default) and not any other destinations of their own choosing, leave "Enable Per Bucket Credentials" at Enabled (the default) and leave "Enable Custom Endpoint" at Disabled (the default). With this approach users provide their own account security credentials for the tiering destination. You can edit the list of

pre-configured tiering destinations that will display for users, and the endpoints for those destinations, as described in ["Configure Tiering Destinations"](#) (page 210) below.

- If you want CMC users to be able to choose from a **pre-configured list of tiering destinations and also give users the option to specify a custom S3 tiering endpoint**, leave "Enable Per Bucket Credentials" at Enabled (the default) and set "Enable Custom Endpoint" to Enabled. With this approach users provide their own account security credentials for the tiering destination. If users specify a custom tiering endpoint, it must be an S3-compliant system and it cannot be an AWS Glacier, AWS Deep Archive, Azure, or Spectra BlackPearl endpoint. (Again, you can edit the pre-configured list as described in ["Configure Tiering Destinations"](#) (page 210) below.)
4. After finishing your edits in the **Configuration Settings** page, click **Save** at the bottom of the page to save your changes. These changes are applied dynamically and no service restart is required.

5.5.2.2. Configure Tiering Destinations

Unless you configure the system so that all users tier to the same one default tiering endpoint (as described above in Step 3's first bullet point), users when they configure auto-tiering for their buckets in the CMC will be able to choose from a list of several common tiering destinations. By default the destinations are AWS S3, AWS Glacier (Glacier Flexible Retrieval), AWS Deep Archive (Glacier Deep Archive), Google Cloud Storage, and Azure; and by default the endpoints for those destinations are as follows:

Destination	Default Endpoint
AWS S3	https://s3.amazonaws.com
AWS Glacier	https://s3.amazonaws.com
AWS Deep Archive	https://s3.amazonaws.com
Google Cloud Storage	https://storage.googleapis.com
Azure	https://blob.core.windows.net

If you want to make any changes or additions to this list -- for example if your users will tier to a custom destination -- you can do so with the `s3.tiering.endpoints.*` configuration settings. For details see ["s3.tiering.endpoints.*"](#) (page 833).

Note that the pre-configured default AWS S3 endpoint, AWS Glacier endpoint, and AWS Deep Archive endpoint are for the default AWS service region, which is *us-east-1*. So, with those default S3 and Glacier and Deep Archive destinations displaying in the CMC interface, a user can tier to an existing bucket that they own in the AWS *us-east-1* region, or they can have HyperStore create a tiering target bucket in that region. If you want users to be able to tier to a different AWS region you can add a tiering endpoint for that regional destination such as <https://s3.us-west-2.amazonaws.com>.

5.5.3. Setting Up Auto-Tiering for a Bucket

Subjects covered in this section:

- ["Setting Up Auto-Tiering for a Bucket \(CMC\)"](#) (page 211)
- ["Setting Up Auto-Tiering for a Bucket \(S3 API and Admin API\)"](#) (page 211)

After you have prepared the auto-tiering feature in the system as described in ["Preparing the Auto-Tiering Feature"](#) (page 209), you can set up auto-tiering for individual buckets as described in the sections below.

5.5.3.1. Setting Up Auto-Tiering for a Bucket (CMC)

HyperStore service users can configure auto-tiering for their own buckets through the CMC's [Bucket Properties](#) page. Here users can set auto-tiering attributes such as the tiering destination, the tiering schedule, tiering filters, and whether or not to temporarily retain a local copy of tiered objects.

Alternatively, as a system administrator you can set auto-tiering for a user's bucket by retrieving the user in the **Manage Users** page and then clicking "View User Data" to open a **Bucket Properties** page for the user's bucket. (This is supported only if you've configured the system to allow system administrators to view and manage users' stored data -- see ["Showing/Hiding CMC UI Functions"](#) (page 264)).

With either approach, the bucket first must be created in the usual way, and then the bucket can be configured for auto-tiering.

For details, see ["Configure a Bucket Lifecycle Policy"](#) (page 315).

Note Auto-tiering cannot be enabled for a bucket that has an underscore in its name. For this reason it's best not to use underscores when naming buckets in HyperStore.

5.5.3.2. Setting Up Auto-Tiering for a Bucket (S3 API and Admin API)

To configure auto-tiering rules on a bucket by using the S3 API, your S3 client application will need to use HyperStore extensions to the S3 API method *PutBucketLifecycleConfiguration*. The extensions take the form of request headers to specify the bucket's auto-tiering attributes. For details about these API extensions see "PutBucketLifecycleConfiguration" in the S3 API section of the *Cloudian HyperStore AWS APIs Support Reference*.

If you plan to configure auto-tiering on a bucket by calling the *PutBucketLifecycleConfiguration* S3 API method, you will **first need to use the HyperStore Admin API to store into the system the tiering destination account security credentials** that HyperStore should use when auto-tiering objects from that bucket. For example, if you want to use the S3 API method *PutBucketLifecycleConfiguration* to configure HyperStore source bucket "my-bucket" to auto-tier to AWS S3, you (or your application) must first use the HyperStore Admin API to post the security credentials that HyperStore should use when accessing AWS S3 on behalf of source bucket "my-bucket". The same requirement applies to other destination types. For details about the relevant Admin API methods, see the "tiering" section of the *Cloudian HyperStore Admin API Reference*.

5.5.4. Accessing Auto-Tiered Objects

After objects have been auto-tiered to a destination system, there are two ways to access those objects:

- ["Indirect Access Through HyperStore"](#) (page 211)
- ["Direct Viewing Through the Destination System"](#) (page 213)

5.5.4.1. Indirect Access Through HyperStore

When you use the CMC's [Objects](#) interface to view the contents list for a HyperStore storage bucket, objects that have been auto-tiered appear in the list and are marked with a [special icon](#). Your options for retrieving such objects through the CMC depend on how auto-tiering was configured for the bucket.

First, for any auto-tiered object (regardless of bucket configuration or tiering destination), the object can be retrieved by temporarily **restoring** a copy of the object into the local bucket. The CMC **Buckets & Objects** interface supports [temporarily restoring auto-tiered objects](#), for a length of time that you can specify.

Restoration of auto-tiered objects does not happen instantly. For example, for an object in Amazon S3, it can take up to six hours before a copy of the object is restored to HyperStore storage. For an object in Glacier it can be up nine hours, factoring in the time it takes for an object to be restored from Glacier to Amazon S3, before being restored to HyperStore. In the interim, the object is marked with an icon that indicates that the object is in the process of being restored. During this stage you cannot download the object.

After a copy of an object has been restored, the icon next to the object name changes again and you can then download the object through the **Buckets & Objects** interface in the usual way.

As a second option for retrieving auto-tiered objects, some objects may be directly downloadable through the **Buckets & Objects** interface without any need for first restoring the objects. This is supported only if both of the following are true:

- The tiered objects are in Amazon S3, Google Storage Cloud, Azure, or a Custom S3 destination (not Amazon Glacier or Spectra BlackPearl)
- The bucket's auto-tiering is configured to support **Streaming** or **Caching** (Stream & Restore) of auto-tiered objects. These options are available (for most destination types) when the bucket's auto-tiering policy is configured by the bucket owner.

If you're uncertain whether an auto-tiered object meets these requirements, you can try directly downloading the auto-tiered object by clicking on its name. If direct download is not supported for the object, a response message will indicate that you need to temporarily restore a local copy of the object rather than directly downloading it.

Note As an alternative to accessing auto-tiered objects through the CMC, you can use a third party S3 client application to submit *RestoreObject* requests (for any auto-tiered objects) or *GetObject* requests (for auto-tiered objects that support streaming or caching) to the HyperStore system's S3 Service. For more information about these S3 API calls see the [Cloudian HyperStore AWS APIs Support Reference](#)

Deletion of Tiered Objects or of Buckets Configured for Tiering

If you want to delete an object that has been auto-tiered, you can do so by deleting the object through the **Buckets & Objects** interface. You do not need to restore the object first. When the HyperStore system is deleting an auto-tiered object, it first triggers the deletion of the object from the destination system, and then after that succeeds it deletes the [local reference to the object](#).

If you delete an empty local bucket that has been configured for auto-tiering, the system will also automatically delete the tiering destination bucket.

Note The behavior described above regarding deletion of tiered objects and deletion of buckets configured for auto-tiering is the default system behavior. Optionally, you can configure the system so that objects and buckets are **not** deleted from the tiering destination system when they are deleted from the local HyperStore system. This option is available only if the tiering destination is S3-compatible. For details about this configurable setting see ["s3.tiering.forwardDeletes"](#) (page 836).

Note As an alternative to the CMC, you can use a third party S3 client application to submit *DeleteObject* requests to the HyperStore system's S3 Service. For more information see the **Cloudian HyperStore AWS APIs Support Reference**

5.5.4.2. Direct Viewing Through the Destination System

IMPORTANT ! Do not overwrite or delete tiered objects directly through the destination system's interfaces. Doing so will cause a discrepancy between the local metadata in HyperStore and the actual data in the destination bucket. If users want to overwrite or delete tiered objects they should do so through HyperStore interfaces (such as the CMC or an S3 application accessing the HyperStore S3 Service). In the case of auto-tiering from one HyperStore region or system to another HyperStore region or system, any overwriting or deleting of objects should be done through the source bucket not the destination bucket.

Objects auto-tiered to a destination system can be viewed directly through the destination system's standard interfaces, such as the AWS Management Console for objects that have been tiered to AWS S3.

For example, if a bucket owner supplied her own AWS credentials when configuring her HyperStore bucket for auto-tiering to AWS S3, she can log into her AWS account and see the HyperStore auto-tiering destination bucket (either named as she had specified or automatically named by HyperStore -- see **"Tiering Destination Accounts, Credentials, and Buckets"** (page 205) for detail). After objects have been auto-tiered from HyperStore to her AWS destination bucket, she can view the bucket content list and retrieve individual tiered objects directly through AWS.

In the case of auto-tiering from one HyperStore region to another region in the same HyperStore system, the tiered objects are accessible through the CMC's **Buckets & Objects** interface, by selecting the destination region.

5.6. Cross-Region Replication

5.6.1. Cross-Region Replication Feature Overview

Subjects covered in this section:

- *Introduction (immediately below)*
- **"Cross-Region Replication Versus Auto-Tiering"** (page 214)
- **"Bi-Directional Replication"** (page 215)
- **"Cross-Region Replication Impact on Usage Tracking"** (page 215)
- **"Cross-Region Replication of Encrypted Objects"** (page 216)
- **"Cross-Region Replication and Object Lock (WORM)"** (page 216)
- **"Cross-System Replication"** (page 216)
- **"Replication Error Handling"** (page 217)
- **"Handling of Pre-Existing Objects"** (page 218)

Like Amazon S3, HyperStore supports cross-region replication (CRR). This feature may be valuable if your HyperStore system consists of multiple [service regions](#). With cross-region replication, a bucket in one service region can be configured so that all objects uploaded into the bucket are replicated to a chosen destination bucket in a different service region within the same HyperStore system. This feature enables a bucket owner to enhance the protection of data by having it stored in two geographically dispersed service regions. The feature is also useful in cases where a bucket owner wants to have the same set of data stored in two different regions in order to minimize read latency for users in those regions.

If you wish, you can also use the CRR feature within a single HyperStore service region, so that objects uploaded into one bucket are replicated to a different bucket in the same service region.

As is the case with Amazon S3's implementation of this feature, with HyperStore **both the source bucket and the destination bucket must have "versioning" enabled** in order to activate cross-region replication.

Object metadata — including any access permissions assigned to an object, and any [user-defined object metadata or object tags](#) — is replicated to the destination bucket along with the object data itself. (The exception is that if an object in the source bucket has an `x-amz-expiration` header, HyperStore does not replicate this header.)

As with Amazon S3, HyperStore's implementation of the cross-region replication feature **does not replicate**:

- Objects that were already in the source bucket before the bucket was configured for cross-region replication (except as noted in ["Handling of Pre-Existing Objects" \(page 218\)](#))
- Objects that are encrypted with user-managed encryption keys (SSE-C) or AWS KMS managed encryption keys
- Objects that are themselves replicas from other source buckets. If you configure "bucket1" to replicate to "bucket2", and you also configure "bucket2" to replicate to "bucket3", then an object that you upload to "bucket1" will get replicated to "bucket2" but will not get replicated from there on to "bucket3". Only objects that you directly upload into "bucket2" will get replicated to "bucket3".
- Deletions of specific object versions.
 - In the case of an object deletion request that **specifies the object version**, the object version is deleted from the source bucket but is **not** deleted from the destination bucket.
 - In the case of an object deletion request that **does not specify the object version**, the deletion marker that gets added to the source bucket is replicated to the destination bucket.

Note HyperStore currently supports only Version 1 of the Amazon S3 specification for replication configuration XML, not Version 2. Those two versions differ in regard to whether or not deletion markers are replicated. The behavior described above is the Version 1 behavior, which is implemented by HyperStore.

5.6.1.1. Cross-Region Replication Versus Auto-Tiering

The key differences between cross-region replication and [auto-tiering](#) are:

- With cross-region replication each replicated object is **stored in both the source bucket and the destination bucket**. With auto-tiering, after an object is auto-tiered the object data is by default stored only in the destination bucket (although there is an option to retain a local copy for a configurable period of time).
- Cross-region replication replicates objects to the destination bucket **immediately after they've been uploaded to the source bucket**. With auto-tiering, typically the tiering of the object occurs on a user-

defined schedule, after the objects have been in the source bucket for a period of time (although there is an option to tier immediately).

- Cross-region replication replicates **nearly all object metadata and user-defined tags** along with the object data, so that an object's full set of metadata resides both at the source and at the destination. With auto-tiering only basic metadata accompanies the tiered object, while the object's full metadata continues to be stored locally at the source.
- Cross-region replication requires that **versioning** be enabled on both the source bucket and the destination bucket. Auto-tiering does not have this versioning requirement.
- With cross-region replication the destination bucket is typically within the **same HyperStore system** as the source bucket (although replicating to an external system is supported, as described in "[Cross-System Replication](#)" (page 216)). With auto-tiering the destination bucket is typically in an external system (although tiering to a bucket within the same HyperStore system is supported).

You cannot use cross-region replication and auto-tiering on the same source bucket.

5.6.1.2. Bi-Directional Replication

HyperStore supports bi-directional replication, whereby you configure two buckets to replicate to each other. For example, objects directly uploaded into "bucket1" can be replicated to "bucket2", while objects directly uploaded into "bucket2" are replicated to "bucket1".

As with the HyperStore cross-region replication generally, objects are not replicated if they are themselves replicas from another source bucket. Only objects directly uploaded into a bucket by a client application will be replicated. In the context of bi-directional replication this means that objects that are uploaded directly into one bucket will be replicated to the other bucket, but they will not then be replicated back into the original bucket and so on in an endless loop.

5.6.1.3. Cross-Region Replication Impact on Usage Tracking

If a user configures cross-region replication and consequently objects are replicated from a source bucket in one HyperStore region system to a destination bucket in a different HyperStore region, the replica data in the destination region will count toward the user's Stored Bytes count in the destination region as well as in the source region. For instance a 100MB object that gets replicated in this way would count as 100MB toward the user's Stored Bytes count in the source region and also as 100MB toward the user's Stored Bytes count in the destination region, for a total of 200MB across the system. (Also, for each replicated object there are about 50 bytes of object metadata associated with implementing this feature).

Note also that to enable cross-region replication the source bucket and destination bucket must both have "versioning" enabled. Once versioning is enabled, when objects are modified by users the system continues to store the older version(s) of the object as well as storing the new version. In a cross-region replication context, this means that over time multiple versions of an object may come to be stored in both the source bucket and the destination bucket, with each version counting toward Stored Bytes counts in both buckets.

Impact on System-Wide Stored Byte Count and Licensed Max Storage Limit

When users use cross-region replication to replicate objects from one HyperStore bucket to another HyperStore bucket, the objects in the source bucket and the object replicas in the destination bucket **both** count toward your system's stored byte count -- the count that is used to determine whether you are in compliance with your licensed maximum storage limit. In this respect bucket-to-bucket cross-region replication is different than storage policy based replication or erasure coding of objects within a region, which is treated as overhead and not counted toward your stored byte count.

5.6.1.4. Cross-Region Replication of Encrypted Objects

When cross-region replication is configured for a source bucket:

- Objects encrypted by the regular SSE method **are** replicated to the destination bucket.
- Objects encrypted by the AWS KMS or SSE-C methods **are not** replicated to the destination bucket.

Note Cross-region replication is not supported for a source bucket that has SSE-KMIP configured. This applies to buckets that use a storage policy that has been configured for SSE-KMIP as well as buckets for which SSE-KMIP has been configured at the bucket level.

5.6.1.5. Cross-Region Replication and Object Lock (WORM)

HyperStore does not support applying cross-region replication on a source bucket that has Object Lock enabled. However, an Object Lock enabled bucket is allowed to be the destination bucket in a cross-region replication relationship.

5.6.1.6. Cross-System Replication

Conventional cross-region replication replicates data from a source bucket to a destination bucket in a different service region within the same HyperStore system. HyperStore also supports an extension to cross-region replication whereby all objects uploaded to a HyperStore source bucket are replicated to a destination bucket in an external S3 compatible system. This is known as **cross-system replication (CSR)**. The external system can be either of the following:

- A different, independent HyperStore system (with its own user base and service regions and management controls)
- An external third party system with native S3 API support, such as AWS S3

Note

* Clouidian, Inc. has tested and verified only HyperStore and AWS as cross-system replication destinations. CSR should work for other native S3 destinations, as long as those destination systems 1) fully support the S3 API with respect to putting, listing, and getting versioned objects; and 2) base their ordering of object versions on write-time (exact time at which the object version is put to the system's S3 service by an external client). Any departures from these requirements may result in versioned objects being ordered differently in the replication destination bucket than they are in the source bucket in the local HyperStore system. This may make it difficult to tell, in the destination bucket, which of an object's versions is truly the current version, which is the next-most-current version, and so on. If you want to use CSR with a native S3 destination system other than HyperStore or AWS, Clouidian recommends that you first use CSR for just a small number of objects, and then confirm in the destination bucket that the object version ordering matches the ordering in the source bucket. If you encounter discrepancies, contact Clouidian Support.

* Neither Microsoft Azure nor Google Cloud Storage are native S3 and so those systems are not supported for cross-system replication.

In most respects cross-system replication works just like cross-region replication, with the same behaviors and limitations described in all the preceding sections of the [Cross-Region Replication Feature Overview](#).

However, there are additional limitations and caveats specific to cross-system replication:

- For the most part, **cross-system replication does not replicate object ACLs**. This is because with cross-system replication the source system and destination system have different user bases, and so grants of permissions to specific users or groups in the source system cannot be meaningfully applied to the destination system. The **exception** is that if the destination is an **external HyperStore system**, cross-system replication will replicate:
 - Object permissions granted to S3 "predefined groups" (the 'Authenticated Users' group or the 'All Users' group)
 - Standard S3 "canned ACLs" (such as 'authenticated-read' or 'public-read')
- With cross-system replication, **bi-directional replication is not supported**. Do not replicate data from (for example) *bucket1* in your local HyperStore system to *bucket2* in an external system, while data in *bucket2* is also replicated to *bucket1*.
- With cross-system replication, **replication from bucket-to-bucket-to-bucket is not supported**. Do not replicate data from (for example) *bucket1* in your local HyperStore system to *bucket2* in an external system, while data in *bucket2* is also replicated to *bucket3* in any system. Combining this CSR prohibition with the aforementioned prohibition against bi-directional CSR, it can be said that if you set up CSR from a source bucket to a destination bucket in an external system, **the destination bucket should not use CRR or CSR to replicate its data to any bucket**.

In the CMC, cross-system replication is **disabled by default**, such that CMC users who are configuring bucket replication will not be presented with the additional fields necessary to configure cross-system replication (such as the destination endpoint). For information about enabling cross-system replication support in the CMC, see ["Preparing the Cross-Region Replication Feature"](#) (page 218).

5.6.1.7. Replication Error Handling

Objects are replicated to the destination bucket immediately after they are uploaded to the source bucket. The replication request is initiated by whichever S3 Service node processes the upload request for the original object. Errors in replicating an object to the destination bucket are handled as follows:

- If the destination system returns an HTTP 403 or 404 error when HyperStore tries to replicate an object to the destination, this is treated as a permanent error. In the ["Cross-Region Replication request log \(cloudian-crr-request-info.log\)"](#) (page 685) on the node that initiates the replication request, an entry for the object replication attempt is written with status FAILED. The system does not retry replicating the object. Examples of scenarios that could result in permanent errors like this are if the destination bucket has been deleted, or if versioning has been disabled on the destination bucket, or if the source bucket owner no longer has write permissions on the destination bucket.
- Any other type of error -- such as the destination region being unreachable due to a network partition, or the destination region returning an HTTP 5xx error -- is treated as temporary. In the Cross-Region Replication request log on the node that initiates the replication request, an entry for the object replication attempt is written with status PENDING. The object replication job will be queued and retried. Retries of object replication jobs with PENDING status are executed by a [system cron job](#) that runs once every four hours. These retries are executed by the **cron job node**. (To see which node this is in your cluster, in the CMC's [Cluster Information](#) page look to see the "System Monitoring / Cronjob Primary Host"). The retries for an object will recur once every four hours until either the object is successfully replicated to the destination bucket or a permanent error is encountered. Each retry attempt

results in a new entry in the Cross-Region Replication request log on the cron job node, with a status of either COMPLETED, PENDING, or FAILED.

IMPORTANT ! If some objects uploaded to a source bucket encounter temporary replication errors, such that those objects are subject to the replication retry cron job, that cron job will also trigger an assessment of the replication status of **all objects in that bucket**. Then, any objects that are not yet replicated will be replicated to the destination bucket -- so that the destination bucket and the source bucket are fully "synced".

A permanent failure for replication of an object applies **only to that object** and does not impact the processing of other objects subsequently uploaded into the same source bucket. The system will continue to replicate -- or attempt to replicate -- other objects that subsequently get uploaded into the bucket. Those objects may also encounter permanent errors, but the system will continue to try to replicate newly uploaded objects unless you disable cross-region replication on the source bucket.

There is no limit on the number of replication retries for a given object or on the number of objects that are queued for retry.

Note If an attempt to replicate an object to the destination -- either the original attempt or a retry attempt -- results in a FAILED status in the Cross-Region Replication request log, so that there will be no further retries, this triggers an Alert in the CMC's **Alerts** page. This type of alert falls within the alert rules for S3 service errors (there is not a separate alert rule category for CRR replication failures).

5.6.1.8. Handling of Pre-Existing Objects

Enabling cross-region replication or cross-system replication on a bucket does not cause the replication of objects that were already in the source bucket when cross-region replication or cross-system replication was enabled. However, if you wish you can run a command to trigger the replication of a bucket's pre-existing objects. For details see ["Triggering the Replication of a Bucket's Pre-Existing Objects"](#) (page 220).

5.6.2. Preparing the Cross-Region Replication Feature

HyperStore's cross-region replication feature is enabled by default. Optionally you can modify certain aspects of the feature's implementation as described below.

- ["Enabling CMC Support for Cross-System Replication"](#) (page 218)
- ["Setting the Scope of the Replication Retry Cron Job"](#) (page 219)

5.6.2.1. Enabling CMC Support for Cross-System Replication

By default CMC users can configure replication from a source bucket to a destination bucket that's in the same HyperStore system as the source bucket. If you wish you can also give CMC users the option to configure replication from a HyperStore source bucket to a destination bucket in an external system that has native S3 API support (cross-system replication).

Note Before enabling CMC support for cross-system replication you should review ["Cross-System Replication"](#) (page 216) including the limitations and caveats.

To enable CMC support for cross-system replication, set the configuration setting ["cmc.externalCRR.enabled"](#) (page 719) to *true* (by default it is *false*). (For instructions on changing this setting and applying your change see the setting description).

Once you've made this change, when CMC users [configure cross-region replication for a source bucket](#) they will be presented with additional fields that allow for replicating to a destination bucket in an external system.

5.6.2.2. Setting the Scope of the Replication Retry Cron Job

Once every four hours a HyperStore [cron job](#) processes any pending cross-region replication jobs. Typically with cross-region replication, objects are replicated to the destination bucket as soon as they are uploaded to the source bucket. Pending cross-region replication tasks result when an attempt to replicate an object from the source bucket to the destination bucket returns a temporary error such as a connection failure or an HTTP 5xx error. The cron job retries the replication of such objects.

By default the cron job simply processes the queue of replication retries, from recent replication tasks that encountered temporary errors. However, if you wish you can configure the system so that each time the cron job runs it also performs a complete sync-up of each bucket for which there are replication tasks in the retry queue. With this complete sync-up, any objects that are in these source buckets but not in their corresponding destination buckets will be replicated to the destination buckets. This includes objects that were in the source buckets before cross-region replication was enabled on the buckets, if any. (The exceptions are objects that were in the source buckets before versioning was enabled on the bucket, if any; and objects that are [encrypted by the SSE-C or AWS-KMS methods](#), if any. Such objects will not be replicated.)

If you want the cron job to perform a complete source bucket to destination bucket sync-up for source buckets for which there are pending replication tasks -- rather than simply retrying the pending replication tasks -- set the configuration setting ["s3.crr.syncBucket"](#) (page 817) to *true* (by default it is *false*). (For instructions on changing this setting and applying your change see the setting description).

Note This change in the cron job behavior will not affect replication processing for source buckets for which there are not any pending replication tasks in the retry queue. It applies only to buckets for which there are pending replication tasks in the retry queue. If you want to trigger a replication sync-up for a particular source bucket you can do so as described in ["Triggering the Replication of a Bucket's Pre-Existing Objects"](#) (page 220).

5.6.3. Setting Up Cross-Region Replication for a Bucket

Bucket owners can configure replication from a source bucket to destination bucket. This can be done either through the CMC or through a different S3 client application that invokes the HyperStore implementation of the S3 API.

- ["Setting Up Cross-Region Replication for a Bucket \(CMC\)"](#) (page 220)
- ["Setting Up Cross-Region Replication for a Bucket \(S3 API\)"](#) (page 220)
- ["Triggering the Replication of a Bucket's Pre-Existing Objects"](#) (page 220)

5.6.3.1. Setting Up Cross-Region Replication for a Bucket (CMC)

Note Before configuring cross-region replication on a source bucket, be sure that both the source bucket and the destination bucket have **versioning enabled**. This is required in order to use cross-region replication. For information about enabling versioning on a bucket in HyperStore, see ["Enable/Disable Versioning for a Bucket" \(page 333\)](#).

In the CMC, bucket owners can enable and configure cross-region replication through the **Buckets & Objects** page. For a given source bucket, cross-region replication is among the options that can be configured as bucket properties. For detail, see ["Configure Cross-Region Replication for a Bucket" \(page 331\)](#). Along with specifying a destination bucket to which objects will be replicated from the source bucket, bucket owners can indicate whether they want all newly uploaded objects to be replicated or only objects for which the full object name starts with a particular prefix (such as `/profile/images`). If [cross-system replication](#) is enabled in your system, bucket owners can also specify the destination system endpoint and their security credentials for the destination system.

Alternatively, as a system administrator you can configure cross-region replication for a user's bucket by retrieving the user in the **Manage Users** page and then clicking "View User Data" to open a **Bucket Properties** page for the user's bucket. (This is supported only if you've configured the system to allow system administrators to view and manage users' stored data -- see ["Showing/Hiding CMC UI Functions" \(page 264\)](#)).

5.6.3.2. Setting Up Cross-Region Replication for a Bucket (S3 API)

S3 applications can call the HyperStore S3 API to configure a source bucket for cross-region replication. For detail see "PutBucketReplication" in the S3 API section of the *Cloudian HyperStore AWS APIs Support Reference*.

Along with supporting conventional cross-region replication, HyperStore's implementation of the *PutBucketReplication* call includes support for extension headers that can be used for cross-system replication. Before using this API extension you should review ["Cross-System Replication" \(page 216\)](#) including the limitations and caveats noted in that section.

As with Amazon S3, the HyperStore implementation of the *PutBucketReplication* call requires that **versioning be enabled** on both the source and the destination bucket before cross-region replication can be applied. HyperStore supports the standard S3 *PutBucketVersioning* operation for enabling versioning on a bucket.

HyperStore also supports the *GetBucketReplication* method for retrieving a bucket's current CRR configuration, and the *DeleteBucketReplication* method for disabling CRR on a source bucket for which it is currently enabled. For detail see the S3 API section of the *Cloudian HyperStore AWS APIs Support Reference*.

5.6.3.3. Triggering the Replication of a Bucket's Pre-Existing Objects

Note If you want to trigger the replication of a bucket's pre-existing objects it's best to do so immediately after cross-region replication has been enabled on the bucket (and before additional objects are uploaded to the bucket and automatically replicated). Triggering sync-up for a bucket after newly uploaded objects have already been replicated to the destination bucket can result in object version sequencing mis-matches between the source bucket and the destination bucket.

Enabling cross-region replication or cross-system replication on a bucket does not cause the replication of objects that were already in the source bucket when cross-region replication or cross-system replication was enabled. However, if you wish **you can trigger replication of the pre-existing objects** in the source bucket by logging into any HyperStore node and running this command:

```
curl -d "" http://localhost:80/.system/syncbucket?bucketName=string
```

For example:

```
curl -d "" http://localhost:80/.system/syncbucket?bucketName=my.bucket
```

This will replicate to the destination bucket any objects in the source bucket that have not already been successfully replicated, including objects that were in the source bucket before you enabled cross-region replication. (The exceptions are objects that were in the source bucket before versioning was enabled on the bucket, if any; and objects that are [encrypted by the SSE-KMIP, AWS KMS, or SSE-C methods](#), if any. Such objects will not be replicated.)

5.7. Object Lock

5.7.1. Object Lock Feature Overview

Subjects covered in this section:

- ["HyperStore Object Lock License Types: Certified and Compatible" \(page 221\)](#)
- ["Object Lock Audit Logging" \(page 222\)](#)
- ["Object Lock and Bucket Lifecycle Policies" \(page 222\)](#)
- ["Object Lock and Cross-Region Replication" \(page 223\)](#)
- ["Object Lock and S3 Notifications" \(page 223\)](#)
- ["Object Lock and User Deletion" \(page 223\)](#)
- ["Object Lock and File Services" \(page 223\)](#)

5.7.1.1. HyperStore Object Lock License Types: *Certified and Compatible*

HyperStore can implement WORM (Write Once Read Many) protection for stored objects by supporting the standard AWS S3 "Object Lock" functionality. To use the Object Lock feature you must have a HyperStore license that activates this feature. Two different types of HyperStore licensing are available for Object Lock functionality:

- **Certified Object Lock.** With this type of Object Lock:
 - HyperStore is fully compliant with the AWS S3 APIs relating to Object Lock, and the safeguards they provide against deletion or modification of locked objects through the S3 API.
 - HyperStore also implements safeguards against deletion or modification of locked objects through means outside of the S3 API. Specifically:
 - To use the Certified Object Lock feature you must disable password-based *root* access to HyperStore nodes and enable the HyperStore Shell (HSH). The HSH allows administrators to log into HyperStore nodes in a manner that is much more restrictive than *root* access.
 - With Certified Object Lock there is no mechanism to delete or modify locked objects through HyperStore administrative commands or the HyperStore Admin API.

Certified Object Lock is appropriate for HyperStore customers who are subject to the data protection mandates of U.S. SEC-17a or a comparable regulatory regime.

- **Compatible Object Lock.** With this type of Object Lock:
 - HyperStore is fully compliant with the AWS S3 APIs relating to Object Lock, and the safeguards they provide against deletion or modification of locked objects through the S3 API. In this respect Compatible Object Lock operates like Certified Object Lock.
 - Unlike Certified Object Lock, with Compatible Object Lock no additional data protection measures are enforced beyond those enforced by the S3 API. Specifically:
 - With Compatible Object Lock you are not required to disable password-based *root* access to HyperStore nodes and you are not required to enable the restrictive HyperStore Shell.
 - With Compatible Object Lock you can use a HyperStore Admin API call to delete all objects from an Object Lock enabled bucket -- including objects that are still within their lock retention period -- and (optionally) to delete the bucket itself. To use this Admin API call on an Object Lock enabled bucket you must obtain a temporary security token from Cloudian Support.

Compatible Object Lock is appropriate for HyperStore customers who want to utilize the WORM functionality provided by the AWS S3 Object Lock APIs, but who are not subject to the data protection mandates of U.S. SEC-17a or a comparable regulatory regime.

If you have a license for Certified Object Lock, you can subsequently switch to a license for Compatible Object Lock if you wish (by requesting this license change from Cloudian Support). But the reverse is not true: **You cannot switch from a Compatible Object Lock license to a Certified Object Lock license.** This is because once Compatible Object Lock has been in use in your object storage system, the strict requirements of regulatory regimes such as U.S. SEC-17a can no longer be met.

For more information on applying Object Lock to buckets and objects in HyperStore, see ["Setting Up Object Lock for a Bucket"](#) (page 224).

For a summary of protections that the S3 API provides to locked objects, see ["Protections for Locked Objects"](#) (page 229).

For more information on deleting all objects from a locked bucket -- if your licensed Object Lock type is Compatible Object Lock -- see the "bucketops" section in the *Cloudian HyperStore Admin API Reference*.

5.7.1.2. Object Lock Audit Logging

All S3 client activity pertaining to setting Object Lock attributes on a bucket or on individual objects, and all S3 client attempts to delete locked objects, are logged in an audit log named *s3-worm.log*. For more information about this log see ["S3 Service Logs \(including Auto-Tiering, CRR, and WORM\)"](#) (page 682).

If your licensed Object Lock type is Compatible Object Lock and you use the Admin API to purge the content of a locked bucket, log messages regarding that purging operation are written to the Admin Service application log (*cloudian-admin.log*) and the Admin Service request log (*cloudian-admin-request-info.log*). For more information about these logs see ["Admin Service Logs"](#) (page 667).

5.7.1.3. Object Lock and Bucket Lifecycle Policies

HyperStore does not support applying bucket lifecycle policies for **auto-tiering** on a source bucket that has Object Lock enabled. However, the system does support auto-tiering to a destination bucket that has Object

Lock enabled.

HyperStore does support applying bucket lifecycle policies for **auto-expiration** on a source bucket that has Object Lock enabled:

- For a lifecycle policy that specifies an expiration schedule for current version of objects, Object Lock is irrelevant because for any bucket with versioning this type of expiration action only results in creation of a delete marker (and does not actually delete any object data from storage).
- For a lifecycle policy that specifies an expiration schedule non-current object versions, when a non-current object version reaches its expiration date the system checks for Object Lock and does not delete the non-current object version if it is still within its lock retention period. When the non-current object version's lock retention period ends, then the system deletes it at the next running of the auto-expiration job.

5.7.1.4. Object Lock and Cross-Region Replication

HyperStore does not support applying cross-region replication on a source bucket that has Object Lock enabled. However, an Object Lock enabled bucket is allowed to be the destination bucket in a cross-region replication relationship.

5.7.1.5. Object Lock and S3 Notifications

In the current release, HyperStore does not support S3 Notifications in regard to Object Lock related events.

5.7.1.6. Object Lock and User Deletion

The system will reject (with an HTTP 412 error response) any attempt to delete a user who currently owns a bucket that has Object Lock enabled:

- Before such a user can be deleted, the user's Object Lock enabled bucket(s) must be deleted.
- Before the user's Object Lock enabled bucket(s) can be deleted, all objects in those bucket(s) must be deleted. Depending on the type of Object Lock configuration used on those buckets and objects, it may be that the objects cannot be deleted until after the end of their retention period. For a summary of restrictions on deletion of "locked" objects through the S3 API, see ["Protections for Locked Objects" \(page 229\)](#).

Note If your licensed Object Lock type is Compatible Object Lock, you can delete a locked bucket and its contents through the Admin API after obtaining a temporary security token from Cloudian Support. See the "bucketops" section of the *Cloudian HyperStore Admin API Reference*.

5.7.1.7. Object Lock and File Services

For information about the relation between Object Lock and the HyperStore File Services functionality, see ["File Services Feature Overview" \(page 238\)](#).

5.7.2. Preparing the Object Lock Feature

Before Object Lock can be set up on buckets and objects your HyperStore system must meet these prerequisites:

- **You must have a HyperStore license that supports the Object Lock feature.** To check whether your license supports this feature, in the CMC's **Cluster Information** page see the "Object Lock License" field: it will indicate either "Certified" (if your licensed Object Lock type is Certified Object Lock) or "Compatible" (if your licensed Object Lock type is Compatible Object Lock) or "Disabled" (if your license does not support Object Lock). If you want to use the Object Lock feature but it is disabled by your current license, contact your Cloudian representative to inquire about obtaining a different license. For a description of how Certified Object Lock differs from Compatible Object Lock, see ["Object Lock Feature Overview"](#) (page 221).
- **If your licensed Object Lock type is Certified Object Lock, you must enable the HyperStore Shell and disable the *root* account password** on your HyperStore hosts. For instructions see:
 - ["Enabling the HSH"](#) (page 106)
 - ["Disabling the root Password"](#) (page 109)

These actions must be completed before Object Lock enabled buckets can be created. If your licensed Object Lock type is Certified Object Lock and you have not yet enabled the HyperStore Shell and disabled the root account password:

- A warning will display in the "Object Lock License" field in the CMC's **Cluster Information** page
- WARNING level log messages will be written to the Admin Service log *cloudian-admin.log*
- Any attempts to create an Object Lock enabled bucket through the CMC or directly through the S3 API will fail and return an error response.

Note If your licensed Object Lock type is **Compatible Object Lock**, the requirement to enable the HyperStore Shell and disable the *root* account password does not apply.

5.7.3. Setting Up Object Lock for a Bucket

Subjects covered in this section:

- ["Setting Up Object Lock on a Bucket \(S3 API or CMC\)"](#) (page 224)
- ["Setting Object Lock Attributes on Individual Objects in a Bucket \(S3 API or CMC\)"](#) (page 226)

5.7.3.1. Setting Up Object Lock on a Bucket (S3 API or CMC)

With a third party S3 application that supports the S3 APIs for object locking, or with the CMC, HyperStore users with appropriate permissions can:

1. Create a new bucket with Object Lock enabled.
2. Optionally, set a default Object Lock configuration for that bucket.
3. Check a bucket's Object Lock configuration status.

The sections that follow provide more information about each of these steps.

Note Object Lock can only be enabled on a new bucket, as the bucket is created. **Object Lock cannot be enabled on an already existing bucket.**

Creating a New Bucket with Object Lock Enabled

To create a new bucket with Object Lock enabled:

- **With a third party S3 client application**, the client application submits a **CreateBucket** request that includes the request header **x-amz-bucket-object-lock-enabled: true**. For HyperStore support of the S3 **CreateBucket** operation, see the S3 section of the **Cloudian HyperStore AWS APIs Support Reference**.
- **With the CMC**, as a user creates a new bucket in the CMC's **Add Bucket** interface she can select a checkbox to enable Object Lock for the bucket. For more information see **"Add a Bucket"** (page 307).

Note that:

- Enabling Object Lock on a new bucket (either through a third party S3 client or through the CMC) **automatically enables Versioning** on that bucket as well. Object Lock can only be used in combination with Versioning (which protects objects from being overwritten).
- If you create a bucket with Object Lock enabled, you **cannot disable Object Lock or suspend versioning** for the bucket (even if you never apply a default Object Lock configuration to the bucket).
- Enabling Object Lock on a new bucket **does not by itself have the effect of locking objects that are subsequently uploaded** into that bucket. It only makes it possible to lock such objects, using the methods described below.

Setting a Default Object Lock Configuration for the Bucket (Optional)

Once a new bucket has been created with Object Lock enabled, a default Object Lock configuration can optionally be set on the bucket. This Object Lock configuration will then by default be applied to all objects that are subsequently added to the bucket (it does **not** apply to any objects that were already in the bucket at the time that the default Object Lock configuration was set for the bucket). The default configuration can be overridden on a per-object basis, as described in **"Setting Object Lock Attributes on Individual Objects in a Bucket (S3 API or CMC)"** (page 226). If a default Object Lock configuration is not set on the bucket, then objects uploaded to the bucket will not be locked unless they have Object Lock attributes set on them on a per-object basis.

To set a default Object Lock configuration on the bucket:

- **With a third party S3 client application**, the bucket owner (or a user with **s3:PutBucketObjectLockConfiguration** permission on the bucket) submits a **PutObjectLockConfiguration** request to the HyperStore S3 Service. For HyperStore support of this S3 API operation see the S3 section of the **Cloudian HyperStore AWS APIs Support Reference**. Along with specifying a retention period in days or years, the request specifies whether the bucket's default Object Lock implementation is to be in **Governance** mode (which allows users who have the applicable permissions to change the object retention period or delete objects through the S3 API before their retention period completes) or **Compliance** mode (which does not allow any user to change the object retention period or delete objects through the S3 API before their retention period completes).
- **With the CMC**, the bucket owner can use the Object Lock tab of the **Bucket Properties** interface to set a default Object Lock configuration on the bucket. For more information see **"Configure Object Lock Properties for a Bucket"** (page 337).

With a default Object Lock configuration set on the bucket, **from that point forward whenever an object is uploaded to the bucket -- and lacks object-specific lock attributes -- the default retention period is applied**

to that object. For example with a 90 day default retention period, every object is locked for 90 days starting from the time of object upload. For objects with multiple versions, in this example each object version is locked for 90 days from time of object version upload.

When an object version is locked, HyperStore rejects an S3 *DeleteObject* request for that object version with an HTTP 403 Forbidden response. The exception is if Governance mode is being used and the requesting user is a user who has the applicable permissions; for more information see "[Protections for Locked Objects](#)" (page 229).

A third party S3 client application can **change or remove the default Object Lock configuration** on a bucket by submitting another *PutObjectLockConfiguration* request; or in the CMC a bucket's default Object Lock configuration can be changed or removed through the **Bucket Properties** interface. However, the change applies only from that time forward, to objects that are subsequently added to the bucket. Locks that are already in place on existing objects -- in accordance with the prior default configuration on the bucket -- are not impacted.

Note

- Object Lock protects each version of an object individually and does not prevent the creation of new versions of the object.
- For a locked object, an S3 *DeleteObject* request that does not specify an object version is allowed and only results in the creation of a delete marker for that object. No object data is actually deleted from storage.
- Once a bucket is assigned a default lock configuration, any S3 requests for uploading objects to that bucket must use Signature Version 4 request authentication. This is consistent with Amazon's Object Lock requirements.

Checking a Bucket's Object Lock Status

To check a bucket's current Object Lock status:

- **With a third party S3 client application**, the bucket owner (or a user with *s3:GetBucketObjectLockConfiguration* permission on the bucket) can submit a ***GetObjectLockConfiguration*** request to the HyperStore S3 Service. For HyperStore support of this S3 operation see the S3 section of the ***Cloudian HyperStore AWS APIs Support Reference***. The request response will indicate whether Object Lock is enabled on the bucket, and what the default Object Lock configuration is for the bucket (if any).
- **With the CMC**, in the bucket owner can view the list of buckets she owns and any buckets for which Object Lock is enabled will have a padlock icon beside the bucket name. The bucket owner can then view the **Bucket Properties** interface for the bucket and check the Object Lock tab to see what the default Object Lock configuration is for the bucket (if any).

5.7.3.2. Setting Object Lock Attributes on Individual Objects in a Bucket (S3 API or CMC)

In a [bucket that has Object Lock enabled](#), there is the option to set Object Lock attributes on individual objects. If the bucket has a default Object Lock configuration, then setting Object Lock attributes on individual objects will -- for those objects only -- override the bucket's default Object Lock configuration. If the bucket does not have a default Object Lock configuration, then setting Object Lock attributes on individual objects is the only way that objects will become locked.

With a third party S3 application, HyperStore users with appropriate permissions can:

- Set Object Lock attributes on objects as they are uploaded to the bucket.
- Set Object Lock attributes on existing objects that are already in the bucket.
- Check an object's lock status.

With the CMC, the bucket owner can:

- Set Object Lock attributes on existing objects that are already in the bucket.
- Check an object's lock status.

Note The CMC does not currently support setting Object Lock attributes on objects as they are uploaded to the bucket. Instead the user can upload the object to the bucket, and then set Object Lock attributes on the object.

The sections that follow provide more information about these options.

Setting Object Lock Attributes on Objects as They Are Uploaded

With a third party S3 application, objects being uploaded to on Object Lock enabled bucket can be assigned Object Lock attributes by the inclusion of the request headers ***x-amz-object-lock-mode***, ***x-amz-object-lock-retain-until-date***, and/or ***x-amz-object-lock-legal-hold*** with any of the standard S3 API requests for uploading objects:

- *PutObject*
- *CopyObject*
- *POST Object*
- *CreateMultipartUpload*

For HyperStore support of these S3 operations see the S3 section of the ***Cloudian HyperStore AWS APIs Support Reference***.

Similarly to the bucket default configuration options, the individual Object Lock configuration options include the choice between **Governance** retention mode and **Compliance** retention mode (as set by the ***x-amz-object-lock-mode*** request header).

Unlike the bucket default configuration options, the individual object configuration attributes also include an option for **Legal Hold** (as optionally set by the ***x-amz-object-lock-legal-hold*** request header). Legal Hold prevents the object from being deleted by any user through the S3 API, for an indefinite period of time until the Legal Hold is explicitly removed from the object (by a user who has the applicable permissions). Legal Hold can be used instead of having a defined retention date for the object, or in combination with having a defined retention date for the object. For example, if both a Governance retention date and a Legal Hold are set on an object, and then the Legal Hold is removed before the retention date, the object will continue to be protected by Governance mode retention until its retention date is reached. For a second example, if both a Compliance retention date and a Legal Hold are placed on an object, and the Compliance retention date is reached while the Legal Hold is still in place, the object continues to be protected by the Legal Hold until the Legal Hold is explicitly removed.

Note

* Setting retention attributes on an object as the object is uploaded can be done by the bucket owner, or by a user who has ***s3:PutObject*** and ***s3:PutObjectRetention*** permissions on the bucket. Setting a legal hold on an object as the object is uploaded can be done by the bucket owner or by a user who

has `s3:PutObject` and `s3:PutObjectLegalHold` permissions on the bucket.

- * Object upload requests that include Object Lock headers must use Signature Version 4 request authentication. This is consistent with Amazon's Object Lock requirements.
- * The CMC does not support setting Object Lock attributes on objects as they are uploaded.

Setting Object Lock Attributes on Existing Objects

- **With a third party S3 application**, existing objects in an Object Lock enabled bucket can be assigned Object Lock attributes by using the standard S3 requests ***PutObjectRetention*** and/or ***PutObjectLegalHold***. For HyperStore support of these S3 operations see the S3 section of the ***Cloudian HyperStore AWS APIs Support Reference***.

Note Setting retention attributes on an existing object can be done by the bucket owner or by a user who has `s3:PutObjectRetention` permission on the object. Setting or removing a legal hold on an existing object can be done by the bucket owner or by a user who has `s3:PutObjectLegalHold` permission on the object.

For an existing object that already has retention attributes, the ***PutObjectRetention*** request can be used to increase the existing retention period on the object but not to reduce the existing retention period (unless Governance mode retention is being used and the requesting user is a user who has the applicable permissions, as described in **["Protections for Locked Objects"](#)** (page 229)).

- **With the CMC**, the bucket owner can assign Object Lock attributes to an existing object in the bucket by accessing the object's **Object Properties** interface and using the Object Lock tab. For more information see **["Set Object Lock Attributes on an Object"](#)** (page 356).

Checking an Object's Lock Status

- **With a third party S3 application**, checking an object's lock status can be done by using the standard S3 requests ***GetObject***, ***HeadObject***, ***GetObjectRetention***, and/or ***GetObjectLegalHold***. For HyperStore support of these S3 operations see the S3 section of the ***Cloudian HyperStore AWS APIs Support Reference***.

An object's lock status will reflect the Object Lock attributes that were set directly on that individual object (if any), or otherwise will reflect the object's inheriting of the bucket's default Object Lock configuration (if any).

Note Checking an object's lock status can be done by the bucket owner or by a user who has `s3:GetObject`, `s3:GetObjectVersion`, `s3:GetObjectRetention` (for retention status), and `s3:GetObjectLegalHold` (for legal hold status) permissions on the object.

- **With the CMC**, the bucket owner can view the list of objects in her bucket, and locked objects will have a padlock icon beside the version ID of each version of the object. The bucket owner can then view an object version's **Object Properties** interface and check the Object Lock tab to see what the lock attributes are for that object version.

5.7.4. Protections for Locked Objects

The table below shows the key differences between Governance mode retention, Compliance mode retention, and Legal Hold in terms of how they **protect locked objects against premature deletion through the S3 API**. Recall that for all forms of Object Lock, the lock is applied to each individual version of an object.

	Governance Mode Retention	Compliance Mode Retention	Legal Hold
Delete a locked object version?	The bucket owner and users who have been granted both <i>s3:DeleteObjectVersion</i> and <i>s3:BypassGovernanceRetention</i> permission can use the <i>DeleteObject</i> request with an <i>x-amz-bypass-governance-retention: true</i> request header to delete a locked object version.	No user can delete a locked object version	No user can delete a locked object version
Remove the lock on an object version?	The bucket owner and users who have been granted both <i>s3:PutObjectRetention</i> and <i>s3:BypassGovernanceRetention</i> permission can use the <i>PutObjectRetention</i> request with an <i>x-amz-bypass-governance-retention: true</i> request header to remove the retention lock on a object version.	No user can remove the retention lock on an object version	The bucket owner and users who have been granted <i>s3:PutObjectLegalHold</i> permission can use the <i>PutObjectLegalHold</i> request to remove the legal hold on an object version.
Reduce the retention period for an object version?	The bucket owner and users who have been granted both <i>s3:PutObjectRetention</i> and <i>s3:BypassGovernanceRetention</i> permission can use the <i>PutObjectRetention</i> request with an <i>x-amz-bypass-governance-retention: true</i> request header to reduce the retention period for a locked object version.	No user can reduce the retention period for an object version	The bucket owner and users who have been granted <i>s3:PutObjectLegalHold</i> permission can use the <i>PutObjectLegalHold</i> request to remove the legal hold on an object version.

Note In the CMC, no users other than the bucket owner can access a bucket or the objects that it contains. So in the table above, the statements about users who have been granted various permissions relating to locked objects apply only to users who are using third party S3 client applications -- not the CMC.

5.7.4.1. Using the Admin API to Purge All Objects from a Locked Bucket

If your licensed Object Lock type is **Compatible Object Lock**, you can use a HyperStore Admin API call to purge all of the objects in a locked bucket. Note that:

- This operation deletes **all** objects from the bucket, regardless of whether the objects are protected by Governance mode, Compliance mode, or Legal Hold.

- There is no option to selectively delete objects from the bucket -- the operation only supports deleting **all** versions of **all** objects.
- The operation allows you (optionally) to delete the bucket itself, as well as all of the objects in it.
- The operation requires that you obtain a temporary security token from Cloudian Support.
- The operation is not allowed if your licensed Object Lock type is Certified Object Lock.

For more information see the "bucketops" section in the Cloudian HyperStore Admin API Reference.

5.8. Object Metadata

5.8.1. Object Metadata and Search Feature Overview

Subjects covered in this section:

- *Introduction (immediately below)*
- **"System-Defined and User-Defined Object Metadata"** (page 230)
- **"Object Tags"** (page 231)
- **"Object Metadata Storage in the Metadata DB"** (page 232)
- **"Cross-Region Replication and Auto-Tiering of Object Metadata and Tags"** (page 233)
- **"Object Metadata Based Search"** (page 233)

One of the advantages of S3-compliant object storage is the ability to associate rich metadata with each object, including metadata defined by users. HyperStore supports the creation of user-defined object metadata through standard S3 API calls. HyperStore also supports the ability to search for objects based on their metadata.

5.8.1.1. System-Defined and User-Defined Object Metadata

There are two types of S3-compliant object metadata:

- System-defined object metadata -- The HyperStore system automatically defines and sets a variety of object metadata having to do with object attributes and status
- User-defined object metadata -- S3 application users can define and set object metadata by using `x-amz-meta-*` request headers when objects are uploaded

System-Defined Object Metadata

For each object, HyperStore maintains object metadata including (but not limited to) the following:

- Creation time
- Last modified time
- Last accessed time

Note An object's last accessed time is updated only if an S3 `GetObject` is performed on the object and the object is in a bucket for which a last accessed time based [bucket lifecycle rule](#) is in effect.

- Size
- ACL information
- Version, if applicable
- Public URL, if applicable
- Compression type, if applicable
- Encryption key, if applicable
- Auto-tiering state, if applicable

User-Defined Object Metadata

S3 user applications can define and set object metadata by including *x-amz-meta-** request headers in the object upload request. The application user defines both the metadata key and its value. For example if metadata identifying a document's author and reviewer was desired, this could be achieved with request headers such as *x-amz-meta-author: hernandez* and *x-amz-meta-reviewer: schmidt*. This is a standard S3 API feature that is supported by HyperStore's S3 Service. For more information see ["Creating and Retrieving User-Defined Object Metadata and Tags" \(page 234\)](#).

The maximum size limit for *x-amz-meta-** based metadata -- the maximum allowed total size of all *x-amz-meta-** header names and header values combined -- is 2KB per object.

5.8.1.2. Object Tags

Object tags -- user-defined key-value pairs that are associated with an object, such as *department=engineering* or *status=complete* -- are similar to object metadata in that they encapsulate information about an object. However, from the perspective of AWS S3, object tags are distinct from object metadata because they are created differently and typically serve different purposes.

Object tags can be attached to an object as it's being uploaded, by the use of a single *x-amz-tagging* request header which supports having a multi-segment value. But more commonly -- and differently from object metadata -- object tags can be attached to an object that's already in storage, by the use of the S3 API call *PutObjectTagging* that creates a tagging sub-resource for a specified existing object. Further, an object's tags can be updated at any time by that same S3 API call. (By contrast, an object's *x-amz-meta-** based object metadata cannot be modified, except by making a copy of the object and specifying different metadata during the copy operation.)

Object tagging is a standard S3 API feature that is supported by HyperStore's S3 Service. For more information see ["Creating and Retrieving User-Defined Object Metadata and Tags" \(page 234\)](#).

Another thing that distinguishes object tags from *x-amz-meta-** based object metadata is that object tags can serve as conditions in a bucket permissions policy or IAM permissions policy, or as filtering criteria in a bucket lifecycle policy. The CMC does not currently support this capability, but HyperStore's S3 API and IAM API do.

Note The HyperStore Search Service (described in ["Object Metadata Based Search" \(page 233\)](#)) does not support finding objects based on object tags.

5.8.1.3. Object Metadata Storage in the Metadata DB

Object Metadata and Storage Policies

HyperStore object metadata and object tags are stored in the Metadata DB, and are protected by replication. The degree of replication depends on the type of storage policy being used. For more information see ["Metadata Replication" \(page 153\)](#).

Object Metadata Structure in the Metadata DB

Object metadata is stored in two different types of records (rows) in the Metadata DB:

- One record for each object (called "skinny row" object metadata because along with the key the row has just one column, for that one object)
- For each bucket, multiple records that in sum include metadata for all the objects in the bucket (called "wide row" object metadata because along with the partition keys the rows have potentially very many columns, with one column per object)

These two different types of object metadata records -- object-level records and bucket-level records -- are used for different purposes within HyperStore. For example, while processing an S3 *GetObject* request or *HeadObject* request the system will retrieve the object-level record. While processing a *ListObjects* request for a bucket the system will retrieve bucket-level records. Bucket-level records are also used during batch delete operations and during some *hsstool* operations.

Partitioning of a Bucket's Object Metadata Into Multiple Records

HyperStore's approach to creating and utilizing bucket-level metadata records ("wide row" object metadata) has evolved across HyperStore versions:

- For buckets created in HyperStore 7.0.x and older, for each bucket a single bucket-level metadata record was created. This resulted in *ListObjects* performance issues in certain circumstances, particularly in cases where a bucket contained a very large number of objects and was subject to large numbers of concurrent deletes.
- For buckets created in HyperStore 7.1.x, 7.2.x, or 7.3.x, for each bucket multiple bucket-level metadata records were created, based in a single common algorithm that determined how many records to create per bucket. This improved *ListObjects* performance in most instances, but still resulted in sub-optimal performance in some cases -- such as buckets in which very large numbers of objects were stored in a single "directory" or buckets that had a very large number of directories each of which stored only a small number of objects.
- For buckets created in HyperStore 7.4 and later, for each bucket multiple bucket-level metadata records are created, using varying algorithms that are tailored to the workload type associated with the bucket (such as workloads associated with popular backup applications, each of which produces a distinctive directory structure in its storage target). In HyperStore 7.4 and later, when you create a storage policy you select a "workload type" for the storage policy (for more information see ["Add a Storage Policy" \(page 513\)](#)). And then when buckets are created that use that storage policy, for those buckets HyperStore uses a bucket metadata partitioning scheme that is optimized for that workload type. By using different bucket metadata partitioning schemes for different workload types, HyperStore is able to optimize *ListObjects* performance across a range of bucket workload types and their associated directory structures.

By default, HyperStore's feature that tailors bucket metadata partitioning to a bucket's workload type works **only for buckets created in HyperStore 7.4 and later**. If you have a bucket created in HyperStore 7.3.x or earlier

that is experiencing poor S3 *ListObjects* performance, contact Cloudian Support. Cloudian Support can work with you to upgrade that bucket to use the optimized bucket metadata partitioning feature.

5.8.1.4. Cross-Region Replication and Auto-Tiering of Object Metadata and Tags

When the [cross-region replication](#) feature replicates objects from a local source bucket to a bucket in a remote HyperStore region or system -- or a different S3 compliant destination system -- user-defined metadata (in the form of *x-amz-meta-** headers) and object tags are replicated along with the object data.

When the [auto-tiering](#) feature transitions objects from a HyperStore bucket to an external destination system, user-defined metadata (in the form of *x-amz-meta-** headers) and object tags are sent along with the object data if the tiering destination is a native S3 system such as Amazon Web Services. Also, the user-defined object metadata and tags associated with auto-tiered objects continue to be retained in the local HyperStore system as well (and if the HyperStore bucket uses a [search-enabled storage policy](#), object metadata is also retained in the HyperStore Search Service).

If the tiering destination is not S3-compliant, such as Microsoft Azure or Spectra Pearl Logic, then the user-defined object metadata and object tags are **not** sent along with the object data. The user-defined object metadata and object tags will exist only in the HyperStore system, even after the corresponding object data is transitioned to the destination system.

5.8.1.5. Object Metadata Based Search

A HyperStore system can optionally support the ability to search for objects based on the objects' metadata. This metadata search capability is provided by the HyperStore Search Service, an optional HyperStore system component. The HyperStore Search Service is designed to run on a single "auxiliary" node that is closely integrated with your HyperStore cluster, but that has lesser resource requirements than the rest of your HyperStore nodes. After you've installed a HyperStore cluster you can optionally install the HyperStore Search Service if you wish.

Once the HyperStore Search Service is installed and running, you can enable metadata search on a per storage policy basis. Subsequently, for each new bucket that uses a search-enabled storage policy, HyperStore will trigger the creation of an object metadata index in the HyperStore Search Service. As the population of objects in a bucket changes -- as objects are uploaded, overwritten, or deleted -- HyperStore will update the bucket's object metadata index in the HyperStore Search Service. The creation and updating of object metadata indexes in the HyperStore Search Service is triggered by an hourly cron job that runs on the HyperStore cron job host. The cron job also triggers the deletion of an index if the corresponding bucket is deleted from the HyperStore system.

You can monitor the status of your HyperStore Search node in the CMC's **Data Centers** page, and optionally you can use the **Alert Rules** page to configure alerts based on HyperStore Search Service status.

An end user who owns a bucket that uses a search-enabled storage policy can use the CMC's **Search** page to search for objects in that bucket based on any of the following object metadata:

- Object name string
- Object size
- Object creation date
- User-defined object metadata (metadata set in *x-amz-meta-** headers during object upload). The CMC cannot be used to create such object metadata, so users would have to create it through a third party S3 client application. But users can use the CMC's **Search** page to search for objects based on this metadata.

In regard to HyperStore's transmission of object metadata to the HyperStore Search Service, note that:

- All object metadata continues to exist [in the HyperStore Metadata DB](#), with the durability protections it affords through replication across the HyperStore cluster. When object metadata is copied to the HyperStore Search node for indexing, it resides on the HyperStore Search node **in addition** to residing in the HyperStore Metadata DB.
- As activity within a bucket occurs -- such as object uploads, downloads, or deletions -- the bucket's object metadata in the HyperStore Search Service is **not updated in real time**. Instead, the updates are executed by the hourly cron job. When a user uploads a new object, there will be a delay before the object becomes findable by a metadata search (the delay until the next run of the hourly cron job).

Note also that the HyperStore Search Service only supports finding objects by reference to their metadata (metadata search):

- It does not support finding objects by reference to their [object tags](#)
- It does not support searching through the content of objects (content search)

For more information on setting up the HyperStore Search Service, see ["Preparing the Metadata Search Feature"](#) (page 236).

5.8.2. Creating and Retrieving User-Defined Object Metadata and Tags

Subjects covered in this section:

- ["Creating User-Defined Object Metadata"](#) (page 234)
- ["Retrieving Object Metadata"](#) (page 235)
- ["Creating Object Tags"](#) (page 235)
- ["Retrieving Object Tags"](#) (page 235)

Note The CMC's built-in S3 client does not support creating or retrieving user-defined object metadata or object tags. To create or retrieve user-defined object metadata or object tags **you will need to use a third party S3 client application**.

5.8.2.1. Creating User-Defined Object Metadata

When uploading a new object, S3 client applications can create user-defined object metadata by including one or more *x-amz-meta-** request headers along with the *PutObject* request. For example, *x-amz-meta-topic: merger* or *x-amz-meta-status: draft*.

The S3 API method *POST Object* — for uploading objects via HTML forms — also allows for the specification of user-defined metadata (through *x-amz-meta-** form fields), and HyperStore supports this method as well.

Once uploaded along with an object, user-defined object metadata in the form of *x-amz-meta-** headers cannot be changed. The only way to associate different user-defined metadata with an object that has already been uploaded is to create a new copy of the object through the S3 *CopyObject* operation, and in that operation specify the desired *x-amz-meta-** headers.

For HyperStore support of these S3 API operations, see the S3 section of the [Cloudian HyperStore AWS APIs Support Reference](#).

5.8.2.2. Retrieving Object Metadata

S3 client applications can retrieve user-defined *x-amz-meta-** based object metadata -- as well as system-defined object metadata -- by using either of two standard S3 API methods, both of which the HyperStore system supports:

- *GetObject* returns this type of object metadata as response headers, as well as returning the object itself.
- *HeadObject* returns this type of object metadata as response headers, without returning the object itself.

For HyperStore support of these S3 API operations, see the S3 section of the ***Cloudian HyperStore AWS APIs Support Reference***.

5.8.2.3. Creating Object Tags

When uploading a new object, S3 client applications can create one or more user-defined object tags by including a single *x-amz-tagging* request header in the *PutObject* request. For example, *x-amz-tagging: team-m=Marketing&project=SEO&status=Needs-Review*.

S3 clients can also create object tags when calling the *POST Object* method, by using the *tagging* form field.

The *CopyObject* method supports copying or replacing an object's tags as it is copied, by making use of the *x-amz-tagging-directive* and *x-amz-tagging* request headers.

S3 clients can also create or update object tags for an object that is already stored in the HyperStore system, by using the S3 API method *PutObjectTagging*. This method creates the object tags as a sub-resource of the object, and allows a maximum of 10 key-value pairs.

For a high-level view of object tagging usage and methods, in the Amazon S3 online documentation see [Object Tagging](#).

For HyperStore support of these S3 API operations, see the S3 section of the ***Cloudian HyperStore AWS APIs Support Reference***.

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 – 10
- Maximum key length – 128 Unicode characters in UTF-8
- Maximum value length – 256 Unicode characters in UTF-8

5.8.2.4. Retrieving Object Tags

The S3 methods *GetObject* and *HeadObject* do not return object tags. They do however return a count of the object tags associated with the object (if any), in an *x-amz-tagging-count* response header.

To retrieve the object tags associated with an object, use the S3 API method *GetObjectTagging*.

For a high-level view of object tagging usage and methods, in the Amazon S3 online documentation see [Object Tagging](#).

For HyperStore support of these S3 API operations, see the S3 section of the ***Cloudian HyperStore AWS APIs Support Reference***.

5.8.3. Preparing the Metadata Search Feature

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Installing the HyperStore Search Service" \(page 236\)](#)
- ["Configuring Alerts Based on HyperStore Search Service Status" \(page 236\)](#)
- ["Enabling Metadata Search For a Storage Policy" \(page 236\)](#)
- ["Triggering the Indexing Of Already-Existing Buckets" \(page 237\)](#)
- ["How End Users Access the Search Feature" \(page 237\)](#)

This section outlines at a high level how to prepare the object metadata search feature so that it is available to your end users. For some preparation tasks, this section provides just an overview of the task, along with a reference to more detailed instructions elsewhere in the HyperStore documentation.

Note For an introduction to the metadata search feature, see ["Object Metadata Based Search" \(page 233\)](#).

5.8.3.1. Installing the HyperStore Search Service

The HyperStore Search Service is an optional HyperStore component that you can add to an existing HyperStore system so that the system will support object metadata search. If you are installing a new HyperStore system and you know from the outset that you want to use the metadata search feature, you will complete the regular HyperStore installation first, and verify that the system is operating properly, and then add the HyperStore Search Service as a separate procedure. If you are already using a HyperStore system in production, you can add the HyperStore Search Service to the system at any time without taking the system offline.

The HyperStore Search Service is designed to run on a dedicated virtual machine (VM) -- not on the hosts on which you run the rest of your HyperStore services.

For installation instructions, in the *Cloudbian HyperStore Installation Guide* see "Installing the HyperStore Search Service".

5.8.3.2. Configuring Alerts Based on HyperStore Search Service Status

You can monitor the status of your HyperStore Search node in the CMC's **Data Centers** page, and optionally you can use the **Alert Rules** page to configure alerts based on HyperStore Search Service status. Alerts based on HyperStore Search Service status are not enabled by default.

For more information about status monitoring see ["Data Centers" \(page 466\)](#). For more information about setting alert rules see ["Alert Rules" \(page 560\)](#).

5.8.3.3. Enabling Metadata Search For a Storage Policy

Once the HyperStore Search Service is installed, you can use the CMC's **Storage Policies** page to enable metadata search for individual storage policies. You can enable metadata search for a new storage policy as you create the storage policy, or you can edit any existing storage policy to enable metadata search for it.

Note that:

- The "Enable Metadata Search" option for individual storage policies will display in the CMC **Storage Policies** page only after you've installed the HyperStore Search Service. If you have not installed the HyperStore Search Service, no such option will display in the CMC.
- Even after you've installed the HyperStore Search Service, **by default metadata search will be disabled on each storage policy**. The system does not support a single setting to enable metadata search for all of your storage policies. Instead you must explicitly enable metadata search for each individual storage policy that you want to support metadata search.
- Metadata search cannot be enabled for buckets that are not using a storage policy that supports metadata search.

For more information about creating and editing storage policies see ["Storage Policies"](#) (page 512).

IMPORTANT ! Once you enable search for a storage policy, then **from that time forward** the system will automatically index all **newly created buckets** that use that storage policy (this is accomplished by an hourly system cron job). The system will not automatically index pre-existing buckets that were already using the storage policy before you enabled search for the storage policy. To index buckets that already existed when you enabled search for the storage policy you must manually trigger the indexing as described in ["Triggering the Indexing Of Already-Existing Buckets"](#) (page 237) below.

5.8.3.4. Triggering the Indexing Of Already-Existing Buckets

If you enable metadata search on an already existing storage policy, from that time forward any **newly created** buckets that use that storage policy will be automatically indexed by the HyperStore Search Service. However, if there are buckets that were already using that storage policy before you enabled metadata search for it, the initial indexing of those pre-existing buckets is not automatic. Instead, you must manually trigger the initial indexing of each such bucket. To do, on the HyperStore Config Controller node run the following *hsctl* command once for each such bucket:

```
# hsctl search sync bucket <bucketname>
```

For example:

```
# hsctl search sync bucket engineering.bucket1
```

Once you've done this for each pre-existing bucket, from that time forward the indexing of each such bucket will be kept up to date by the hourly Search sync cron job.

5.8.3.5. How End Users Access the Search Feature

In the CMC, a user who owns a bucket that uses a search-enabled storage policy can use the **Search** page to search for objects based on any of the following object metadata:

- Object name string
- Object size
- Object creation date
- User-defined object metadata (metadata set in *x-amz-meta-** headers during object upload).

Note As stated in ["Creating and Retrieving User-Defined Object Metadata and Tags"](#) (page 234), user-defined object metadata cannot be created in the CMC. A third party S3 client applic-

ation must be used to create such metadata. The CMC **Search** page does support searches based on such metadata.

In the **Search** page a user will first select the bucket in which they want to search, and then enter the search criteria in a visual query editor (an option to enter queries in a JSON editor is also presented).

A CMC user can search only in buckets that they own, and not in any other users' buckets.

If none of a user's buckets use a storage policy for which you've enabled metadata search, and the user tries to use the CMC's **Search** page, a message will display indicating that the user cannot use the **Search** page because none of their buckets use a search-enabled storage policy.

Note also that:

- Search is not supported for buckets that have bucket names exceeding 50 characters.
- Search is not supported for objects that have object names (including full path) exceeding 200 characters.

For system administrators, note that:

- The CMC **Search** page does not display if you are logged into the CMC as a system administrator. The **Search** page displays only for persons logged in as a group administrator or a regular user.
- Even if you have the configuration setting "[cmc.ui.admin.manageUsers.view.objectDataGrantees](#)" (page 739) set to allow system admins to view user data, a system admin cannot access the CMC's **Search** page to perform a metadata search on a user's data.

For Search Service related operations see "[Search Service Operations](#)" (page 653).

5.9. File Services (SMB/NFS)

5.9.1. File Services Feature Overview

Subjects covered in this section:

- *Introduction (immediately below)*
- "[File Service Pools](#)" (page 239)
- "[Geo-Distribution](#)" (page 239)
- "[Bi-Modal Access](#)" (page 240)
- "[WORM \(Object Lock\)](#)" (page 240)

A HyperStore system can optionally support Server Message Block (SMB) and Network File System (NFS) based file services. This SMB and NFS support is provided by the HyperStore File Service, an optional HyperStore system component. For information about HyperStore File Service pricing, contact your Cloudbian representative.

The basic file services delivery model is that the HyperStore File Service runs on dedicated nodes and provides SMB and NFS protocol support to client applications. The HyperStore File Service uses configurable caching to store some file data locally on the auxiliary nodes while also archiving all file data to the HyperStore object storage cluster for highly durable and scalable storage. Each SMB or NFS file share uses its own corresponding HyperStore bucket as its storage back-end. The CMC provides for unified management of the File Service feature within the context of your HyperStore system. You can configure and monitor the HyperStore

File Service and its functionality through the CMC just as you do for the rest for your HyperStore services and features.

Once you've set up File Service functionality in your HyperStore system, you can select HyperStore users for whom the File Services feature will be enabled. Those HyperStore users will be able to use the CMC to create and configure SMB and/or NFS file shares, for access by File Service end users. The HyperStore users for whom you enable the File Service feature -- HyperStore users who will be able to create file shares -- can be group administrators or regular users. System administrator accounts cannot create file shares.

For SMB and NFS versions, the HyperStore File Service supports only:

- SMB versions 2 and 3
- NFS versions 3 and 4

For information about how to set up File Services for your HyperStore system, and how to enable File Services for selected users, see ["Preparing the File Services Feature" \(page 241\)](#).

5.9.1.1. File Service Pools

When you add the File Service to your HyperStore system, you can deploy one or more **File Service pools**. Each File Service pool consists of either one File Service node or three File Service nodes, residing in a single data center. One node may be sufficient for a File Service pool that will be used only for development, testing, or evaluation purposes. One node may also be sufficient for production environments that will have only a small number of SMB or NFS clients and that do not require high-availability file services. Three nodes are required for a File Service pool that will be used in a production environment that has many concurrent SMB or NFS clients, or that needs high-availability file services.

If you wish you can deploy multiple File Service pools, in different locations. For example if your HyperStore system encompasses multiple data centers you could deploy a File Service pool in each data center. Each File Service pool will consist of either one node or three nodes. When you grant File Service privileges to a HyperStore user you can choose which File Service pool(s) the user will have access to. If you grant a HyperStore user access to multiple File Service pools, the HyperStore user can then publish different file shares to different File Service pools (and thereby to different sets of end users), or optionally the HyperStore user can publish a [geo-distributed file share](#) to multiple File Service pools.

5.9.1.2. Geo-Distribution

The HyperStore File Service supports "geo-distribution". A geo-distributed file share is published to two or more [File Service pools](#), typically in different locations and servicing different end users.

For example a HyperStore user could publish a file share to two different pools named "DC1 File Service" and "DC2 File Service", with one bucket serving as the back-end storage for the geo-distributed file share. When end users of the "DC1 File Service" pool create, update, or delete a file, that change becomes visible not only to "DC1 File Service" end users but also to "DC2 File Service" end users. Likewise, when end users of the "DC2 File Service" pool create, update, or delete a file, that change becomes visible not only to "DC2 File Service" end users but also to "DC File Service" end users.

This synchronization of the file share view across different File Service pools is rapid but not instantaneous. There will be a brief delay before a new change is tiered from the originating File Service pool to the back-end object storage bucket (typically 20 seconds but may vary depending on how administrators have configured the system); and another brief delay as the file share metadata synchronization to the other File Service pool is processed. The file share views from the two different File Service pools are "eventually consistent".

Note In the uncommon event of near-simultaneous conflicting changes made by users of the different File Service pools -- for example a user of "DC1 File Service" and a user of "DC2 File Service" make changes to the same file, nearly simultaneously -- the latest version tiered to the back-end object storage bucket "wins" and becomes the current version visible to users of both File Service pools. Meanwhile the version that "lost", by having been superseded by the slightly later winning version, still exists as an object version in the bucket (since versioning must always be enabled on a bucket that serves as the back-end storage for a file share).

Typically a geo-distributed file share would be published to multiple locations **within** a HyperStore service region (for example, to File Service pools in two different data centers within a service region). However, HyperStore does also support a mechanism for geo-distribution of share content to File Service pools in two different service regions.

Though system administrators cannot create and manage shares, you can see ["Add a File Share"](#) (page 368) for information about how HyperStore users can create a geo-distributed share. This includes information about how shares can be geo-distributed across two service regions if desired.

5.9.1.3. Bi-Modal Access

The HyperStore File Service supports "bi-modal access". With bi-modal access, the content of a file share can be accessed and modified by SMB/NFS application users **and also** by S3 application users. SMB/NFS users read and write to the file share via the SMB/NFS front-end presented by the HyperStore File Service, while S3 users read and write directly to the file share's back-end object storage bucket, using the S3 protocol. SMB/NFS users can access content created or changed by S3 users, and vice versa.

HyperStore users for whom you have enabled File Services functionality can enable bi-modal access for a file share after they create the share. By default, when a HyperStore user creates a share bi-modal access is disabled for the share (with the exception of [geo-distributed shares](#) for which bi-modal access is required and enabled by default.)

Though system administrators cannot create and manage shares, you can see ["Set File Share Custom Properties"](#) (page 373) for information about how HyperStore users can enable bi-modal access on a share that they have created.

Note When bi-modal access is disabled on a share, an S3 user with access to the share's back-end bucket can list and read content created by SMB/NFS users of the share. But if the S3 user makes any changes to content in the bucket -- such as by adding an object to the bucket via the S3 protocol -- SMB/NFS users of the share will not see that change (the change will not be indexed into the file share's database in the HyperStore File Service). If it is expected that one or more S3 users will be making changes directly to the content of a bucket that serves as the back-end for a file share, the HyperStore user who manages that file share should enable bi-modal access on the share.

5.9.1.4. WORM (Object Lock)

HyperStore users for whom you have enabled File Services can, if they wish, create a file share that uses a bucket on which [Object Lock](#) is enabled. The result will be a file share that operates in WORM mode (Write Once Read Many). It is important to note that **while object data storage in the share's back-end bucket will comply fully with U.S. SEC-17a and comparable regulatory regimes** (if your system has a license for [Certified Object Lock](#)), **the share's file data stored in the File Services front-end cache will not**. The cached file

data will be managed according to general WORM principles, but the cache implementation does not meet all of the stringent requirements of regulations such as U.S. SEC-17a.

The main behaviors of a WORM file share backed by an Object Lock enabled bucket are:

- Files created through the File Services interface (SMB or NFS front end) will automatically take on the retention properties defined in the bucket's default Object Lock configuration. For example if the back-end bucket's default Object Lock configuration specifies 30 days retention then each new file created through the File Services interface will have WORM protection for 30 days. Also, as a file is archived into object storage, the resulting object will have the appropriate S3 Object Lock headers indicating not only the retention date but also the S3 Object Lock "mode" (Governance mode or Compliance mode; the mode is relevant only for direct S3 access to the archived object and does not impact how the file is managed by the File Services front end).
- When a file is being created in a WORM file share, the file moves into WORM mode after 40 seconds of inactivity by the client application (40 seconds with no change to the file's *mtime*).
- During its retention period, a WORM protected file cannot be deleted or modified (including its metadata) through the File Services interface. After the file's retention period ends, the file can be deleted through the File Services interface but cannot be modified through the File Services interface.
- File retention attributes cannot be modified through the File Services interface. Any and all retention related configuration occurs exclusively on the S3 side.
- 'Legal Hold' is not supported at the file system level.
- The File Service detects and loads a bucket's default Object Lock configuration only when a share is being opened. If a user wants to change the default Object Lock configuration on a bucket to which a file share is attached, the user must close the share, make their change to the bucket's default Object Lock configuration, then re-open the share.
- If [bi-modal access](#) is enabled on the file share (as well as Object Lock being enabled on the bucket):
 - Objects created in the bucket directly via the S3 interface will become available through the file share and will have whatever retention attributes they were assigned through the S3 interface. The retention attributes could come from the bucket's default Object Lock configuration or from object-specific Object Lock attributes specified in the S3 upload operation for the object.
 - Although a WORM protected file in a share cannot be modified or deleted through the File Services interface during the file's retention period, the **file could be overwritten or deleted as a result of S3 activity** that gets synced to the file system. Suppose a WORM protected file is still within its retention period, and its corresponding archived object resides in the Object Lock enabled bucket. Through the S3 interface, a new version of the object could be uploaded to the bucket (while the original, locked object version is retained since versioning is enabled on the bucket); and the new "version" of the object could even be a delete marker. Because the bi-modal access feature automatically keeps the file share in sync with the back-end bucket contents, the file in the share will be updated to match the latest object version in the bucket -- including that the file will be deleted from the share if the latest object version is a delete marker. If users want stricter WORM protection of files in the file share they should not enable bi-modal access on the file share.

5.9.2. Preparing the File Services Feature

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Installing the HyperStore File Service" \(page 242\)](#)
- ["Configuring Alerts Based on HyperStore File Service Status" \(page 242\)](#)
- ["Configuring System Level Settings for File Services" \(page 242\)](#)
- ["Enabling and Configuring File Services for Selected HyperStore Users" \(page 243\)](#)
- ["Disabling File Services for a HyperStore User" \(page 245\)](#)
- ["Deleting a HyperStore User Who Has File Services Enabled" \(page 245\)](#)

This section outlines at a high level how to prepare the file services feature so that it is available to your end users. For some preparation tasks, this section provides just an overview of the task, along with a reference to more detailed instructions elsewhere in the HyperStore documentation.

Note For an introduction to the file services feature, see ["File Services Feature Overview" \(page 238\)](#).

5.9.2.1. Installing the HyperStore File Service

The HyperStore File Service is an optional HyperStore component that can be added to an existing HyperStore system so that the system will support SMB and NFS file services. For more information see [Installing the HyperStore File Service](#). For more information see "Installing the HyperStore File Service" in the *Cloudian HyperStore Installation Guide*.

5.9.2.2. Configuring Alerts Based on HyperStore File Service Status

You can monitor the status of your HyperStore File nodes in the CMC's **Data Centers** page, and optionally you can use the **Alert Rules** page to configure alerts based on HyperStore File Service status. Alerts based on HyperStore File Service status are not enabled by default.

For details about status monitoring see ["Data Centers" \(page 466\)](#). For details about setting alert rules see ["Alert Rules" \(page 560\)](#).

5.9.2.3. Configuring System Level Settings for File Services

After the HyperStore File Service has been installed, you can configure system level settings for file services in the CMC page **Cluster -> Cluster Config -> File Service Settings**. Broadly the system level file service settings are of two types:

- Settings that can only be set at the system level. These are:
 - The username and password that the CMC will initially use to connect to the HyperStore File Service administrative API.
 - The HyperStore File Service caching "waterlevel", which determines how much file data will be cached locally on the HyperStore File Service nodes.
- Settings for which you can set default values at the system level, but which can also be customized at the user level (for individual users for whom you enable file services functionality) and, for some settings, at the file share level (for individual file shares). These are:
 - Settings pertaining to how file data is moved to and read back from back-end object storage buckets

- SMB settings
- NFS settings
- DNS settings for the HyperStore File Service to use when resolving domains on its front end (such as Active Directory domains)

The first time you access the CMC's **File Service Settings** page you will need to connect the CMC to your HyperStore File Service by opening the **File Connection** section of the page. In the **File Connection** interface enter *admin* as the user name and *public* as the initial password. Then click **Connect**. The interface will then require you to change the password to one of your choosing:

- Enter the current password (*public*) and your chosen new password
- The new password must:
 - Be at least 9 characters long and no more than 36 characters
 - Include at least one upper case letter, one number, and one special character

After connecting to the HyperStore File Service and resetting the admin password, you can review and customize the file services configuration settings in the **File Service Settings** tab.

For details about these settings, see ["File Services Configuration Setting Matrix"](#) (page 384).

IMPORTANT ! Although the CMC does not prevent having both SMB and NFS enabled for the same file share, the **HyperStore File Service does not support multi-protocol write access to the same file share** -- and potential conflicts or corruptions may occur if the same file share is concurrently written to by SMB clients and NFS clients. Therefore, if both SMB and NFS are enabled on the same file share, a safe access policy must be implemented such as SMB write access, NFS read-only.

5.9.2.4. Enabling and Configuring File Services for Selected HyperStore Users

Topics in this sub-section:

- ["What a File Services Enabled HyperStore User Can Do"](#) (page 243)
- ["Enabling File Services for a HyperStore User"](#) (page 244)
- ["Configuring Default File Services Settings for a HyperStore User"](#) (page 245)

What a File Services Enabled HyperStore User Can Do

In the HyperStore file services delivery model, only a HyperStore user with S3 access can create and manage SMB and NFS file shares, and each file share will use one of the HyperStore user's buckets for its back-end storage. HyperStore system administrator accounts do not have S3 access and therefore **as a system administrator you cannot create and manage file shares**.

For file shares to be created, you must carefully choose HyperStore group administrators or HyperStore regular users for whom to enable file services. Those group admins or regular users will then be able to use the CMC to create and configure file shares, and their buckets will be used as the storage repositories for those file shares. Because persons for whom you enable file services will be able to create and configure SMB and NFS file shares, you should enable this capability only for persons who have experience with managing SMB and NFS file shares. (Note that although you as a system administrator can configure suitable defaults for most file services settings, the CMC will allow file services enabled users to customize those settings for their own file shares.)

Alternatively, you could create a HyperStore group administrator or regular user account for yourself to use, and use that account to create buckets and then to create and configure file shares backed by those buckets. Note though that with this approach, the S3 service usage associated with this activity (particularly, storage consumption by the buckets) will be attributed to this user account.

When you enable file services functionality for a HyperStore user, that user will be able to do the following through the CMC:

- Configure 'global' settings to apply across all of their own file shares.
- Create one or more file shares, with each file share corresponding to one of the user's own buckets.
- Configure settings for each file share.
- Open and close file shares.
- Monitor file share usage (basic statistics such as number of files and size of files).
- Delete file shares.

To perform these tasks the user will work in the CMC's **Buckets & Objects** section, using interface functions that display only for users for whom you've enabled file services. Users can click **Help** while in the CMC's **Buckets & Objects** section for detailed descriptions of these tasks. (Though system administrators cannot perform these tasks, you can see ["File Share"](#) (page 365) for information about how these tasks can be performed by users.)

Note Although versioning must be enabled on buckets that serve as archives for file shares, non-current objects cannot be accessed through the file services front-end (SMB or NFS).

Note If you have the CMC ["View user data"](#) function enabled in your system, this does not give you visibility into or control over a user's file shares. It only gives you access to users' S3 buckets. **You as a system administrator cannot create, view, edit, open/close, or delete a user's file shares through the CMC.**

Enabling File Services for a HyperStore User

Enabling file services functionality for a HyperStore user can only be done through the CMC. You cannot enable file services for HyperStore users through the Admin API.

To enable file services functionality for a HyperStore user in the CMC:

- For a new user -- As you create the user in the **Add New User** panel set "File Services" to Enabled. If you've established more than one [File Services pool](#) in your system, select the pool(s) to which you want to allow the user to publish file shares.
- For an existing user -- Retrieve the user, select **Edit** for the user, then in the **Edit User** panel set "File Services" to Enabled (and select File Service pool[s] if applicable) and then **Save** the change.

For details see ["Add a User"](#) (page 401) and ["Edit or Suspend a User"](#) (page 407).

Note Group administrators cannot enable file services for a user. Only system administrators can enable file services for a user.

Configuring Default File Services Settings for a HyperStore User

Once you've enabled file services functionality for a HyperStore user, when you retrieve that user in the CMC you will see a **File Setting** option among your controls for that user. Clicking **File Setting** will open a panel where you can set default file services setting values for that user. These are mostly the same settings for which you configured system level defaults, but here you can assign custom values for this specific user (values which will override the system defaults, just for this user). Here you can also configure Active Directory settings for authenticating end users of this HyperStore user's file shares. Active Directory settings are not configurable at the system default level -- they are only configurable at the level of specific HyperStore users for whom file services are enabled.

Note that a user for whom you've enabled file services will be able to set customized configuration settings for their file shares if they wish, overriding the default settings that you establish for the user.

For details about these settings, see "[File Services Configuration Setting Matrix](#)" (page 384).

5.9.2.5. Disabling File Services for a HyperStore User

If you've enabled file services for a user, and then at a later point in time you wish to disable file services for that user, you can do so through the CMC's [Edit User](#) interface. In the **Edit User** page, change the user's "File Services" setting from Enabled to Disabled and then **Save** the change.

Note If the user for whom you want to disable file services **still has file shares**, the user will need to delete those file shares before you can disable file services for the user. The CMC will not allow you to disable file services for a user who still has file shares. In a scenario where an uncooperative user will not delete their file shares, contact Cloudian Support for information about how you can delete the file shares on your own. (You as a system administrator cannot delete a user's file shares through the CMC, but Cloudian Support can guide you through "back door" methods for doing so.)

5.9.2.6. Deleting a HyperStore User Who Has File Services Enabled

You can delete a HyperStore user through the CMC's **Manage Users** page (or through the Admin API's *DELETE /user* call).

When you delete a user who has no buckets, it does not matter if the user has file services enabled -- the user will be deleted from the system just like any other user you would delete.

If you want to delete a user who still owns buckets, then the system behavior depends on your configuration:

- If the configuration setting "[common.user.allowDeleteWithBuckets](#)" (page 754) is set to *true*, then a user who owns buckets can be deleted and the system will not only delete the user but will also automatically delete the user's buckets and all the data in those buckets. If file services are enabled for the user and the user owns file shares, the system will also delete the file share data and metadata from the HyperStore File node(s).

Note If the user owns **file shares that are still Open**, then the system will not be able to delete those file shares and you will not be able to delete the user. Have the user close their file shares before you delete the user. If the user is uncooperative or has departed your organization, contact Cloudian Support for information about how you can close the shares. (You as a system

administrator cannot close a user's file shares through the CMC, but Cloudian Support can guide you through "back door" methods for doing so.)

- If the configuration setting "[common.user.allowDeleteWithBuckets](#)" (page 754) is set to *false*, you cannot delete a user who owns buckets. If the user owns buckets and corresponding file shares, have the user close and delete their file shares, and delete their buckets. Once the user has taken these actions this you can delete the user. If the user has departed your organization or is uncooperative, you could set *common.user.allowDeleteWithBuckets* to *true* (at least temporarily) and then delete the user as described in the bullet point above.

For File Service related operations see "[File Service Operations](#)" (page 647).

5.10. Quality of Service Controls

5.10.1. Quality of Service (QoS) Feature Overview

Subjects covered in this section:

- *Introduction (immediately below)*
- "[Service Usage Types Subject to QoS Controls](#)" (page 246)
- "[QoS Assignment Granularity](#)" (page 247)
- "[Automatic Rate Limiting Per Bucket](#)" (page 248)

The Cloudian HyperStore system supports user-level and group-level Quality of Service (QoS) settings:

- **User QoS settings** place upper limits on service usage by individual users.
- **Group QoS settings** place upper limits on aggregate service usage by entire user groups.

When the QoS feature is enabled, the HyperStore system enforces QoS settings by rejecting S3 requests that would result in a user (or a user's group) exceeding the allowed service usage level.

Note All traffic and storage consumption associated with a bucket is counted as service usage by the bucket owner. This includes access to the bucket by other users to whom the bucket owner has granted permission. It also includes access by public, unauthenticated users if the bucket owner has granted public access to bucket content -- such as in the case of a bucket configured as a static web-site.

Note [Bucket logging](#) is excluded from quality of service controls. Even if QoS limits are reached for a user or group, bucket logging (if enabled for the bucket) can continue.

5.10.1.1. Service Usage Types Subject to QoS Controls

Several types of service usage metrics can be configured for QoS controls:

- Storage quota, by number of KiBs.
- Storage quota, by number of objects.

- Peak HTTP request rate, in requests per minute. The user is not allowed more than this many requests in a 60 second interval.
- Peak data upload rate, in KiBs per minute.
- Peak data download rate, in KiBs per minute.

When configuring QoS controls, you have the option of limiting some of the usage types above while leaving others unrestricted. For example, if you wish you could limit per-user and/or per-group storage volume (by KBs), while placing no restrictions on number of stored objects. Similarly, if you wish you could cap data upload rate while placing no cap on data download rate.

When the system rejects a user request because of a storage quota, it returns an HTTP 403 response to the client application. When the system rejects a user request due to rate controls, it returns an HTTP 503 response to the client application. Any time a user request is rejected due to a QoS threshold, the system writes a WARN level message to the S3 Service application log.

For each type of QoS metric, the system also supports a configurable warning level -- which you can set to a lower threshold of usage than the threshold at which requests will be rejected. If a user's request results in the warning threshold being exceeded, the request will succeed but the system will write a WARN level message to the S3 Service application log. Note that the system does not inform the user that the warning threshold has been exceeded -- it only writes the aforementioned log message.

If you wish you can configure the system so that WARN level messages in the S3 Service application log trigger an alert. (By default configuration, alerts are only triggered by ERROR level messages in the application log.) For more information see ["Preparing the QoS Feature" \(page 248\)](#).

Note The **storage overhead associated with replication or erasure coding does not count toward a user's storage quota**. For example, a 1MiB object that is protected by 3X replication counts as only 1MiB toward the storage quota.

Note For information on how auto-tiering impacts the implementation QoS controls, see ["How Auto-Tiering Impacts Usage Tracking, QoS, and Billing" \(page 208\)](#).

5.10.1.2. QoS Assignment Granularity

The system also provides you the flexibility to assign different QoS settings to different users and groups. In the case of **user QoS settings**, the system provides you three levels of granularity:

- For the system as a whole, you can configure a **system default for user QoS settings**, applicable to all users of your S3 service.
- For particular groups, you can configure a **group-specific default for user QoS settings**. If you do, then for users in that group, the group-specific user QoS defaults will override the system-wide user QoS defaults.
- For particular users, you can configure **user-specific QoS settings**. If you do, these settings will override any group-wide or system-wide defaults for user QoS settings.

For **group QoS** settings you have two levels of granularity:

- For the system as a whole, you can configure a **system default for group QoS settings**, applicable to all groups in your S3 service.

- For particular groups, you can configure **group-specific group QoS settings**. If you do, these settings will override the system-wide defaults.

Note for multi-region systems

If your HyperStore service has multiple service regions, QoS limits are applied separately to each region, and the system provides you the ability to configure different QoS settings for different regions. For example, you could configure default user QoS settings that allow users 20GiB of storage in your "North" service region and 30GiB in your "South" service region.

5.10.1.3. Automatic Rate Limiting Per Bucket

HyperStore has a rate-limiting feature that automatically ensures that **no single bucket places excessive load on the Metadata DB (Cassandra)**. Processing incoming S3 requests entails queries for bucket and object metadata in the Metadata DB -- typically multiple Metadata DB queries per S3 request. The system continually tracks the Metadata DB requests resulting from S3 activity associated with each bucket. Buckets that generate an excessively high load (more than a threshold number of Metadata DB queries per 5 minute interval) over a sustained period of time (five successive 5-minute intervals) will be automatically put into a restricted state. While in a restricted state, the bucket will not be allowed to execute Metadata DB queries, and S3 requests incoming to the bucket will fail with an HTTP 503 error response. This restricted state for the bucket will last 5 minutes and then the bucket will be released from restriction. Any recurrence of sustained excessive Metadata DB load from the bucket will result in the bucket going back into a 5 minute restriction period again.

The per-bucket Metadata DB request threshold is calculated dynamically, based on factors including the number of nodes in your system, the number of disks per node, the disk space usage levels in your system, and the storage policy that a bucket uses (because of this last factor, different buckets may have different thresholds due to using different storage policies). The per-bucket thresholds are updated periodically in response to changes in disk space usage levels in your system.

In the CMC's **Node Advanced** page the "Cassandra QoS Management" commands can be used to check on and control certain aspects of the Metadata DB rate limiting feature. For more information see "[Cassandra QoS Commands](#)" (page 485).

HyperStore also implements a rate limiting feature for **total number of incoming S3 requests for each bucket**. Similar to the Metadata DB rate limiting feature, the per-bucket S3 request threshold is dynamically calculated, taking into account the number of nodes in your system, the number of disks per node, the disk space usage levels in your system, and the storage policy that a bucket uses. Different from the Metadata DB rate limiting feature, the time interval is 1 minute rather than 5 minutes. An excessively high incoming S3 request load (more than a threshold number of request per 1 minute interval) over a sustained period of time (five successive 1-minute intervals) will automatically put the bucket into a restricted state. While in a restricted state, S3 requests incoming to the bucket will fail with an HTTP 503 error response. This restricted state for the bucket will last 1 minute and then the bucket will be released from restriction. Any recurrence of sustained excessive incoming S3 request load into the bucket will result in the bucket going back into a 1 minute restriction period again.

The objective of these per-bucket rate limiting features is to prevent any single bucket from negatively impacting service performance for your user population as a whole.

5.10.2. Preparing the QoS Feature

This section covers the following QoS system set-up topics:

- ["Enabling QoS Enforcement" \(page 249\)](#)
- ["Enabling Alerts for QoS Events" \(page 249\)](#)

5.10.2.1. Enabling QoS Enforcement

By default, the HyperStore system's QoS functionality is **disabled**. Before enabling the functionality, think first about whether you want to implement QoS controls based just on storage quotas, or if you also want to apply QoS controls for request rates (HTTP requests and upload/download bytes). This choice impacts the configuration changes that you will make. In general, for optimal system performance you should enable only the QoS functionality that you actually intend to apply to service users.

Note that in enabling QoS functionality as described below, you are merely "turning on" the S3 Service mechanisms that enforce whatever QoS restrictions you establish for users and groups. The creation of specific QoS restrictions is a separate administrative task as described in ["Applying QoS Controls to Users" \(page 250\)](#).

To enable HyperStore QoS enforcement, in the CMC go to the [Configuration Settings](#) page and open the **Quality of Service** panel. Then:

- To enable enforcement of **storage quotas only** (number of bytes and/or number of objects), set just the "QoS Limits" setting to "enabled".
- To enable enforcement of **storage quotas and also traffic rates** (number of HTTP requests per minute, or bytes uploaded or downloaded per minute), set both the "QoS Limits" setting and the "QoS Rate Limits" setting to "enabled".

After you **Save**, your configuration changes are applied to the system dynamically — no service restart is required.

Note Enforcing QoS for traffic rates but not for stored bytes and objects is not supported at the system configuration level. If you want to use QoS in this way, set both "QoS Limits" and "QoS Rate Limits" to enabled, then when you're configuring QoS limits for groups and users set the stored bytes and stored objects controls to unlimited and the rate controls to your desired levels.

5.10.2.2. Enabling Alerts for QoS Events

If you've enabled QoS enforcement (as described above), then optionally you can also configure the system to generate alerts whenever either of these QoS-related event types occur:

- A user's S3 request has been fulfilled and as a result a user-level or group-level QoS warning threshold has been exceeded.
- A user's S3 request has been rejected because if the request was fulfilled a user-level or group-level QoS upper threshold would be exceeded.

When either of these event types occur the system writes a WARN level message to the S3 application log. By default WARN level messages to the S3 application log do not trigger alerts -- by default only ERROR level messages trigger alerts.

To enable alerting for WARN level messages in the S3 application log, in the CMC go to the ["Alert Rules" \(page 560\)](#) page and create a new alert rule for the Alert Type "S3 service status" and the Condition "Warning".

The screenshot shows the Cloudian web interface for creating an alert rule. The top navigation bar includes links for Analytics, Buckets & Objects, Users & Groups, IAM, Cluster, Alerts (2), and Admin. The 'Alert Rules' tab is active. The 'CREATE ALERT RULE' form has the following fields:

- Alert Type:** A dropdown menu with 'S3 service status' selected.
- Condition:** Radio buttons for 'WARNING' (selected), 'ERROR', 'SERVICE DOWN', and 'SERVICE RESTORED'.
- Target Email:** An empty text input field.
- Severity Level:** A dropdown menu with 'Medium' selected.
- Checkboxes:** 'USE DEFAULT EMAIL ADDRESS' is checked, and 'SEND SNMP TRAP' is unchecked.
- CREATE:** A green button to submit the form.

Once you've created this alert rule, alerts will be generated whenever there are WARN level messages written to the S3 application log (this includes WARN level messages related to QoS as well as other WARN level messages.)

The alert text for QoS threshold related alerts will include the **canonical user ID** of the user whose fulfilled S3 request resulted in a warning level QoS threshold being crossed or whose S3 request was rejected because of an upper level QoS threshold.

To obtain the user name that corresponds to a canonical user ID you can use the Admin API call `GET /user-?canonicalUserId=string`. For more information on this API call see the "user" section of the *Cloudian Hyper-Store Admin API Reference*.

5.10.3. Applying QoS Controls to Users

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Setting System Default QoS Limits for Groups and Users \(CMC\)" \(page 251\)](#)
- ["Setting QoS Limits for Specific Groups and Users \(CMC\)" \(page 251\)](#)
- ["Setting QoS Limits for Groups and Users \(Admin API\)" \(page 251\)](#)

By default, QoS controls for all service usage types — storage quota by bytes, storage quota by number of objects, HTTP requests per minute, upload bytes per minute, and download bytes per minute — are set to "unlimited". So when you [enable QoS enforcement by the system](#), groups and users still are not subject to QoS controls until you set QoS limits for the various service usage types.

You can set system-wide defaults for group QoS limits and individual user QoS limits before provisioning any groups or users, if you wish. After [provisioning groups and users](#) you can set QoS limits specific to those groups and users, which would override the system-wide defaults.

Note All traffic and storage consumption associated with a bucket is counted as service usage by the bucket owner. This includes access to the bucket by other users to whom the bucket owner has granted permission. It also includes access by public, unauthenticated users if the bucket owner has

granted public access to bucket content -- such as in the case of a bucket configured as a static web-site.

5.10.3.1. Setting System Default QoS Limits for Groups and Users (CMC)

HyperStore supports setting system default QoS limits that apply to all groups (default limits on the aggregate service usage per group) and system default QoS limits that apply to all users (default limits on service usage per user). If you wish you can set these system default QoS limits before you've provisioned groups and users into the system. You can also set or change these system default QoS limits at any time after you've provisioned groups and users.

To set default QoS limits for all groups in your HyperStore system go to the CMC's **Manage Groups** page and click **Group QoS Default** to open the **Group QoS Limits: Defaults** panel. In this panel you can set your desired default QoS limits for groups.

To set default QoS limits for all users in your HyperStore system go to the CMC's **Manage Users** page and click **User QoS Default** to open the **User QoS Limits: Defaults** panel. In this panel you can set your desired default QoS limits for users.

For details see ["Set Quality of Service \(QoS\) Limits" \(page 430\)](#).

5.10.3.2. Setting QoS Limits for Specific Groups and Users (CMC)

Once you have provisioned groups and users into your HyperStore system you can set QoS limits for specific groups and users if you wish. If you do so, these group-specific and user-specific QoS limits will override the system default QoS limits, for those particular groups and users.

To set group QoS limits for a specific group, first retrieve the group in the **Manage Groups** page. Then click **Group QoS** for the group. This opens the **Group QoS Limits: Overrides** panel, where you can configure group QoS limits for that specific group.

To set default per-users QoS limits for all users who belong to a specific user group, first retrieve the group in the **Manage Groups** page. Then click **User QoS Group Default** for the group. This opens **User QoS Limits: Group Defaults** panel, where you can configure default user QoS limits for the group.

To set QoS limits for a specific individual user, first retrieve the user in the **Manage Users** page, then click **Set QoS** for the user. This opens the **User QoS Limits: Overrides** panel, where you can configure QoS limits for that specific user.

For details see ["Set Quality of Service \(QoS\) Limits" \(page 430\)](#).

5.10.3.3. Setting QoS Limits for Groups and Users (Admin API)

With the Admin API method *POST /qos/limits* you can set QoS limits for HyperStore groups and users. With the method's URI parameters you can specify whether you're setting system defaults for group QoS settings; or system defaults for user QoS settings; or QoS settings for a specific group or user. URI parameters also enable you to set the numerical limits for each service usage type — for example, a Storage Bytes limit of 100GB.

The Admin API also supports:

- A *GET /qos/limits* method, for retrieving QoS limits for groups or users
- A *DELETE /qos/limits* method, for deleting QoS limits for groups or users

For details about these API calls see the "qos" section of the *Cloudian HyperStore Admin API Reference*.

5.11. Usage Reporting

5.11.1. Usage Reporting Feature Overview

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Bucket Usage Statistics" \(page 252\)](#)
- ["How Usage Data is Calculated, Tracked, and Processed" \(page 253\)](#)
- ["Protection of Usage Data Through Replication" \(page 254\)](#)

By default the HyperStore system keeps track of the following service usage metrics for each user group and each individual user:

- Number of Storage Bytes
- Number of Storage Objects

Optionally, you can configure the system to also track the following metrics for each group and user (these metrics are disabled by default):

- Number of HTTP Requests
- Number of Bytes IN (bytes uploaded into the system)
- Number of Bytes OUT (bytes downloaded from the system)

Like Amazon S3, the HyperStore system attributes all service usage to **bucket owners**. If a bucket owner grants permission (via ACL policies) for other users to use the bucket, the system attributes the service activity of the grantees to the bucket owner. For example, if grantees upload objects into the bucket, the associated Bytes IN activity and Storage Bytes impact is attributed to the bucket owner — not to the grantees.

If a HyperStore account root user creates IAM users, then storage activity associated with those IAM users' buckets counts toward the usage of the account root user.

The HyperStore system's tracking of service usage by groups and users serves two main purposes:

- **Usage reporting.** Based on service usage tracking data, you can generate service usage reports for individual users, for user groups, for a whole service region, or for your entire HyperStore system.
- **Billing.** Usage tracking provides the foundation for [billing users or groups](#) on the basis of their service usage level.

Note **Cloudian HyperIQ** is a product that provides advanced analytics and visualization of HyperStore S3 Service usage by users and groups, as well as HyperStore system monitoring and alerting. For more information about HyperIQ contact your Cloudian representative.

5.11.1.1. Bucket Usage Statistics

Optionally, you can also configure the system to track usage statistics on a per-bucket basis. This may be useful if for example your HyperStore service is set up such that there is just single "user" for service access purposes, with that one user having multiple buckets each for a different purpose.

The bucket statistics features are disabled by default. For information on enabling these features see ["Preparing the Usage Reporting Feature"](#) (page 255).

5.11.1.2. How Usage Data is Calculated, Tracked, and Processed

The table below provides a high level view of how the system generates and handles usage data for users and groups. Usage tracking for Storage Bytes and Storage Objects is handled somewhat differently than tracking for HTTP Requests and Bytes IN/OUT, including that the latter is disabled by default.

Usage Data Types	System Handling
- Storage Bytes - Storage Objects	For each S3 request that the S3 Service processes, the S3 Service updates per-user and per-group Storage Bytes and Storage Objects counters that are maintained in the Redis QoS database. In calculating the increment to Storage Bytes associated with a given S3 request, the system includes the object name size and object metadata size as well as the size of the object itself. Note that storage overhead associated with replication or erasure coding does not count toward a user's Storage Bytes count. For example, a 1MB object that is protected by 3X replication or by 4+2 erasure coding counts as only 1MB toward the Storage Bytes count. These per-user and per-group Storage Bytes and Storage Objects counters in Redis are used by the system to enforce storage quotas, if such quotas are part of your Quality of Service configurations. Every five minutes, freshly updated Redis QoS counts for Storage Bytes and Storage Objects are written to the Raw column family in Cassandra's Reports keyspace, where the data is subjected to additional processing in support of reporting and billing functionality. Each hour the Raw data is automatically processed to derive hourly roll-up data which is written to the RollupHour column family. The hourly roll-up data includes, for each user and each group, the hour's maximum value and weighted average value for Storage Bytes and for Storage Objects. For example, if during a given hour User1 has 10MB of Storage Bytes for the first 20 minutes of the hour and then 15MB for the next 40 minutes of the hour, her weighted average Storage Bytes for the hour is: <pre> 10MB X 20/60 = 3.33MB + 15MB X 40/60 = 10 MB = 13.33MB weighted average for hour </pre> This hourly data is in turn rolled up once each day to derive daily roll-up values (including, for each user and group, the day's maximum and day's average for Storage Bytes and Storage Objects); and the daily roll-up values are rolled up once each month to derive monthly roll-up values (including monthly maximums and averages for each user and group). This data is stored in the RollupDay column family and RollupMonth column family, respectively.

Usage Data Types	System Handling
	Note The writing of Storage Bytes and Storage Objects counters from Redis over to Cassandra, and the usage data roll-ups that take place within Cassandra, are triggered by system cron jobs.
• HTTP Requests • Bytes IN • Bytes OUT	By default system configuration, HTTP Requests, Bytes IN, and Bytes OUT are not tracked. If you enable usage tracking for HTTP Requests and Bytes IN/OUT, then for each S3 request the S3 Service writes transactional metadata directly to the Raw column family in Cassandra's Reports keyspace. It does so asynchronously so that S3 request processing latency is not impacted. In recording the Bytes IN and Bytes OUT impacts of a given S3 transaction, both the request and the response are counted, and HTTP header size is counted as well as body size. For example, a GET request/response pair will count as Bytes IN (typically very small) as well as Bytes OUT (varies with object size); and conversely a PUT request/response pair will count as Bytes OUT (typically very small) as well as Bytes IN (varies with object size). Together with the Storage Bytes and Storage Objects data, the HTTP Request and Bytes IN/OUT data in the Raw column family is rolled up each hour. For each user and each group the roll-up calculates the hour's total for HTTP GETs, PUTs, and DELETES, and for Bytes IN and Bytes OUT. For Bytes IN and Bytes OUT, maximums for the hour (largest single upload and largest single download) and averages for the hour (average upload size and average download size) are derived for each user and group. The hourly roll-up data is rolled up daily; and the daily roll-up data is rolled up monthly. If you fully enable HyperStore Quality of Service functionality, then for each S3 request the S3 Service also updates HTTP Request and Bytes IN/OUT counters in the Redis QoS database, in support of QoS enforcement.

5.11.1.3. Protection of Usage Data Through Replication

All HyperStore service usage data is stored in Cassandra, in the Reports keyspace. This data is protected against loss or corruption by maintaining multiple copies of the data, with each copy stored on a different node. During system installation the install script prompts you to specify how many replicas to keep of service metadata, which includes usage data. If you are running HyperStore in multiple data centers, during installation you choose how many service metadata replicas to keep in each data center.

5.11.2. Preparing the Usage Reporting Feature

5.11.2.1. Enabling Additional Usage Reporting Features

By default the HyperStore system keeps track of the number of stored bytes and the number of stored objects for each user group and each individual user. This topic describes how to enable additional HyperStore usage reporting features that are disabled by default:

- Per-user and per-group traffic rates (HTTP request rates and data transfer rates)
- Per-bucket usage tracking
- Per-bucket object and byte counts

Per-User and Per-Group Traffic Rates

By default the HyperStore system tracks Storage Bytes and Storage Objects for each user and each group. If you want the system to also track HTTP Requests, Bytes IN, and Bytes OUT for each user and group, log into the CMC and go to the **Configuration Settings** page, and then open the **Usage Tracking** panel. There, enable the "Track/Report Usage for Request Rates and Data Transfer Rates" setting and **Save** your change. Your change is applied to the system dynamically -- there is no need to restart services.

Once you enable this feature, this type of usage data will be tracked and available for reporting from that point in time forward. There will not be any per-user and per-group traffic rate data from prior to the time that you enabled this feature.

Note Enabling this feature results in additional data being stored in the Metadata DB, and additional work for the cron jobs that roll up usage data into hourly, daily, and monthly aggregates.

Per-Bucket Usage Tracking

HyperStore supports usage tracking and reporting on a per-bucket basis, but this feature is disabled by default. To enable per-bucket usage statistics, set the configuration setting "**s3.usage.bucketLevel**" (page 838) to *true* (by default it is *false*). (For instructions on changing this setting and applying your change see the setting description).

Once you enable this feature, bucket usage data for storage consumption and traffic rates will be tracked and available for reporting from that point in time forward, through the *usage* Admin API calls (for details see the "usage" section of the **Cloudian HyperStore Admin API Reference**). There will not be any per-bucket usage data from prior to the time that you enabled this feature.

Note Enabling this feature results in additional metadata being stored in the Metadata DB, and additional work for the cron jobs that roll up usage data into hourly, daily, and monthly aggregates.

Note Once enabled, the per-bucket usage tracking feature allows a view into bucket activity across a specified period of time. By contrast, the per-bucket object and byte count feature -- described below -- allows just snapshots of current counts.

Per-Bucket Object and Byte Counts

HyperStore supports keeping counts of the number of objects and bytes in each bucket, but this feature is disabled by default. To enable per-bucket object and byte counts, set the configuration setting "[s3.qos.bucketLevel](#)" (page 828) to *true* (by default it is *false*). (For instructions on changing this setting and applying your change see the setting description).

Next, run the Admin API call *POST /usage/repair?groupId=ALL* to bring the counters up to date for buckets that already have objects in them. For details see the "usage" section of the *Cloudian HyperStore Admin API Reference*.

Subsequently, the counters will automatically be updated for each bucket each time there is an S3 transaction that impacts the byte count or object count for the bucket.

With this feature enabled, you can query the current stored-bytes and stored-objects counts for individual buckets by using the Admin API calls *GET /system/bytecount* and *GET /system/objectcount*. For details see the "system" section of the *Cloudian HyperStore Admin API Reference*.

Note In some atypical circumstances the keeping of these per-bucket stored-bytes and stored-object counts, which will be updated after every S3 transaction, may cause a minor-to-moderate decline in S3 write performance. Example circumstances would be if there are an exceptionally large number of buckets in your HyperStore system, or if you have a multi-data center HyperStore deployment with more than usual latency between the data centers. If you want to enable per-bucket stored-bytes and stored-objects counters in your system but are concerned about potential performance impacts, consult with Cloudian Support.

5.11.2.2. Setting Usage Data Retention Periods

Data retention periods are separately configurable for hourly roll-up, daily roll-up, and monthly roll-up usage data. After rolled up data has been stored for its configured retention period -- also known as its "time-to-live" or TTL -- it is automatically deleted from the system.

The relevant settings are:

- "[s3.usage.reports.hourlyRollupTTL](#)" (page 839) (default = 5616000 seconds [65 days])

IMPORTANT ! The *billing* Admin API (and the CMC's "[Account Activity](#)" (page 427) page) calculates monthly bills for service users by aggregating **hourly** roll-up data. Once hourly data is deleted, you will not be able to use the Admin API (or the CMC) to generate bills for the service period covered by that data. So be sure to have *s3.usage.reports.hourlyRollupTTL* set to a value large enough to accommodate your billing routine.

- "[s3.usage.reports.dailyRollupTTL](#)" (page 839) (default = 5616000 seconds [65 days])
- "[s3.usage.reports.monthlyRollupTTL](#)" (page 839) (default = 15552000 seconds [180 days])

5.11.3. Generating a Usage Report

You can generate usage report data through either the CMC or the Admin API.

5.11.3.1. Generating a Usage Report (CMC)

In the CMC's [Usage By Users & Groups](#) page you can generate usage reports for a user, a group, a whole region, or your whole system. You can choose a report interval, a report granularity (raw, hourly, daily, or monthly roll-up), and the usage type to report on (stored bytes or stored object counts, or — if [request tracking is enabled](#) — statistics for HTTP requests and data transfer).

You can display reports in the CMC in tabular format or dynamic graph format, or download report data in comma-separated value (CSV) format.

The CMC does not support per-bucket usage reporting. For that you must use the Admin API.

5.11.3.2. Generating a Usage Report (Admin API)

The Admin API method `GET /usage` returns usage data for a specified user, group, or bucket. You can choose a report interval, a report granularity (raw, hourly, daily, or monthly roll-up), and the usage type to report on (stored bytes or stored object counts, or — if [request tracking is enabled](#) — statistics for HTTP requests and data transfer).

Note To retrieve usage data for a whole region or the whole system, you must execute `GET /usage` separately for each group.

The Admin API also supports:

- `GET system/bytecount` and `GET system/objectcount` calls that return the current total bytes count and total objects count the system, a group, a user, or a bucket (if bucket counts are enabled)
- A `POST /usage/bucket` call that can retrieve raw usage data for multiple specified buckets at once (if bucket usage statistics are enabled)

For more detail on usage related Admin API calls see the "system" and "usage" sections of the **Cloudian HyperStore Admin API Reference**.

5.11.4. Validating Storage Usage Data

As described in ["Usage Reporting Feature Overview" \(page 252\)](#), each time the S3 Service processes an S3 request it updates per-user and per-group counters for Storage Bytes and Storage Objects which are maintained in the Redis QoS database. These counts are regularly written over to the Cassandra Reports key-space, where they are post-processed for reporting and for bill generation.

Because the Redis QoS counters for Storage Bytes and Storage Objects can impact your billing of service users (if you charge users based on volume of storage used), it's important that the counts be accurate.

Various types of errors and conditions can on occasion result in discrepancies between the Redis QoS counts and the actual stored bytes and objects owned by particular users. The HyperStore system provides mechanisms for detecting and correcting such discrepancies.

5.11.4.1. Routine Automated Validation

Twice a day, a [system cron job](#) executes the HyperStore Admin API method `POST /usage/repair/dirtyusers`. This operation randomly selects up to 1000 users for whom Storage Bytes and/or Storage Objects counts in Redis have changed since the last time a validation operation was run. For those users, the operation

validates the Redis counters by comparing them to the information in the Cassandra "UserData_<policyId>" keyspaces' "CLOUDIAN_METADATA" column family, which stores metadata (including size) for every object belonging to each user of the HyperStore service. If any users' Redis counters are found to be out-of-sync with counts derived from the object metadata, the Redis counters are corrected.

In normal circumstances, this automated mechanism should suffice for maintaining the accuracy of usage data for Storage Bytes and Storage Objects.

5.11.4.2. Validation for Special Cases

If you have reason to question the accuracy of Storage Bytes and/or Storage Objects counts for a particular user — for example, if a user claims their usage report or their bill to be inaccurate — the Admin API supports a method for validating the counts for a specified user: *POST /usage/repair/user*.

. The Admin API also supports a method for validating Storage Bytes and Storage Object counts for a whole user group, a whole service region, or the whole system: *POST /usage/repair*. Depending on how many users are in your system, this is potentially a resource-intensive operation. This operation should only be considered in unusual circumstances, such as if the Redis QoS database has been brought back online after being unavailable for some period of time.

For details on these API calls, see the "usage" section of the ***Cloudian HyperStore Admin API Reference***.

5.12. Billing

5.12.1. Billing Feature Overview

Subjects covered in this section:

- *Introduction (immediately below)*
- ***"Retention of Usage Data Used for Billing"*** (page 259)
- ***"Retrieving Bucket Tags"*** (page 259)

The HyperStore system maintains service usage data for each group and each user in the system. This usage data -- which is automatically generated by recurring [cron jobs](#) that call the Admin API -- serves as the foundation for HyperStore service billing functionality.

The system provides you the ability to create rating plans that specify charges for the various types of service usage activity, and to assign each group and each user a rating plan. You can then generate bills for a user or for a whole user group, for a selected service month. The CMC has a function for displaying a single user's bill report in a browser, but in the more typical use case you will use the HyperStore Admin API to generate user or group billing data that can be ingested a third party billing application.

HyperStore also allows for special treatment of designated source IP addresses, so that the billing mechanism does not apply any data transfer charges for data coming from or going to these "allowlisted" domains.

Note For information on how auto-tiering impacts billing calculations, see ***"How Auto-Tiering Impacts Usage Tracking, QoS, and Billing"*** (page 208).

5.12.1.1. Retention of Usage Data Used for Billing

Billing calculation is derived from hourly rollup usage data that is automatically generated by system cron jobs. The retention period for this hourly rollup usage data is configurable and defaults to **65 days**. **Once this rollup data is deleted it can no longer be used to generate users' bills.**

For more information about this configurable setting see ["Setting Usage Data Retention Periods" \(page 256\)](#)

5.12.1.2. Retrieving Bucket Tags

If your billing scheme makes use of bucket tags (as created by the S3 API method *PUT Bucket tagging*): The HyperStore Admin API supports a method for retrieving all the bucket tags for all users in a specified group. Because it is implemented through the Admin API, that method does not require the users' S3 access credentials. For more information see *GET /bucketops/gettags* in the "bucketops" section of the ***Cloudian HyperStore Admin API Reference***.

5.12.2. Preparing the Billing Feature

This section covers the following topics for setting up billing in the system:

- ["Retention of Usage Data Used for Billing" \(page 259\)](#)
- ["Creating Rating Plans" \(page 259\)](#)
- ["Creating an "Allowlist" for Free Traffic \(Optional\)" \(page 261\)](#)

5.12.2.1. Retention of Usage Data Used for Billing

Billing calculation is derived from hourly rollup usage data that is automatically generated by system cron jobs. The retention period for this hourly rollup usage data is configurable and defaults to **65 days**. **Once this rollup data is deleted it can no longer be used to generate users' bills.**

For more information about this configurable setting see ["Setting Usage Data Retention Periods" \(page 256\)](#).

5.12.2.2. Creating Rating Plans

The HyperStore system supports the creation of billing rating plans in which charges can be specified for each of several different service activity types. The system supports rating plans that charge:

- Per GiB of data in storage (based on a calculated average storage volume for the billing period)
- Per GiB of data uploaded
- Per GiB of data downloaded
- Per 10,000 HTTP GET or HEAD requests
- Per 10,000 HTTP PUT or POST requests
- Per 10,000 HTTP DELETE requests

A user's bill can then be calculated by applying the user's assigned rating plan to the user's activity levels for each of these activity types, and adding together the charges for each activity type to get a total charge for the billing period.

You can create multiple, named rating plans, each of which applies different charges to the various service activity types. Once you've created rating plans, those plans are then available for you to [assign to users](#).

For example, you can create higher-priced and lower-priced rating plans and then assign different plans to different users based on the users' quality of service terms.

IMPORTANT ! If you want to bill for data upload or download volume, or for HTTP request volume, you must enable the "Track/Report Usage for Request Rates and Data Transfer Rates" setting in the CMC's **Configuration Settings** page, Usage Tracking section. By default this setting is disabled and the system does not maintain per-user HTTP request counts and data transfer byte counts.

You can create rating plans either through the CMC or through the HyperStore Admin API. The system also comes equipped with an editable default rating plan.

Creating Rating Plans for Billing (CMC)

To create rating plans through the CMC, use the [Rating Plan](#) page. For each plan you create, you can select a currency and then specify the charges to apply per each of the chargeable service activity types (per GiB of stored data, per GiB of data uploaded, and so on). For each rating plan, the interface supports the creation of single-tier or multi-tier pricing schemes for each chargeable activity.

The **Rating Plan** page also supports viewing and editing existing plans, including the [default rating plan](#) that comes with the HyperStore system.

Creating Rating Plans for Billing (Admin API)

To create a rating plan through the Admin API, use the *PUT /ratingPlan* method.

The Admin API also supports:

- Retrieving a list of existing rating plans: *GET /ratingPlan/list*
- Retrieving a rating plan: *GET /ratingPlan*
- Editing a rating plan: *POST /ratingPlan*
- Deleting a rating plan: *DELETE /ratingPlan*

For details of these API calls see the "ratingPlan" section of the *Clouddian HyperStore Admin API Reference*.

Example of a Rating Plan Applied to Calculate a User's Monthly Bill

This example walks through a hypothetical rating plan and how it would apply to a user's service usage for a month.

Storage Charges

- Types: One type only. The average number of GiBs of data stored for the month.
- Example:
 - Unit: Dollars per GiB-months.
 - Pricing: From 0-1 TiB at \$0.14 per GiB-month, 1-10 TiB at \$0.12 per GiB-month, 10+ TiB at \$0.10 per GiB-month.
 - Usage: Store 0.1 TiB for first 10 days of month, then 20 TiB for remaining 21 days of month.
 - Sum up usage over month: 0.1TiB X 240 hours + 20TiB X 504 hours = 10,104 TiB-hours.

- Convert to GiB-months: $10,104 \text{ TiB-hours} \times (1024 \text{ GiB/TiB}) \times (1 \text{ month}/744 \text{ hours}) = 13,906.58 \text{ GiB-months}$
- Apply tiered pricing: $(\$0.14 \times 1 \times 1024 \text{ GiB}) + (\$0.12 \times 9 \times 1024 \text{ GiB}) + (\$0.10 \times 3666.58 \text{ GiB}) = \$1615.94$.

Data Transfer Charges

- Types: 2 types. Data Transfer IN and Data Transfer OUT.
 - Each type (IN, OUT) has own cost.
- Example:
 - Unit: Dollars per GiB.
 - Pricing: IN: 0GiB+ at \$0.10. OUT: 0-10GiB at \$0.20, 10GiB+ at \$0.10
 - Usage: Transfer-IN 300GiB, Transfer-OUT 100GiB.
 - $(300 \times \$0.10) + (10 \times \$0.20 + 90 \times \$0.10) = \$41.00$.

Request Charges

- Types: 3 types of requests are HTTP PUT/POST, HTTP GET/HEAD, and HTTP DELETE.
 - Each request type has own cost.
- Example:
 - Unit: Dollars per 10,000 requests.
 - Pricing: PUT/POST: \$0.20 per 10,000 requests, GET/HEAD: \$0.01 per 10,000 requests, DELETE: \$0.00 per 10,000 requests.
 - Usage: For the month, 25,000 PUTs/POSTs, 300,000 GETs/HEADs, 1000 DELETES.
 - $(\$0.20 \times 25,000 / 10,000) + (\$0.01 \times 300,000 / 10,000) + (\$0.00 \times 1000 / 10,000) = \$0.53$.

Total Bill for Month

- Storage charges: \$1615.94
- Data transfer charges: \$41.00
- Requests charges: \$0.53
- TOTAL: \$1657.47

5.12.2.3. Creating an "Allowlist" for Free Traffic (Optional)

There may be certain source domains from which you want users to be able to submit S3 requests to the HyperStore system without incurring any data transfer charges. The HyperStore system allows you to create such an "allowlist", consisting of IPv4 addresses and/or subnets. For traffic originating from these addresses or subnets, there is no charge for data IN or data OUT, nor is there any charge for HTTP requests — regardless of users' assigned rating plans.

Note that the allowlist does not have any impact on what users are charged for data **storage**. It allows only for free **traffic** from the specified origin domains. For data storage billing, a user's regular assigned rating plan pricing is used, even if all of the user's S3 requests originate from an allowlisted IP address.

You can creating a billing allowlist through either the CMC or the Admin API.

IMPORTANT ! If your S3 Servers are behind a load balancer, the load balancer must be configured to pass through request source IP addresses in order for the allowlist feature to work. Also, note that when S3 requests are submitted via the CMC, the S3 Servers consider the CMC itself to be the source of the request. The CMC does not pass to the S3 Servers the IP addresses of CMC clients.

Enabling the Allowlist Feature

The HyperStore billing allowlist feature is enabled by default. To confirm that it is currently enabled in your system, on the Config Controller node run this command:

```
# hscctl config get admin.billing.allowlist.enabled
```

This should return a value of *true*. If it returns *false*, change the setting to *true*. (For instructions see "[admin.-billing.allowlist.enabled](#)" (page 702)).

Creating a Source IP Allowlist (CMC)

To create an allowlist through the CMC, use the [Allowlist](#) page. That page shows the ID and name of the default allowlist, which initially has no IP addresses in it. To create a functioning allowlist, you edit the default allowlist by adding IP addresses or subnets to it.

Once created, the allowlist takes effect. From that time forward, the HyperStore billing system will no longer apply traffic charges to users' traffic originating from the allowlisted IP addresses and subnets.

Note If you want to see a particular user's allowlisted traffic volume during a given billing period, you can do so as part of the HyperStore system's functionality for .

Creating a Source IP Allowlist (Admin API)

The HyperStore Admin API provides two different methods for creating an allowlist:

- Use the *POST /allowlist* method to post your allowlist as a JSON-encoded request payload.
- Use the *POST /allowlist/list* method to post your allowlist as a URI parameter.

The Admin API also supports a *GET /allowlist* method for retrieving the contents of your current allowlist.

For details on these API calls, see the "allowlist" section of the *Cloudian HyperStore Admin API Reference*

5.12.3. Applying Billing to Users

This section covers the following topics for applying billing to users:

- "[Assigning Rating Plans to Groups and Users](#)" (page 262)
- "[Generating Billing Data for a User or Group](#)" (page 263)

5.12.3.1. Assigning Rating Plans to Groups and Users

When you create a new user group you can assign to the group a [rating plan](#) that by default will apply to each user in the group. Optionally you can assign a rating plan to specific users within the group, overriding the group's default rating plan.

In a multi-region HyperStore system, you have the option of assigning groups or individual users different rating plans for activities in different regions. For example, you might charge users more for stored data in buckets that they've created in your North region than for stored data in buckets created in your South region.

If you do not explicitly assign a rating plan to a user, the user is automatically assigned the rating plan that's assigned to the user's group. If you do not explicitly assign a rating plan to a group, then the system [default rating plan](#) is automatically used for that group.

You can assign rating plans to users through either the CMC or the Admin API.

Assigning Rating Plans to Groups and Users (CMC)

To use the CMC to assign a rating plan to a group, use the [Manage Groups](#) page. From here you can create a new group, and while doing so assign the group a rating plan from a drop-down list of rating plans that exist in the system (the system default plan plus any plans you've created). From here you can also retrieve an existing group and change the group's rating plan assignment. Whichever rating plan you associate with a group will be applied to each user in the group, except for any users to whom you explicitly assign a rating plan.

To use the CMC to assign a rating plan to an individual user, use the [Manage Users](#) page. From here you can create a new user, and while doing so assign the user a rating plan from a drop-down list. By default the new user is assigned whichever plan is assigned to the user's group. From the **Manage Users** page you can also retrieve an existing user and change her rating plan assignment.

Assigning Rating Plans to Groups and Users (Admin API)

To use the HyperStore Admin API to assign a rating plan to a group, you must have first created the group (with the *PUT /group* method). Once a group exists, you can assign the group a rating plan by using the *POST /group/ratingPlanId* method. You could subsequently use this same API method to change a group's rating plan assignment. The Admin API also supports a *GET /group/ratingPlanId* method, for retrieving a group's current rating plan identifier.

The approach is similar if you want to assign a rating plan to an individual user. The user must have already been created (with *PUT /user*), and then you can use *POST /user/ratingPlanId* to assign the user a rating plan (and to subsequently update that assignment). The method *GET /user/ratingPlanId* lets you retrieve the identifier of the rating plan currently assigned to a specified user, and there's also a *GET /user/ratingPlan* method for retrieving the user's rating plan in full.

For details of these API calls, see the "group" and "user" sections of the **Cloudian HyperStore Admin API Reference**.

5.12.3.2. Generating Billing Data for a User or Group

The HyperStore system supports generating a bill for a specified user or for a whole user group. The system applies the user's or group's assigned rating plan to their service usage during the billing period. Bills can be generated for any **completed calendar month** of service usage.

With the CMC you can generate a billing report for an individual user (but not for a whole group). With the Admin API you can generate billing data for a specified user or for a whole user group.

IMPORTANT ! Billing calculation is derived from hourly rollup usage data. The retention period for hourly rollup usage data is configured by the setting **"s3.usage.reports.hourlyRollupTTL"** (page 839). The default retention period is 65 days. Once this rollup data is deleted it can no longer be used to gen-

erate users' bills. So be sure to have `s3.usage.reports.hourlyRollupTTL` set to a value large enough to accommodate your billing routine.

Generating Billing Data for a User (CMC)

To generate billing data through the CMC, use the [Account Activity](#) page. In this page you can specify a user and select a billing period. The most recent period you can select is the most recently completed calendar month. You cannot generate a billing report for an in-progress month.

The billing data that you can generate from this page displays in the form of a printable billing document that includes the user's name and user-ID, their user group, the billing period, and the bill generation date. The document shows a summary of the user's rating plan, the user's service activity for the billing period, and the associated charges.

The **Account Activity** page also provides you an option to view a user's service traffic originating from allowlist domains, if any.

Generating Billing Data for a User or Group (Admin API)

To generate user or group billing data through the HyperStore Admin API, use the `POST /billing` method. This triggers the calculation of the billing data for the specified calendar month, and returns the billing data as a JSON-encoded response payload.

The Admin API method also supports a `GET /billing` method, which simply retrieves billing data that you've previously generated with the `POST /billing` method. Like the `POST /billing` method, the `GET /billing` method returns billing data as a JSON-encoded response payload.

For details on these API calls, see the "billing" section of the *Clodian HyperStore Admin API Reference*.

5.13. Customizing the CMC

5.13.1. Showing/Hiding CMC UI Functions

The CMC provides granular configuration control over which UI functions and sub-functions display for the three types of users that the CMC supports: system admins, group admins, and regular S3 service users. The table below summarizes the CMC's configurability for showing or hiding certain functionality. The third column indicates which user types have access to the functionality by default. The fourth column indicates the configuration setting that controls access to the functionality. You can edit these settings if you want the availability to be different than the default -- for example, any of the functionalities that by default display for system admins and group admins can be reconfigured to display only for system admins.

Note For instructions on changing these settings and applying your changes see the setting descriptions.

Function Area	Functionality	Default Availability	Controlling Setting
Manage Users	Whole user management interface	System admins and group admins	<code>"cmc.ui.admin.manageUsers.grantees"</code> (page 739)

Function Area	Functionality	Default Availability	Controlling Setting
	Create users	System admins and group admins	" cmc.ui.admin.manageUsers.createGrantees " (page 736)
	Edit users	System admins and group admins	" cmc.ui.admin.manageUsers.edit.grantees " (page 738)
	Delete users	System admins and group admins	" cmc.ui.admin.manageUsers.deleteGrantees " (page 737)
	View and manage users' security credentials	System admins and group admins	" cmc.ui.admin.manageUsers.edit.credentialsGrantees " (page 737)
	View and manage users' stored data	System admins and group admins	" cmc.ui.admin.manageUsers.view.objectDataGrantees " (page 739)
	Edit users' Quality of Service settings	System admins and group admins	" cmc.ui.admin.manageUsers.edit.qosGrantees " (page 738)
Manage Groups	Whole group management interface	System admins and group admins	" cmc.ui.admin.manageGroups.grantees " (page 736)
	Create groups	System admins	" cmc.ui.admin.manageGroups.createGrantees " (page 734)
	Edit groups	System admins and group admins	" cmc.ui.admin.manageGroups.edit.grantees " (page 735)
	Delete groups	System admins	" cmc.ui.admin.manageGroups.deleteGrantees " (page 734)
	Set default user QoS for a group	System admins and group admins	" cmc.ui.admin.manageGroups.edit.defaultQOSGrantees " (page 735)
Billing whitelist	Set source IP addresses allowed free traffic	Hidden from all user types	" admin.billing.allowlist.enabled " (page 702)
User Account Self-Management	Edit own profile	System admins, group admins, and regular users	" cmc.ui.account.profile.editGrantees " (page 733)
	Whole security credentials interface	System admins, group admins, and regular users	" cmc.ui.account.credentials.grantees " (page 732)

Function Area	Functionality	Default Availability	Controlling Setting
	Manage own S3 credentials	Group admins and regular users	" cmc.ui.account.credentials.accessKeyGrantees " (page 732)
	Change own CMC password	System admins, group admins, and regular users	" cmc.ui.account.credentials.signinGrantees " (page 732)
Usage Reporting	Whole usage reporting interface	System admins, group admins, and regular users	" cmc.ui.account.view.usageGrantees " (page 734)
	Reporting on HTTP request rates and byte transfer rates	Hidden from all user types	" Track/Report Usage for Request Rates and Data Transfer Rates " (page 502) in the CMC Configuration Settings page
Auto-Tiering	Allow buckets to be configured for auto-tiering	Hidden from all user types	" Enable Auto-Tiering " (page 504) in the CMC Configuration Settings page

5.13.2. Rebranding the CMC UI

If you wish you can customize various aspects of the CMC interface to reflect your organization's own branding. The interface elements that you can customize are:

- Logo (default is the Cloudian company logo)
- Browser tab title text (default is "Cloudian® Management Console")
- Color scheme (default is the Cloudian color scheme with black, white, gray, and green)
- Application name in URLs (default is "Cloudian")
- Login banner (by default the CMC login page has no banner)

To rebrand any or all of these interface elements you will use a HyperStore tool that simplifies the process of putting the relevant files into the proper location.

Before starting, decide whether you want a change of logos to be part of your rebranding of the CMC UI. If so, you will need three images of your organization's logo, using the following file names and pixel sizes:

Logo Image	Required Image File Name	Size in Pixels
Login screen logo	<i>logo_new.png</i>	225 X 225
Header logo	<i>logo_2.png</i>	144 X 28
Favicon for browser tab	<i>favicon.ico</i>	16 X 16

Before starting the rebranding procedure below, you should have the three image files on your local machine (for instance a laptop computer from which you will connect to the Config Controller node).

To rebrand the CMC UI, follow these steps:

1. Log into the Config Controller node as *root*.

*If you are using the **HyperStore Shell***

As an alternative to logging in as *root* you can log into the [HyperStore Shell](#) (HSH) on the Config Controller node to perform this procedure, so long as you are an HSH [Trusted user](#). In the steps below that call for using the *rebrand_cmc.sh* tool, run the tool simply as *rebrand_cmc.sh <command>* without specifying a path to the tool.

2. Use the *rebrand_cmc.sh* tool to copy the CMC's default branding assets into a pre-defined directory */etc/cloudian/cmc/custom/current* where you will subsequently customize the assets (as described in Step 4):

- Change into the */opt/tomcat/webapps/Cloudian/images* directory. Then run this command (and when prompted, type *y* to proceed with the asset copying):

```
/opt/cloudian/tools/rebrand_cmc.sh --images
```

- Change into the */opt/tomcat/webapps/Cloudian/WEB-INF* directory. Then run this command (and when prompted, type *y* to proceed with the asset copying):

```
/opt/cloudian/tools/rebrand_cmc.sh --resources
```

- Change into the */opt/tomcat/webapps/Cloudian/css* directory. Then run this command (and when prompted, type *y* to proceed with the asset copying):

```
/opt/cloudian/tools/rebrand_cmc.sh --css
```

- Change into the */opt/tomcat/webapps/Cloudian/jsp/includes* directory. Then run this command (and when prompted, type *y* to proceed with the asset copying):

```
/opt/cloudian/tools/rebrand_cmc.sh --custom_banner
```

3. Still on the Config Controller node, create a directory in which you will store backup copies of the existing, default CMC branding assets, before you implement your asset customizations (for example, create a directory under your home directory). Then change into that directory. Then run this command (and when prompted, type *y* to proceed with the asset backup):

```
/opt/cloudian/tools/rebrand_cmc.sh --backup
```

This creates under your working directory a sub-directory *web_backup_<timestamp>* containing the backup copies of the default CMC branding assets.

4. Change into the */etc/cloudian/cmc/custom/current* directory. Then make whichever CMC branding asset customizations you desire:

Replace logos

Copy your organization's logo image files -- as specified in the introduction to this procedure -- from your local machine to the */etc/cloudian/cmc/custom/current/images* directory on the Config Controller node (by using *scp*, for example). You will be overwriting the default image files with your custom files.

Change browser tab title text

By default the title text that displays at the top of a browser tab for the UI is "Cloudian® Management Console". To change this title, do the following:

- a. Change into the */etc/cloudian/cmc/custom/current/WEB-INF* directory.
- b. Use a text editor to edit the *resources.properties* file as follows:
 - i. In the file find this line:

```
header.title=Cloudian&reg; Management Console
```

- ii. Change it to the desired browser tab name:

```
header.title=<NEW TAB NAME>
```

- c. Make the same change in the *resources_<language-code>.properties* files in the directory.

Change UI color scheme

To change the UI's color scheme -- as defined in its CSS file -- follow the steps below. (This assumes some familiarity with CSS files, and knowledge of your organization's preferred color scheme.)

- a. Change into the */etc/cloudian/cmc/custom/current/WEB-INF* directory.
- b. Use a text editor to edit the copy of *master.css* file, to replace all occurrences of the existing color codes with the desired new values.

Create a login page banner

By default the CMC does not have a login page banner (text that displays at the top of the login page every time any user accesses the CMC). To create a login page banner, follow the steps below.

- a. Change into the */etc/cloudian/cmc/custom/current/jsp/includes* directory.
- b. Use a text editor to make the following edits to the *custom_banner.jsp* file:

- Uncomment the starting *style* tag.

Default in *custom_banner.jsp* file (with starting *style* tag commented out):

```
<!-- style>
```

After uncommenting:

```
<style>
```

- Replace *Title* with your desired banner title text. Do not alter the *h2* tags.

```
<h2>Title</h2>
```

- Replace *Message* with your desired banner body text. Do not alter the *p* tags.

```
<p>Message</p>
```

Change the application name in the CMC's URLs

By default all CMC URLs include the application name "Cloudian" (for example *https://<host>:8443/Cloudian/dashboard.htm*). To change the application name, run the following *hscctl* command:

```
# hscctl config set cmc.applicationName=<string>
```

5. Apply your customizations to the CMC:

```
# hscctl config apply cmc
# hscctl service restart cmc --nodes=ALL
```

Note If you customize the branding of the CMC, and then subsequently upgrade your HyperStore system to a newer version, only your customized logos and your customized application name will be retained after the upgrade. After the upgrade you will need to re-implement any changes that you had made to the browser tab title and/or the color scheme, by again following the instructions above.

5.13.2.1. Rebranding the CMC Help

The CMC has three Help systems attached to it: one for system administrators, one for group administrators, and one for end users. The Help that displays for a logged-in CMC user depends on which of those three types of user he or she is.

Each of the three Help systems has a Clodian logo. For each Help system the logo image file is named *CloudianLogoFull_2.png* and its size is 235 X 44 pixels.

To replace the Clodian logo in the CMC Help with your organization's logo, replace these three instances of the logo file on each of your HyperStore nodes:

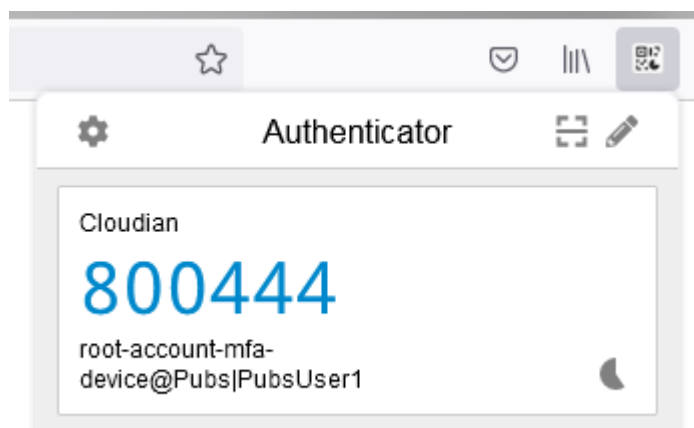
- `/opt/cloudian-packages/cloudian-cmc/apache-tomcat-8.5.93/webapps/Cloudian/help/HyperStoreHelp/Skins/Default/Stylesheets/Images/CloudianLogoFull_2.png`
- `/opt/cloudian-packages/cloudian-cmc/apache-tomcat-8.5.93/webapps/Cloudian/help/HyperStoreHelpGroupAdmin/Skins/Default/Stylesheets/Images/CloudianLogoFull_2.png`
- `/opt/cloudian-packages/cloudian-cmc/apache-tomcat-8.5.93/webapps/Cloudian/help/HyperStoreHelpEndUser/Skins/Default/Stylesheets/Images/CloudianLogoFull_2.png`

Note that:

- Changing this image file is not supported by the *rebrand_cmc.sh* script, and this image file is not under Salt control. To replace this image file you must do it manually on each of your HyperStore nodes.
- Changing this image file requires *root* access and cannot be performed through the HyperStore Shell.
- In the Help for group administrators and the Help for regular users, the Help text makes no reference to "Clodian" or "HyperStore" or "the CMC". Instead the text uses generic, non-branded terminology such as "the object storage system" and "the console".

5.13.2.2. Rebranding the MFA "Issuer"

Users can enable multi-factor authentication (MFA) on their CMC login accounts (see the description of the CMC **"Security Credentials"** (page 572) interface). The **"iam.mfa.totp.issuer"** (page 792) setting specifies the MFA TOTP (time-based one-time password) Issuer name that will display in a user's virtual MFA device if the user enables MFA on their CMC account. The example below is from the virtual MFA device Google Authenticator, with the Issuer at its default value "Clodian".



If you wish you can change this setting's value (so that your own organization's name is used rather than "Clodian").

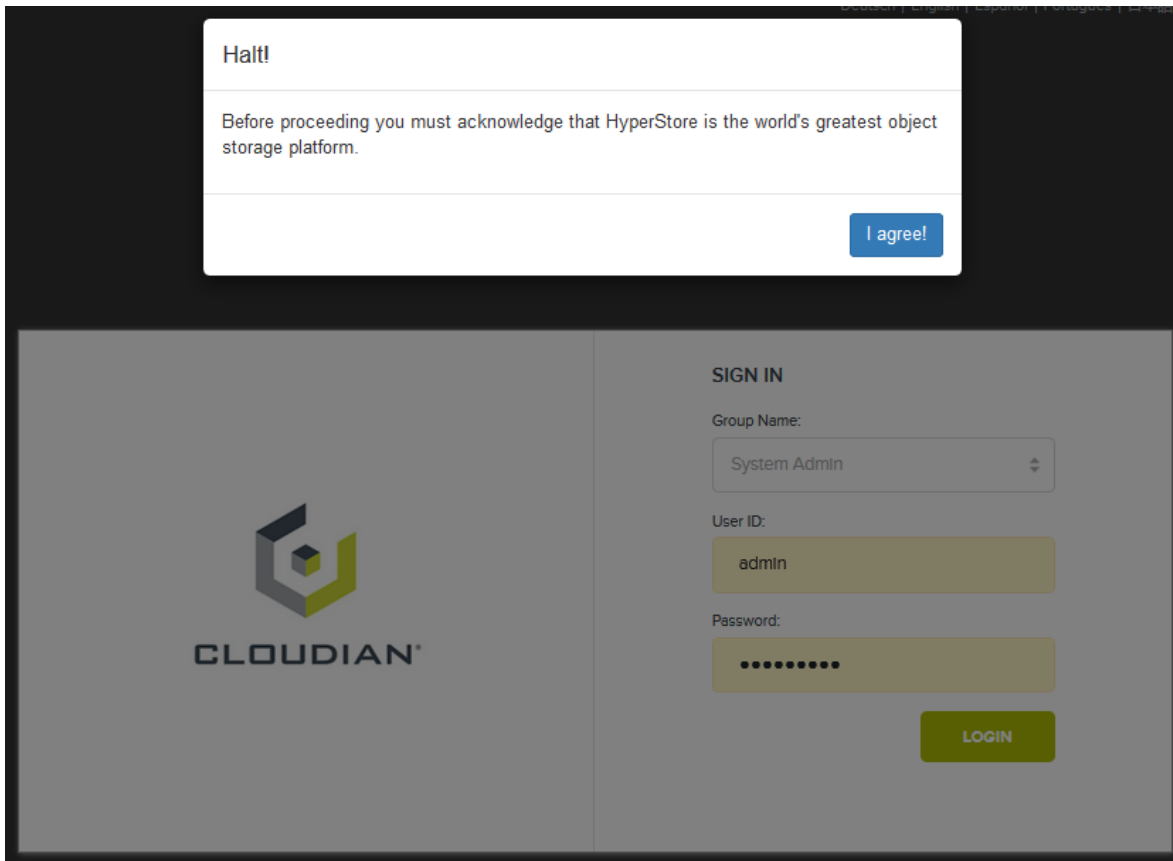
5.13.3. Configuring a Login Page Acknowledgment Gate

The CMC supports two types of customizations specifically for the login page:

- You can configure a **text banner** that displays at the top of the login page every time any user accesses the CMC.
- You can configure an **acknowledgment gate** that displays as an overlay in front of the login page every time any user accesses the CMC, and requires that the user acknowledge having read the gate text before they can log into the CMC.

By default the CMC login page has no banner and no acknowledgment gate. You can enable either one of these customizations, or enable both of them. This section describes how to configure an acknowledgment gate; for instructions for configuring a plain login page banner that does not require acknowledgment see ["Rebranding the CMC UI" \(page 266\)](#).

Here is an example in which an acknowledgment gate has been added to the login page. The CMC implements the acknowledgment gate as a modal dialog.



To configure a CMC login page acknowledgment gate, on the Config Controller node run these *hsctl* commands (see further below for setting descriptions):

```
# hsctl config set cmc.login.gate.size=<value>
# hsctl config set cmc.login.gate.title=<value>
# hsctl config set cmc.login.gate.message=<value>
# hsctl config set cmc.login.gate.buttonConfirm=<value>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

- `cmc.login.gate.size` -- This setting controls the width of the acknowledgment dialog. The valid values are 0, 1, 2, or 3, with 0 being the narrowest and 3 the widest. The default is 1. In the screen shot above, the width is set to 1.
- `cmc.login.gate.title` -- Title text of the acknowledgment dialog.
- `cmc.login.gate.message` -- Body text of the acknowledgment dialog. If you want to apply any formatting to this text, use HTML format tags within the text. For example, `<br><br>` to start a second paragraph with a line break between paragraphs; or `<b>text</b>` to create bold text.
- `cmc.login.gate.buttonConfirm` -- Text of the "confirm" button in the acknowledgment dialog. For example, you could use *Confirm* or *Acknowledge* or *Agree* or *OK* as the button text.

Note If any of the values that you assign to these settings has spaces or other non-alphanumeric characters, enclose the value in quotes.

After completing the configuration, go to the CMC in your browser to verify that the acknowledgment gate is working as desired.

5.13.4. Implementing Single Sign-On for the CMC

To enable integration between a portal and the Cloudian Management Console, the Cloudian HyperStore system employs a one-way hash based Single Sign-On (SSO) solution. It allows for cross-domain sign-ons from the portal to CMC.

User provisioning is beyond the scope of the provided SSO solution. The HyperStore provides an Admin API for user provisioning (see the "user" section of the *Cloudian HyperStore Admin API Reference*) but the implementation of user mapping is left to the portal application integrating with CMC.

5.13.4.1. How SSO for the CMC Works

Cloudian HyperStore SSO is ideal for sites that already have an authentication model in place using a browser/login session and that want to incorporate the Cloudian Management Console into their web portal application.

The idea is that a portal application calculates an one-way hash (also known as a signature) based on Cloudian HyperStore user identification, timestamp and the shared key. Then the user's browser accesses `sso-securelogin.htm` with the signature. The CMC checks for this signature to determine whether a user is authenticated or not. If the signature is found valid, access to the CMC from the client will skip the login page and take the user directly to a CMC interior page such as the **Buckets & Objects** page.

To use the Cloudian HyperStore SSO feature, the following system configuration settings must be set to *true*:

- `"cmc.tls.required"` (page 731) -- This is *true* by default. Leave it *true* if you want to use SSO.
- `"cmc.sso.enabled"` (page 727) -- This is *false* by default. Change it to *true* if you want to use SSO.

Also, for security reasons you should set this configuration setting to a custom value (**do not use the default key**):

- `"cmc.sso.sharedKey"` (page 727)

5.13.4.2. CMC SSO Secure Login Using One-Way Hash

The single sign-on with one-way hash method relies on a one-way hash of query string parameters (also known as a signature).

The following HTTP API, using a signature, prompts the CMC to create an authenticated session for the client that submitted the request:

Note Submit this as a GET, not a POST. POST is not supported for CMC SSO login.

```
https://<cmc_FQDN>:<cmc_port>/Cloudian/ssossecurelogin.htm?user=USERID&group=GROUPID
&timestamp=TIMESTAMP&signature=SIG&redirect=RELATIVE_OR_ABSOLUTE_URL
```

- *user*: Cloudian HyperStore userId of the user
- *group*: Cloudian HyperStore groupId of the group to which the user belongs
- *timestamp*: Current Epoch time in milliseconds (eg. "1346881953440"). The timestamp is used to implement the request expiration period which by default is one hour. This period is configurable by the "[cmc.sso.tolerance](#)" (page 727) setting.
- *signature*: This is the URI encoding of the base64 representation of the calculated signature. For further information see below.
- *redirect*: This optional parameter can be used to redirect the client to the given URL upon successful sign-in. It is typically set to a CMC interior page such as *bucket.htm*.

Each value must be URL-encoded by the client. Order of the parameters does not matter.

If the signature is found valid, the CMC creates an authenticated session for the HyperStore user, allowing the client to skip the login page and access to a CMC interior page.

How to Create the Signature

The portal server can create the signature by the following steps.

1. Assemble the query string.
 - querystring = "user=USERID&group=GROUPID&timestamp=TIMESTAMP"

Note When using the querystring to create the signature, do not URL-encode the querystring. Also do not reorder the items. (By contrast, when the client subsequently submits the SSO secure login request to the CMC, it's desirable to URL encode the request querystring.)

2. Calculate one-way hash for the querystring using the standard HmacSHA1 and the CMC SSO shared key. The shared key is configured by *common.csv*: *cmc_sso_shared_key*.
 - hashresult = HmacSHA1(querystring, sharedkey)
3. Base64 encode the resulting hash.
 - base64string = Base64Encode(hashresult)
4. URI encode the base64 encoded hash result.
 - signature = encodeURIComponent(base64string)

For a sample of a Python script that uses the one-way hash login API, see "[Cloudian HyperStore SSO Sample Script](#)" (page 274).

Access to a CMC's Interior Page

After creating the signature, the portal server can return an HTML page with a hyperlink to the CMC SSO secure login API. The following example will display CMC's **Buckets & Objects** page (*bucket.htm*) embedded in the inline frame on the portal's page.

```
<iframe src="https://<cmc_FQDN>:<cmc_port>/Cloudian/ssossecurelogin.htm
?user=USERID&group=GROUPID&timestamp=TIMESTAMP&signature=SIG
&redirect=bucket.htm"></iframe>
```

CMC SSO Secure Login HTTP Response

If *redirect=RELATIVE_OR_ABSOLUTE_URL* is given, the CMC's SSO secure login API returns an HTTP redirect response.

- If the request was successful, the redirect response will take the client to the URL specified by *redirect*.
- If the request failed, the redirect response will take the client to the CMC's Login panel.

If *redirect=RELATIVE_OR_ABSOLUTE_URL* is not given, the CMC's SSO secure login API returns an HTTP response with content-type "text/plain".

- If the request was successful, the HTTP response status is 200 OK.
- If the request failed, a 400 BAD REQUEST status is returned, along with a plain text status description. Possible reasons for failure include:

- Missing required parameters
- SSO token already exists (request is ignored)
- Timestamp in request is outside of configured tolerance range
- Invalid signature
- Invalid credentials (group ID and/or user ID is invalid)

CMC Logout

This API method allows for immediately invalidating the CMC session.

```
https://<cmc_FQDN>:<cmc_port>/Cloudian/logout.htm&redirect=RELATIVE_OR_ABSOLUTE_URL
```

- *redirect*: This optional parameter can be used to redirect the client to the URL after logging out from the CMC. It is typically set to a portal page. The URL must be URL-encoded by the client.

CMC Logout HTTP Response

If *redirect=RELATIVE_OR_ABSOLUTE_URL* is given, the CMC's logout API returns an HTTP redirect response to take the client to the given URL after logging out from the CMC.

If *redirect=RELATIVE_OR_ABSOLUTE_URL* is not given, the CMC's logout API returns an HTTP redirect response to take the client to the CMC's Login panel.

Logging Out from the CMC and Portal at Once

You may want the logout link on the portal page to also trigger logout from the CMC. You can achieve this by using the *redirect* parameter.

For example, if you have the portal's logout link like this:

```
<a href="/auth/logout">Logout</a>
```

You can change it to the following:

```
<a href="https://<cmc_FQDN>:<cmc_port>/Cloudian/logout.htm
?redirect=https:%2F%2F<portal_FQDN>:<portal_port>%2Fauth%2Flogout">Logout</a>
```

- The redirect URL must be an absolute URL including the protocol (e.g. https://) and portal's FQDN.
- The redirect URL must be URL-encoded.

5.13.4.3. Cloudian HyperStore SSO Sample Script

Below is a sample Python script that outputs a HyperStore SSO secure login URL for use with the one-way hash method of having the CMC create a cookie. The script also creates an SSO logout URL.

```
#!/usr/bin/python

import time
import hmac
import hashlib
import base64
import urllib

# TODO: Move these config options to configuration file
SSO_DOMAIN = 'cmc.cloudian.com'
SSO_PORT = 8443
SSO_KEY = 'aa2gh3t7rx6d'

# TODO: Dynamically choose user/group based on the user
# and group you want to login using.
SSO_USER = 'sso@group'
SSO_GROUP = 'ssogroup'

# Do Not Change
SSO_PROTO = 'https://'
SSO_PATH = 'Cloudian/ssossecurelogin.htm'
SSO_LOGOUT_PATH = 'Cloudian/ssologout.htm'

def sso_sig(user, group, timestamp):
    # query string with no urlencoding for signature
    signme = 'user=%s&group=%s&timestamp=%s' % (user, group, timestamp)
    hmacsha1 = hmac.new(SSO_KEY, signme, hashlib.sha1).digest()
    return base64.b64encode(hmacsha1)

def sso_url(user, group):
    timestamp = int(time.time() * 1000)
    signature = sso_sig(user, group, timestamp)
    params = {'user': user,
              'group': group,
              'timestamp': timestamp,
              'signature': signature}
    query = urllib.urlencode(params)
    url = '%s%s:%d/%s?%s' % (SSO_PROTO, SSO_DOMAIN, SSO_PORT, SSO_PATH, query)
```

```

return url

def sso_logout_url():
    url = '%s%s:%d/%s' % (SSO_PROTO, SSO_DOMAIN, SSO_PORT, SSO_LOGOUT_PATH)
    return url

print 'login: ' + sso_url(SSO_USER, SSO_GROUP)
print '\nlogout: ' + sso_logout_url()

```

Note The sample script hard-codes the SSO secret key, which is not advisable for actual practice. In practice, you should keep the secret key safely on the server side.

5.14. Provisioning Groups and Users

5.14.1. Group and User Provisioning Feature Overview

Through the HyperStore Admin API or through the CMC, you can provision the user groups and individual users who you want to authorize to use the HyperStore S3 service. **You will provision groups first, and then you can add individual users to each group.** All users must belong to a group.

As a system administrator you can act on groups in a variety of ways:

- Each group can be configured for integration with an external LDAP system as a means of authenticating users, if applicable to your environment. For more information see ["LDAP Integration"](#) (page 278).
- Each group can be assigned [quality of service](#) (QoS) limits that will enforce upper bounds on the service usage levels of the group as a whole. Each group can also be assigned default user-level QoS controls that will limit the service usage of individual users within the group. (Optionally, you can also assign per-user QoS limits that will supersede this default.)

Note You can set system-wide defaults for group QoS limits and individual user QoS limits before provisioning any groups or users, if you wish. After provisioning groups and users you can set QoS limits specific to those groups and users, which would override the system-wide defaults. For more information see ["Quality of Service \(QoS\) Feature Overview"](#) (page 246).

- You can generate [service usage reports](#) for groups (and also for individual users).
- Each group can be assigned a default [rating plan](#) which will determine how users in that group will be charged for HyperStore service usage. (Optionally, you can also assign per-user rating plans that will supersede this default.)
- You can create one or more users who have group administrator privileges. By default system configuration, group administrators are able to perform the following operations through the CMC:
 - Create a user within the group
 - Edit a user's profile
 - Retrieve a list of users in the group
 - Assign user-specific QoS limits
 - Provide user support by accessing a user's data in the S3 object store

- Delete a user
- Generate a usage report for the group
- Generate a usage report for an individual user in the group

Note The set of privileges that you make available to group administrators is configurable in a granular way (see ["Showing/Hiding CMC UI Functions" \(page 264\)](#)). Individual CMC UI functions and sub-functions can be displayed to or hidden from group administrators depending on your configuration settings.

5.14.2. Provisioning Groups

You can provision user groups through either the CMC or the Admin API.

Note Optionally, when creating a group you can enable LDAP-based authentication of group members. For more information see ["LDAP Integration" \(page 278\)](#).

5.14.2.1. Provisioning Groups (CMC)

In the CMC's **Manage Groups** page you can perform group operations including:

- ["Add a Group" \(page 416\)](#)
- ["Set Quality of Service \(QoS\) Limits" \(page 430\)](#)
- ["Retrieve a Group or a List of Groups" \(page 420\)](#)
- ["Edit a Group" \(page 421\)](#) (including suspending a group)
- ["Delete a Group" \(page 422\)](#)

5.14.2.2. Provisioning Groups (Admin API)

Through the HyperStore Admin API you can perform group operations including:

- Create a new group: *PUT /group*
- Assign a rating plan to a group: *POST /group/ratingPlanId*
- Assign QoS limits to a group: *POST /qos/limits*
- Retrieve a list of groups: *GET /group/list*
- Edit a group (including suspending a group): *POST /group*
- Delete a group: *DELETE /group*

For details on these API calls see the "group" section of the [Cloudian HyperStore Admin API Reference](#).

5.14.3. Provisioning Users

You can provision individual users through the CMC, or the Admin API, or by enabling LDAP integration. This section covers the following topics:

- ["Provisioning Users \(CMC\)"](#) (page 277)
- ["Provisioning Users \(Admin API\)"](#) (page 277)
- ["Provisioning Users \(LDAP\)"](#) (page 277)
- ["Provisioning IAM Users"](#) (page 277)
- ["SAML-Based Assumption of IAM Roles"](#) (page 278)

Note The HyperStore system does not support bulk provisioning of users. Users must be added one at a time.

5.14.3.1. Provisioning Users (CMC)

In the CMC's **Manage Users** page you can perform user operations including:

- ["Add a User"](#) (page 401)
- ["Set Quality of Service \(QoS\) Limits"](#) (page 430)
- ["Retrieve a User or List of Users"](#) (page 404)
- ["Edit or Suspend a User"](#) (page 407) (including suspending a user)
- ["Delete a User"](#) (page 413)

5.14.3.2. Provisioning Users (Admin API)

Through the HyperStore Admin API you can perform user operations including:

- Create a new user: *PUT /user*
- Assign a rating plan to a user: *POST /user/ratingPlanId*
- Assign QoS limits to a user: *POST /qos/limits*
- Retrieve a list of users: *GET /user/list*
- Update a user's profile (including suspending a user): *POST /user*
- Delete a user: *DELETE /user*

For details on these API calls see the "user" section and the "qos" section in the ***Cloudian HyperStore Admin API Reference***.

5.14.3.3. Provisioning Users (LDAP)

As an alternative to provisioning users through the CMC or the Admin API, on a per-group basis you can enable integration between the CMC and your Active Directory or other LDAP system, such that users will be automatically provisioned within HyperStore when they log into the CMC with their LDAP credentials. For more information see ["LDAP Integration"](#) (page 278).

5.14.3.4. Provisioning IAM Users

Users provisioned by any of the methods noted above are **HyperStore account root users**. They have full permissions for performing S3 actions such as creating buckets and uploading and downloading objects, and they can perform S3 actions either through the CMC (which has a built-in S3 client) or through a third party S3 client application. Like Amazon Web Services, HyperStore allows account root users to create **IAM users** under their

root accounts. IAM (Identity and Access Management) users differ from HyperStore account root users in these important ways:

- IAM users have only the permissions granted to them by IAM policies created by the account root user.
- IAM users cannot log into the CMC. To perform S3 actions, IAM users must use a third party S3 client application to access the HyperStore S3 Service.
- HyperStore does not track service usage data specifically for IAM users. Instead, an IAM user's S3 storage activity counts toward the HyperStore usage data of the account root user who created the IAM user.

HyperStore account root users can create IAM groups, users, and policies in the CMC's **IAM** section. For more information see ["IAM" \(page 435\)](#).

HyperStore account root users can also create IAM groups, users, and policies by using a third party IAM client application that calls the HyperStore IAM Service. The HyperStore IAM Service supports all the IAM API calls associated with creating and managing IAM groups, users, and policies. For more information see the IAM section of the ***Cloudian HyperStore AWS APIs Support Reference***.

5.14.3.5. SAML-Based Assumption of IAM Roles

HyperStore account root users can create **IAM roles**, either through the CMC or by using a third party IAM client application that calls the HyperStore IAM Service. Those IAM roles, and IAM policies governing the roles, can be set up in a way such that the roles can be temporarily assumed by federated users who have been authenticated by Security Assertion Markup Language (SAML 2.0) identity provider systems external to HyperStore.

SAML-authenticated external users who temporarily assume an IAM role:

- Have only the permissions granted to them by the IAM policies associated with the role.
- Cannot log into the CMC. To perform S3 actions, SAML users temporarily assuming an IAM role must use a third party S3 client application to access the HyperStore S3 Service.
- Do not have their own service usage counts. Instead the S3 storage activity of such users counts toward the HyperStore usage data of the account root user who created the IAM role.

For more information see "SAML Support" in the IAM section of the ***Cloudian HyperStore AWS APIs Support Reference***.

5.14.4. LDAP Integration

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Enabling LDAP Authentication for a Group" \(page 279\)](#)
- ["Provisioning of Users within LDAP-Enabled Groups" \(page 279\)](#)
- ["Deleting Users from the CMC and/or LDAP" \(page 280\)](#)
- ["Disabling LDAP Authentication After Having Used It" \(page 280\)](#)
- ["LDAP Authentication of System Administrators and HSH Users" \(page 281\)](#)

HyperStore supports integrating with Active Directory or other types of LDAP systems so that users can log into the CMC with their LDAP-based login credentials. This feature is implemented on a per-group basis, so you have the option of creating some groups that are LDAP-enabled and others that are not. The system also sup-

ports having different groups use different Active Directory or LDAP servers for authentication, or having all LDAP-enabled groups use the same Active Directory or LDAP server.

Within an LDAP-enabled group, along with users who the CMC will authenticate against an Active Directory or other LDAP system you can optionally also have local users who the CMC will authenticate by use of a CMC-based password rather than LDAP.

Note

- * Under no circumstances does the CMC try to write to your Active Directory or LDAP server — it only reads from it, for the purpose of authenticating users.
- * Only system administrators can enable Active Directory or LDAP authentication for a group. Group administrators cannot enable Active Directory or LDAP authentication for their groups.
- * For LDAP authentication of HyperStore Shell users to work (see ["LDAP Authentication of System Administrators and HSH Users" \(page 281\)](#)), your LDAP server must use either TLS or START_TLS.

5.14.4.1. Enabling LDAP Authentication for a Group

To use the CMC to enable LDAP authentication for a group, use the [Add Group](#) interface (to create a new group with LDAP authentication enabled) or the [Edit Group](#) interface (to enable LDAP authentication for an existing group). When creating or editing the group select the "Enable LDAP Authentication" option and provide the required Active Directory or LDAP information.

To use the Admin API to enable LDAP authentication for a group, use the *PUT/group* method (to create a new group with LDAP authentication enabled) or the *POST/group* method (to enable LDAP authentication for an existing group). When creating or editing the group, in the *GroupInfo* object in the request body set the *ldapEnabled* attribute to true and also set the other LDAP-related attributes. For details on these API calls see the "group" section of the [Cloudian HyperStore Admin API Reference](#).

Note If you enable LDAP Authentication for an **existing group** to which you have already added users via the CMC's **Add User** function, those existing users will continue to be authenticated by reference to their CMC-based passwords -- not by reference to an LDAP server. LDAP Authentication will apply only for new users.

Note If you wish you can enable LDAP Authentication for the **System Admin group**, by editing the group either through the CMC or the Admin API. For further information specific to this group, see ["LDAP Authentication of System Administrators and HSH Users" \(page 281\)](#).

5.14.4.2. Provisioning of Users within LDAP-Enabled Groups

Within a HyperStore group that has LDAP authentication enabled you can have both LDAP-authenticated users and users who are authenticated by a CMC-based password rather than LDAP:

- **For users who you want to be authenticated by Active Directory or LDAP, do not manually create those users** through the CMC (or the Admin API). Instead, simply have those users log into the CMC using their LDAP credentials. If a user tries to log into the CMC as a member of an LDAP-authenticated group and the user is not already registered in HyperStore as a member of the group, the CMC will attempt to authenticate the user against the LDAP system. If the authentication succeeds, the CMC will

automatically provision the user into HyperStore. This includes automatic creation of security keys for accessing the HyperStore S3 data store. Going forward whenever the user logs in the CMC will recognize the user as a registered HyperStore user, but will continue to authenticate the user against the LDAP system each time rather than by reference to a CMC-based password.

Note that for such users to be successfully provisioned, the user names that they use when logging into the CMC must satisfy HyperStore restrictions for user names:

- Must be unique within the group.
- Only letters, numbers, dashes, and underscores are allowed. No spaces or special characters.
- Maximum allowed length is 64 characters.
- Must not be any of the following: "anonymous", "public", "null", "none", "admin", "0". These names are reserved for system use.

Note If you want the group administrator to be authenticated by LDAP, have the user log into the CMC using their LDAP credentials. Once this occurs and the CMC automatically provisions the user, you can subsequently edit the user's profile (using the CMC's **Edit User** function or the Admin API's *POST /user* method) to promote them to the group admin role.

- For users who you want to be authenticated by a CMC-based password rather than by the LDAP system, create those users through the CMC's **Add New User** interface (or the Admin API's *PUT /user* method). The CMC will not use LDAP-based authentication for users created through the **Add New User** interface or the *PUT /user* method.

5.14.4.3. Deleting Users from the CMC and/or LDAP

After you've enabled LDAP integration and have been using this feature, the HyperStore system behaves in the following way in respect to users being deleted from the CMC and/or LDAP:

- If you delete a user from the CMC but that user still exists in LDAP, the user will be able to log in to the CMC as if they were a first-time user and the CMC will auto-provision the user once again. If you want to prevent a user from accessing the CMC and HyperStore, but the user still exists in LDAP, the thing to do is to **deactivate** the user in the CMC (through the CMC's **Edit User** function), rather than deleting them. This will prevent the user from logging into the CMC or accessing HyperStore storage, even though they still exist in LDAP.
- If you delete a user from LDAP but do not delete them from the CMC, the user will not be able to log into the CMC. However, they still have valid S3 credentials and can access the HyperStore storage layer through a different S3 client. If you want a user who you've deleted from LDAP to not have access to the HyperStore S3 system, you should delete them from CMC also (which prevents access and also deletes the user's stored data) or else deactivate them in the CMC (which prevents access but leaves their stored data in place).

5.14.4.4. Disabling LDAP Authentication After Having Used It

If you have LDAP enabled for a particular group for some period of time, and during that time LDAP-based users from the group logged into the CMC with their LDAP credentials and were auto-provisioned into the HyperStore system, and then you subsequently disable LDAP for that group — those auto-provisioned users will no longer be able to log into the CMC.

5.14.4.5. LDAP Authentication of System Administrators and HSH Users

HyperStore supports enabling LDAP authentication for the System Admin group, which is a pre-existing group in any HyperStore system. You can do this through the CMC's **Edit Group** interface or the Admin API's *POST /group* method, just as you would for any group. Just as with any other group, any users who already exist in the System Admin group at the time that you enable LDAP authentication for the group will continue to be authenticated by reference to their CMC-based password. Just as with any other group, after enabling LDAP authentication for the System Admin group, if you want a new system admin user to be authenticated by LDAP **do not manually create that user** through the CMC or the Admin API -- instead, have that user log into the CMC with his or her LDAP credentials, and the user will then be automatically be provisioned into HyperStore (see ["Provisioning of Users within LDAP-Enabled Groups" \(page 279\)](#)).

When using LDAP authentication for the System Admin group:

- The default System Admin user -- with user ID *admin* -- is considered a pre-existing user and can only be authenticated by reference to the CMC-based password for that user. LDAP authentication is not supported for the *admin* user.
- To edit the System Admin group via the CMC or the Admin API you will need to know the group ID, which is "0" (the number zero).
- When a new system admin user is auto-provisioned into HyperStore as an LDAP-authenticated user (upon their first login to the CMC with their LDAP credentials), and the system automatically creates a corresponding HyperStore Shell user -- so that the new system admin user can log into and use the HyperStore Shell -- that user's **logins to the HyperStore Shell will also be LDAP-authenticated**. That is, when logging into the HyperStore Shell the user will supply their LDAP username and password, and the system will verify those credentials against your LDAP service.
 - For "local" system admin users -- who have been manually added through the CMC or the Admin API and who are not configured for LDAP authentication -- their HyperStore Shell user name is their CMC login user name prefixed by "sa_" (such as "sa_admin2"). By contrast, for LDAP-authenticated system admin users their HyperStore Shell user name is simply the same user name that they use to log into the CMC (without any prefix).
 - For LDAP authentication of HyperStore Shell users to work, along with enabling LDAP for the System Admin group in the CMC's **Edit Group** interface (or through the Admin API's *POST /group* method) you must perform this additional system configuration step:
 1. Log in to Config Controller node.
 2. Set the Distinguished Name for binding to your LDAP service, and the bind password:

```
hsctl config set hsh.ldap.bindDN=<bind Distinguished Name>
hsctl config set hsh.ldap.bindPassword=<bind password>
hsctl config apply hsh
```

For more information on the HyperStore Shell see:

- ["HyperStore Shell \(HSH\) Feature Overview" \(page 105\)](#)
- ["Enabling the HSH and Managing HSH Users" \(page 106\)](#)
- ["Using the HSH" \(page 111\)](#)

Note For LDAP authentication of HyperStore Shell users to work, your LDAP server must use either TLS or START_TLS.

Note LDAP authentication for HyperStore Shell users works only for system admin users created in HyperStore version **7.2.3 and later**.

Chapter 6. Cloudian Management Console (CMC)

6.1. Cloudian Management Console (CMC)

The Cloudian Management Console (CMC) is a web-based user interface for Cloudian HyperStore system administrators, group administrators, and end users. The functionality available through the CMC depends on the user type associated with a user's login ID (system admin, group admin, or regular user). System admins can perform a wide range of system maintenance and operational tasks as well as provisioning and managing user accounts. Group admins can perform a much more limited range of tasks, pertaining specifically to their group. Regular users can use the CMC to create and configure storage buckets and to upload or download data.

Just as the CMC's functionality is tailored to the logged-in user type, so too is the CMC's online help: users can only view the Help topics for the functionality that is available to them based on their user type.

Note All user logins to the CMC are recorded in the CMC application log [cloudian-ui.log](#).

Note The console supports Firefox and Chrome browsers. Internet Explorer is not supported — you may experience display problems with some console pages if you use IE.

To access the CMC for the first time, follow these steps:

1. Point a browser to `https://<CMC_host>:8443/Cloudian`

Since the CMC runs on all of your HyperStore nodes, for `<CMC_host>` you can use the fully qualified domain name (FQDN) or IP address of any node.

Note For load balancing CMC traffic (such as you would want to do if you expose the CMC to regular users so that they can use the CMC's S3 client interface), you can use the CMC's service endpoint rather than specifying a particular HyperStore node. By default this would be `https://cm-c.<your-domain>:8443/Cloudian`

2. You will get an SSL certificate warning. Follow the prompts to add an exception for the certificate. You should then see the CMC's login screen.



LOGIN

Group Name:

System Admin

User ID:

Username

Password:

Password

LOGIN

Note For information about managing SSL certificates in HyperStore -- including the option to replace the CMC's default self-signed certificate with a CA-signed certificate -- see ["Security and Privacy Features"](#) (page 177).

- Enter the system administrator user ID *admin* and the default password *public*. When you do so, the login screen will display additional fields in which you must create a new password for the *admin* user. After you create the new password and click **Save** you will be logged into the CMC.

Password update is required.



Group Name:

System Admin

User ID:

admin

Current Password:

.....

New Password:

.....

Confirm Password:

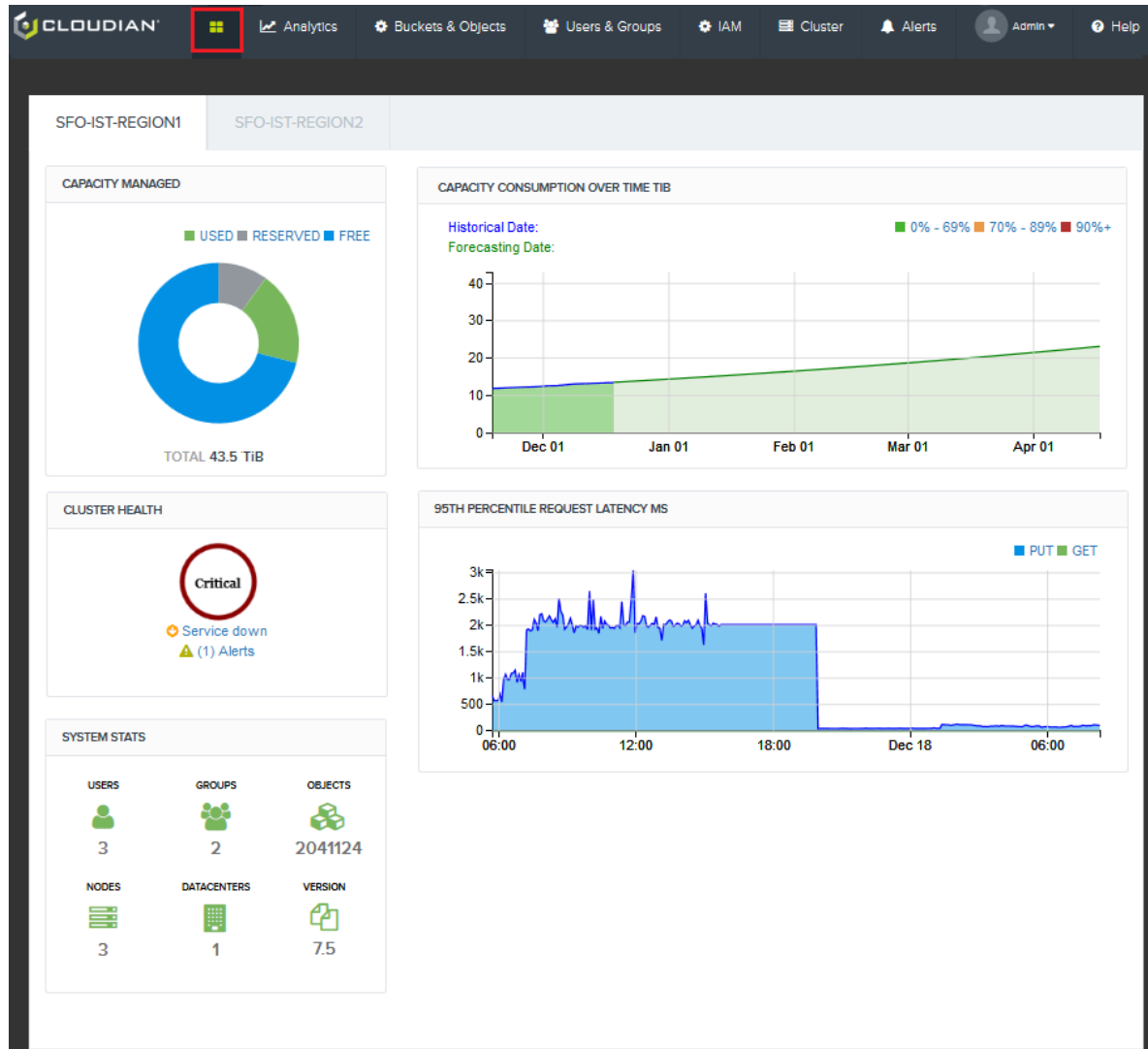
.....

SAVE

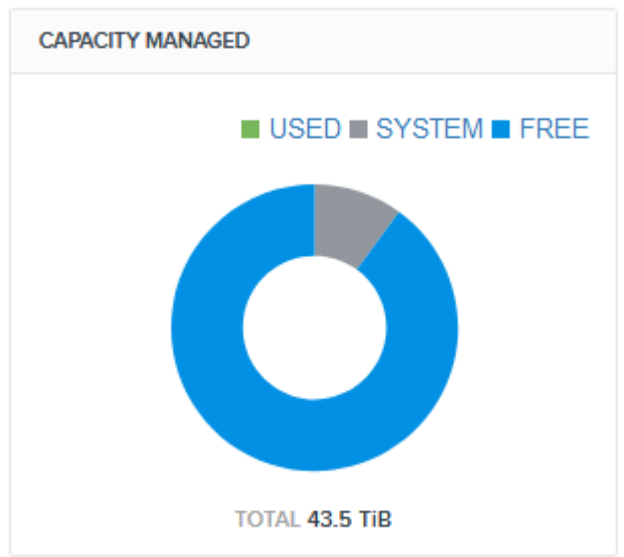
Note By default -- and as shown in the screen shot above -- the login page allows you to select your Group Name from a drop-down list. The system administrator group name in the drop-down list is System Admin. If the system has been configured with non-default settings for the CMC login page, it may be that the group name drop-down list does not appear. In this case you either need to check the "Admin Group" checkbox, or else enter the system admin **group ID** into the group name box if there is no "Admin Group" checkbox. The system admin group ID is **0**.

6.2. Dashboard

The CMC **Dashboard** provides a summary view of your HyperStore system status. In a multi-region system there is a separate dashboard tab for each region.



Capacity Managed



The **Capacity Managed** graphic shows the percentages of total disk space that are currently Used, System, or Free, across the whole service region in aggregate. To see the percentage numbers hold your cursor over each portion of the tri-colored circle.

- The **Used** segment of the circle indicates what portion of total disk space is consumed by stored object data. This segment displays in **green** if less than 70% of total capacity is used; or in **orange** if from 70% to 89% is used; or in **red** if 90% or more is used.

Note The Used capacity measurement here is **raw** usage and includes overhead from object replication or erasure coding. For example a 1MiB object that's replicated three times in the system counts as 3MiB toward the Used capacity measurement.

- The **System** segment indicates the portion of total disk space that is allocated to system needs and cannot be used for object data storage. The System portion consists of the Linux "reserved blocks percentage" plus the HyperStore stop-write buffer:
 - By default in Rocky Linux and RHEL the "reserved blocks percentage" for a file system (the portion of the disk space that's reserved for privileged processes) is 5% of disk capacity. In a HyperStore Appliance it's customized to 0%. See your OS documentation if you want to check or change the current reserved blocks percentage for your HyperStore host machines.
 - By default the HyperStore stop-write buffer is 10% of disk capacity. For information on this feature see ["Automatic Stop of Writes to a Disk at 90% Usage"](#) (page 125).

The System segment always displays in **gray**.

- The **Free** segment indicates the portion of total disk space that is neither used nor allocated to system needs, and is therefore available for storing new object data. The Free segment always displays in **blue**.

This graphic links to the **Capacity Explorer** page, where you can see a breakdown of your available capacity by region, data center, and node.

Note If there are any nodes down in the system, the used, free, and total capacity associated with the down node(s) will not be included in the totals shown in the Capacity Managed graphic.

Operation Status

OPERATION STATUS				
TYPE	OPERATION NAME	STATUS	PROGRESS	LAST UPDATE
	addNode	 In progress		May-27-2018 10:30

The **Operation Status** section displays the status of any **in-progress** system operations that you have recently launched from the CMC. This section of the Dashboard will not appear if there are no in-progress operations.

Status reporting in this section is supported for these operation types:

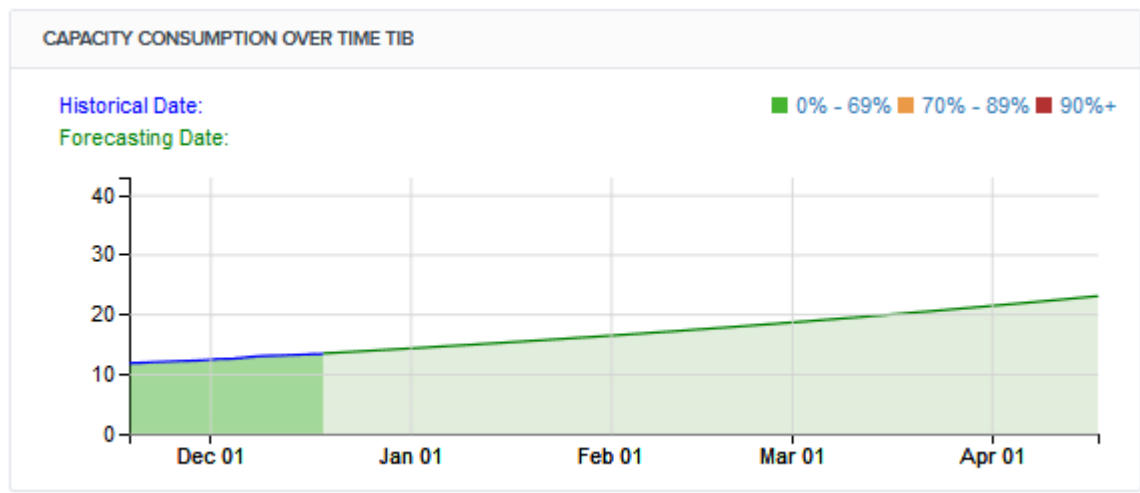
- Add Node
- Add Data Center
- Add Region
- Uninstall Node
- hssstool rebalance
- hssstool repair or repairec
- hssstool cleanup or cleanupec

For each operation the **Operation Status** section displays the operation name, the current status, the progress (as an approximate percentage of completion), and the last update time (the last time that the CMC obtained status information for the operation).

Clicking in this section takes you to the **Operation Status** page where you can view additional status information for these in-progress operations as well as status information for operations that have recently completed.

Note For *hssstool* operations, the Dashboard's **Operation Status** section and the **Operation Status** page show status only for *hssstool* operations that you initiate through the CMC's [Node Advanced](#) page -- not for *hssstool* operations that you initiate on the command line. For commands that you launch on the command line you can use [hssstool opstatus](#) to track the operation status.

Capacity Consumption Over Time



The **Capacity Consumption Over Time** graphic shows the recent and forecasted raw storage consumption for the region, in GiBs, TiBs, PiBs, or EiBs (depending on the size of your system). For this graphic, storage capacity is considered to be "consumed" if it is either used or allocated to system needs. For more information on these categories of capacity consumption see ["Capacity Managed"](#) (page 285).

The left side of the graph (in darker shades of color) shows the storage consumption level over the past 30 days. Consumption up to 69% of total disk space capacity is shown in **green**; consumption from 70% to 89% of capacity (if applicable) is shown in **orange**, layered on top of the green portion; and consumption from 90% or above of capacity (if applicable) is shown in **red**, layered on top of the green and orange portions.

The center and right of the graph show (in lighter shades of the same colors) the 120 day forecast of raw storage consumption level for the region, based on trend analysis from the past 30 days.

For a numerical display of the storage consumption level at a particular point in time, hold your cursor over that part of the timeline.

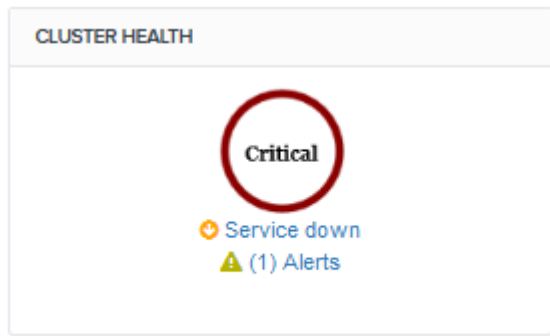
Links to the **Cluster Usage** page, where you can break this capacity consumption data down by data center and node.

IMPORTANT ! If your cluster is projected to reach 90% usage in the next 120 days, it's time to plan for a cluster expansion. See ["Capacity Monitoring and Expansion"](#) (page 656).

Note A "Critical" message will display at the top of the screen if your total disk space consumption is forecasted to reach 95% of capacity in the next 90 days. The message indicates when this 95% threshold is forecasted to be reached. If an 80% threshold is forecasted to be reached, a "Warning" message displays.

Note If there have been any nodes down in the system at the times when the system collected capacity statistics for the Capacity Consumption Over Time graph, the used and total capacity associated with the down node(s) will not be included in the capacity consumption percentages calculated from those times. This may give the appearance of temporary "dips" in the capacity consumption graph.

Cluster Health



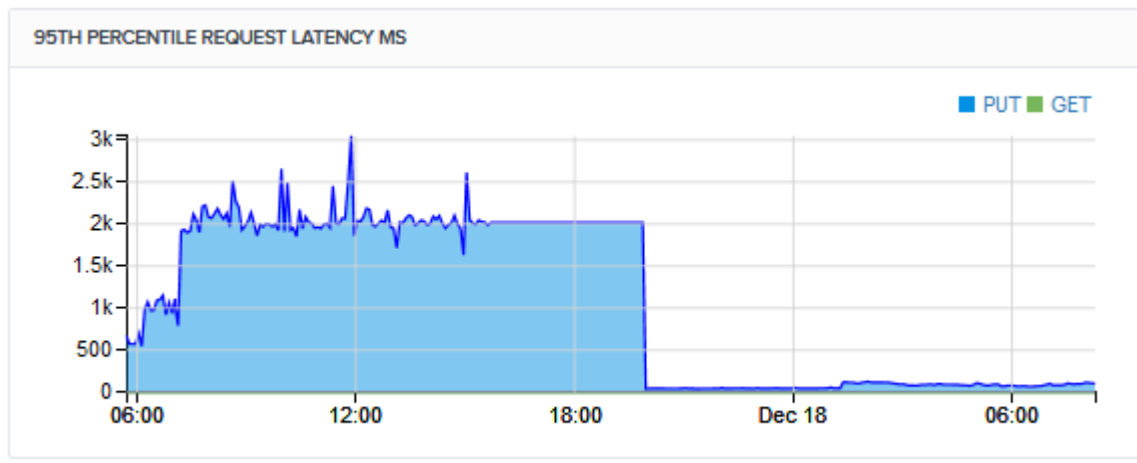
The **Cluster Health** section displays high level status information for the cluster. This section displays one or more condition messages, and an overall cluster status icon. The table below shows the possible condition messages along with the cluster status icon that displays if that condition is the most severe condition currently present in the cluster.

Cluster Status Icon	Condition Message(s)	Action to Take
	"System is X% full" (where X is 90 or more)	Click the condition message to go to the Capacity Explorer page for more detail regarding system capacity usage. At this level of storage capacity utilization the system may soon stop supporting new writes if it has not done so already. It's urgent to add more nodes to your system as soon as possible. For more information see "Capacity Monitoring and Expansion" (page 656). Note The percentage shown in this message is the used percentage plus the percentage allocated to system needs. For more information on these categories of capacity consumption see "Capacity Managed" (page 285).
	"Hardware issues"	This message indicates that a data disk has errors or has been disabled. Click the condition message to go to the Data Centers page where you can determine the node on which the bad disk is located. Then from there click the node icon to go to the Node Status page to determine which disk is bad. You may need to replace the disk. For more information see "Replacing a HyperStore Data Disk" (page 642).
	"Service down"	A HyperStore service is down on a node. Click the condition message to go to the Data Centers page. By reviewing the "Service Status" section of that page you can determine which node has a service down, and which service it is. Then in the "Service Status" section click the node's hostname to go to the Node Status page where you can start the service that's down (alternatively you can use the service initialization script to start the down service as described in "Starting, Stopping, and Checking Status of Services" (page 91)).

Cluster Status Icon	Condition Message(s)	Action to Take
	"Node stopped write"	A node has stopped accepting writes -- and the system has stopped directing S3 write requests to that node -- because all of the node's disks are nearly full. For more information on this condition and how to remedy it see "Automatic Stop of Writes to a Node at 90% Usage" (page 125)
	"License usage X%" (where X is 90 or more)	Contact Cloudian to request a new license. For more information see "Licensing and Auditing" (page 36). Note Only your capacity usage for object storage counts toward this percentage -- capacity allocated to system needs does not count toward this.
	None	This status icon displays if the system's S3 Service is in read-only mode because your HyperStore license has expired and your grace period (if any) has ended. Contact Cloudian to request a new license. For more information see "Licensing and Auditing" (page 36).
	"System forecast 90% full in X days" (where X is 90 or less)	Aggregate disk usage in the region is forecast to reach 90% of the region's total storage capacity in 90 days or less (based on usage trend analysis from the past 30 days). Click the condition message to go to the Cluster Usage page for more detail regarding current and forecasted system capacity usage. Add more nodes to your system as soon as possible. For more information see "Capacity Monitoring and Expansion" (page 656). Note The percentage shown in this message is the used percentage plus the percentage allocated to system needs. For more information on these categories of capacity consumption see "Capacity Managed" (page 285).
	"Node's Disks X% Full" (where X is 80 or more)	A node's data disks are nearing capacity. Click the condition message to go to the Capacity Explorer page for more detail regarding that node's capacity usage. Check also the capacity usage of other nodes as well, since if one node is nearing capacity other nodes may be approaching that level also. Add more nodes to your system as soon as possible. See "Capacity Monitoring and Expansion" (page 656). Note The percentage shown in this message is the used percentage plus the percentage allocated to system needs.

Cluster Status Icon	Condition Message(s)	Action to Take
		ated to system needs. For more information on these categories of capacity consumption see "Capacity Managed" (page 285).
	"Long proactive repair queue"	A node has a long " Proactive Repair" (page 131) queue. This warning is triggered if a node's proactive repair queue has built up to the point that proactive repair needs to write data for 10,000 or more objects to the node. Click the condition message to go to the Repair Status page to see which node has status "Proactive Repair Pending". Click that node's icon to see detail about the proactive repair queue length. Then click the node's hostname to go to the Node Status page for that node to check whether any services are down on the node. From that page you can start any down services. Other possible explanations for a long proactive repair queue are that the node is in main-tenance mode or in a stop-write condition; if so then in the Data Centers page the node's icon will be blue.
	"License usage X%" (where X is from 70 to 89)	Storage usage in the system as a whole is at 70% of licensed usage or higher, but still below 90%. Contact Cloudian to request a new license soon. For more information see "Licensing and Auditing" (page 36). Note Only your capacity usage for object storage counts toward this percentage -- capacity allocated to system needs does not count toward this.
	None or "(X) Alerts" (where X is the number of alerts)	This status icon displays if none of the conditions noted above apply. Note that even if the system is considered to be Healthy, there may be unacknowledged alerts in the system. Click the condition message to go to the Alerts page where you can review the alerts and acknowledge them.

95th Percentile Request Latency



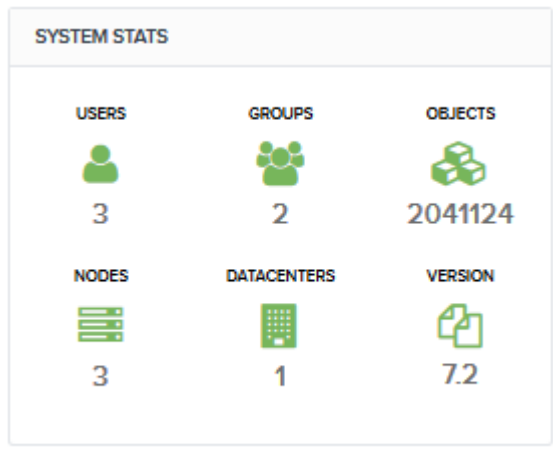
The **95th Percentile Request Latency MS** graphic shows the region-wide 95th percentile latencies for S3 GET and PUT transactions in milliseconds. New statistic values are calculated and plotted each five minutes, based on the most recent approximately 1000 GET transactions and 1000 PUT transactions. Each plotted 95th percentile latency value indicates that of the last 1000 transactions of that type, 95% completed in that many milliseconds or less.

PUT latency is shown in blue and GET latency is shown in green. Hold your cursor over any point in time to view the specific PUT and GET latency 95th percentile figure for that point in time.

Note If request activity completely stops for some period of time -- such that there is no new raw data going into the statistical calculation -- the graph will for a while continue to plot the statistic value from the last-completed 1000 transactions (resulting in a perfectly horizontal line across part or all of the inactive period) rather than the graph line immediately dropping down to zero.

Note In regard to the per-transaction latency measurements that go into calculating a 95th percentile figure over a given time period, note that S3 uploads of very large objects are typically implemented as multipart upload (MPU) operations, as S3 client applications use the S3 API calls for MPU. In terms of PUT latency metrics, each part upload of an MPU is treated as a separate transaction -- that is, a latency is recorded for each part upload rather than for the aggregate multipart upload operation. By contrast a GET of a very large object is a single transaction with a single latency measurement recorded.

System Stats



The **System Stats** graphic displays basic information about your HyperStore system:

Users

Number of users in your whole HyperStore system. If your system has multiple service regions, users are registered to the system as a whole (they are not tied to a particular region). So if you toggle through the service region tabs, this number will remain the same -- showing the number of users in the system as a whole.

Along with regular S3 service users, the total number includes system admin users and group admin users.

Links to the **Manage Users** page, where you can create new users or retrieve and edit existing users.

Groups

Number of user groups in your whole HyperStore system. If your system has multiple service regions, user groups are registered to the system as a whole (they are not tied to a particular region). So if you toggle through the service region tabs, this number will remain the same -- showing the number of user groups in the system as a whole.

Links to the **Manage Groups** page, where you can create new groups or retrieve and edit existing groups.

Objects

Number of S3 objects stored in the service region. In a multi-region system, this count is for the region for which the tab is selected at the top of the Dashboard.

This statistic counts each object, not each replica — for example, if an object is replicated three times across your cluster (to protect data durability and availability), this counts as one object, not three.

In the case of versioned objects, each version of the object counts towards this stat.

S3 objects that have been auto-tiered to external systems (such as Amazon or a different HyperStore system) count toward this stat.

Links to the **Object Locator** page, where you can retrieve information about where a particular object is stored within your cluster.

Note The Objects count display uses the International System of Units, and so "M" stands for million, "G" stands for billion, and "T" stands for trillion. For example a count of "2.5 G" means 2.5 billion objects.

Nodes

Number of nodes in the service region. In a multi-region system, this count is for the region for which the tab is selected at the top of the Dashboard.

Links to the **Node Status** page, where you can see status details for individual nodes, start and stop HyperStore services on nodes, and review and acknowledge node alerts.

Data Centers

Number of data centers in the service region. In a multi-region system, this count is for the region for which the tab is selected at the top of the Dashboard.

Links to the **Data Centers** page, where you can see your HyperStore node inventory in each of your data centers.

Version

HyperStore software version.

Links to the **Cluster Information** page, where you can view static information about your cluster such as license information, the S3 service endpoint for your system, and the mapping of specialized service roles (such as the Redis QoS database master role) to individual nodes.

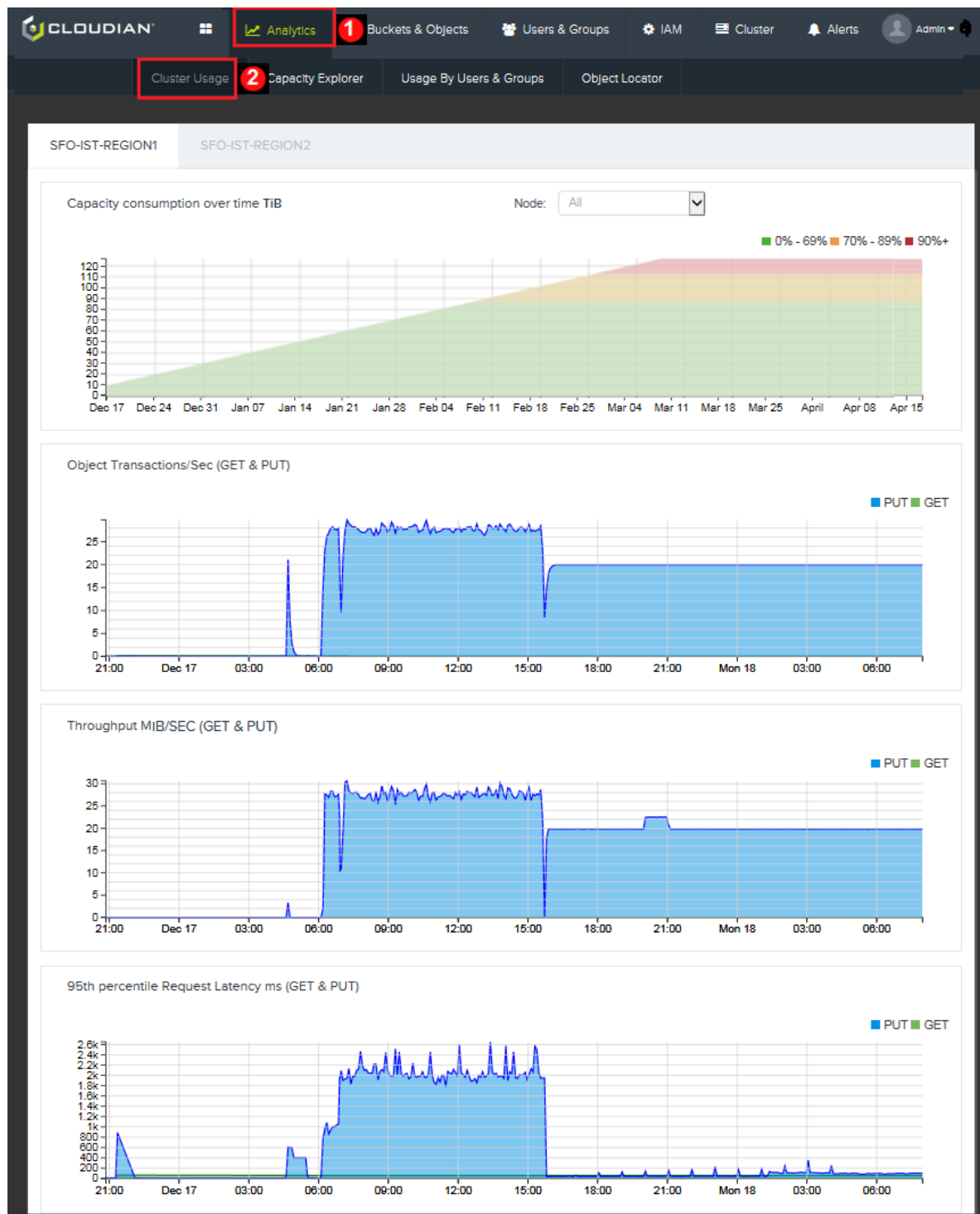
6.3. Analytics

The **Analytics** tab contains the following functions:

- [Cluster Usage](#)
- [Capacity Explorer](#)
- [Usage By Users & Groups](#)
- [Object Locator](#)

6.3.1. Cluster Usage

Path: **Analytics** → **Cluster Usage**



Supported tasks:

- View recent and forecasted cluster usage

In the CMC's **Cluster Usage** page you can view cluster usage graphs that cover the past 30 days of activity (or less if the cluster has been in operation for less than 30 days). The time period is not editable. For storage capacity consumption, along with recent capacity consumption you can also view a forecast of future consumption.

IMPORTANT ! See "[Capacity Monitoring and Expansion](#)" (page 656) for guidance about capacity management and the importance of early planning for cluster expansions.

The following graphs are displayed:

Capacity consumption over time

This graphic shows the recent and forecasted raw storage consumption for the region, in GiBs, TiBs, PiBs, or EiBs (depending on the size of your system). For this graphic, storage capacity is considered to be "consumed" if it is either used or reserved. That is, **capacity "consumption" at any given time is equal to "used" capacity plus "reserved" capacity**. For more information on "reserved" capacity see "[Capacity Managed](#)" (page 285).

The left side of the graph (in darker shades of color) shows the storage consumption level over the past 30 days. Consumption up to 69% of total disk space capacity is shown in **green**; consumption from 70% to 89% of capacity (if applicable) is shown in **orange**, layered on top of the green portion; and consumption from 90% or above of capacity (if applicable) is shown in **red**, layered on top of the green and orange portions.

The center and right of the graph show (in lighter shades of the same colors) the 120 day forecast of raw storage consumption level for the region, based on trend analysis from the past 30 days.

For a numerical display of the storage consumption level at a particular point in time, hold your cursor over that part of the timeline.

The drop-down list at the top of the graphic lists all the nodes in the service region. You can select from the list to view recent and forecasted usage for a specific node.

IMPORTANT ! If your cluster is projected to reach 90% consumption in the next 120 days, it's time to plan for a cluster expansion. See "[Capacity Monitoring and Expansion](#)" (page 656).

Object transactions/sec (GET & PUT)

This graph shows the number of S3 transactions processed per second. The graph shows the PUT transaction activity in blue and the GET transaction activity in green. To highlight one or the other trend line, hold your cursor over the "PUT" or "GET" label off to the right of the graph.

The transactions per second values are calculated and plotted each five minutes, based on the past five minutes of activity.

Note HEAD transactions are counted toward the GET stat, and POST transactions are counted toward the PUT stat.

Throughput MiB/sec (GET & PUT)

This graph shows the S3 data throughput as KiB or MiB or GiB per second. The graph shows the PUT transaction activity in blue and the GET transaction activity in green. To highlight one or the other trend line, hold your cursor over the "PUT" or "GET" label off to the right of the graph.

The throughput per second values are calculated and plotted each five minutes, based on the past five minutes of activity.

Note HEAD transactions are counted toward the GET stat, and POST transactions are counted toward the PUT stat.

95th percentile Request Latency ms (GET & PUT)

This graph shows the 95th percentile latencies for S3 GET and PUT transactions in milliseconds. The graph shows the PUT transaction activity in blue and the GET transaction activity in green. To highlight one or the other trend line, hold your cursor over the "PUT" or "GET" label off to the right of the graph.

New statistic values are calculated and plotted each five minutes, based on the most recent approximately 1000 GET transactions and 1000 PUT transactions. Each plotted 95th percentile latency value indicates that of the last 1000 transactions of that type, 95% completed in that many milliseconds or less.

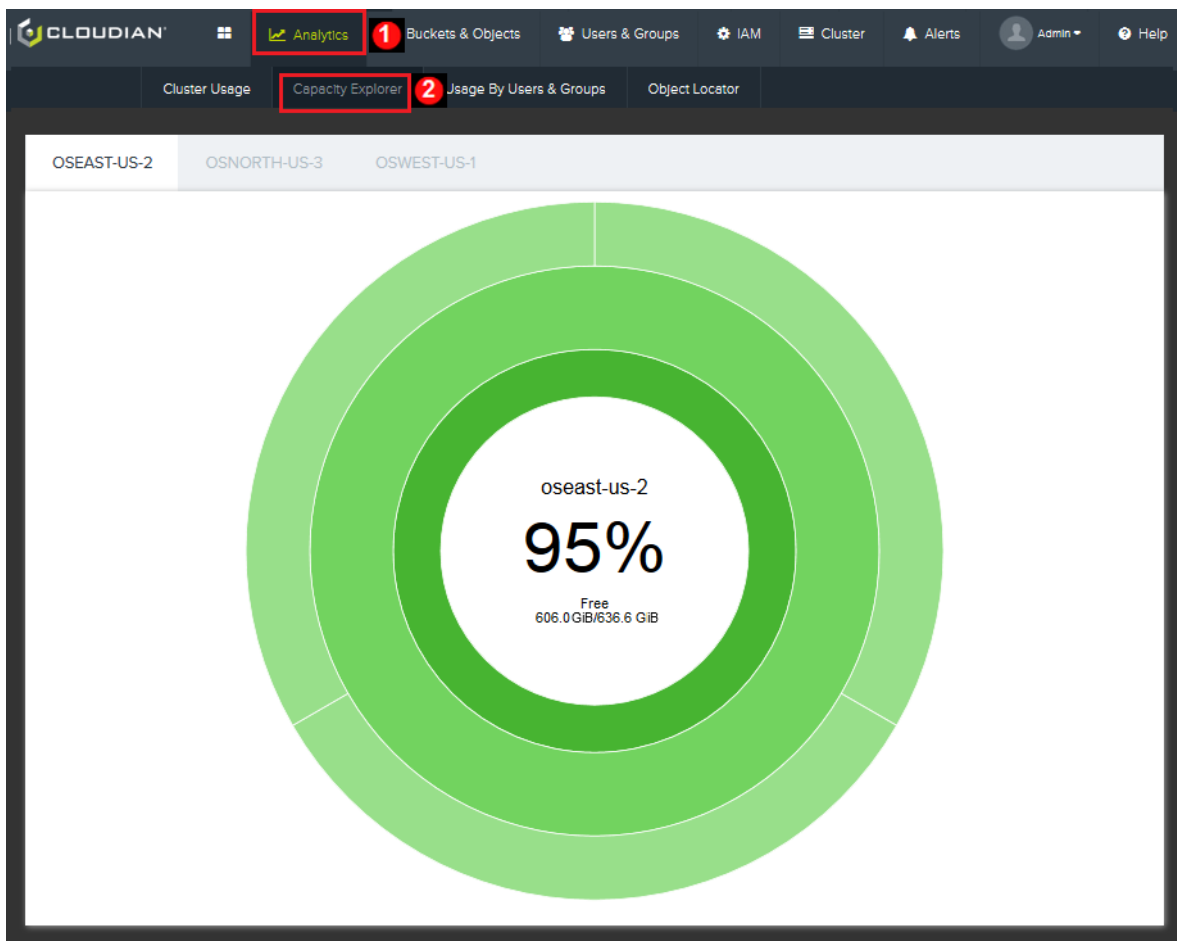
Note HEAD transactions are counted toward the GET stat, and POST transactions are counted toward the PUT stat.

Note If request activity completely stops for some period of time -- such that there is no new raw data going into the statistical calculation -- the graph will for a while continue to plot the statistic value from the last-completed 1000 transactions (resulting in a perfectly horizontal line across part or all of the inactive period) rather than the graph line immediately dropping down to zero.

Note S3 uploads of very large objects are typically implemented as multipart upload (MPU) operations, as S3 client applications use the S3 API calls for MPU. In terms of PUT latency metrics, each part upload of an MPU is treated as a separate transaction -- that is, a latency is recorded for each part upload rather than for the aggregate multipart upload operation. By contrast a GET of a very large object is a single transaction with a single latency measurement recorded.

6.3.2. Capacity Explorer

Path: **Analytics** → **Capacity Explorer**



Supported task:

- View remaining free space for S3 object data storage

With the CMC's **Capacity Explorer** page you can view your **remaining free storage capacity** by region, by data center, and by node. **"Free" capacity here is capacity that is neither used nor reserved** (that is, free capacity is what remains after deducting both used capacity and reserved capacity from total capacity). For more information on "reserved" capacity see ["Capacity Managed"](#) (page 285).

First choose a region tab at the top of the page (if you have just region there will be just one tab). Then, in the graphical display:

- The **inner circle** represents the service region as a whole
- The **middle circle** has one segment for each data center in the region
- The **outer circle** has one segment for each node in each data center

The circle segments are color-coded as follows:

- **Green** indicates that free space is 30% or more of total space for that region, data center, or node. (Slightly different shades of green are used merely to differentiate the concentric circles from each other. Green has the same meaning regardless of the particular shade of green.)
- **Orange** indicates that free space is between 10% and 29% of total space for that region, data center, or node.
- **Red** indicates that free space is less than 10% of total space for that region, data center, or node.

Hold your cursor over a segment to see specific storage space availability for that region, data center, or node, expressed as "<Free space>/<Total capacity>". Click on a segment to have the space availability information for that region, data center, or node display in the center of the circle, where it will include the free space percentage as well as the absolute numbers for free and total space. By default the free space information for the region as a whole displays in the center of the circle.

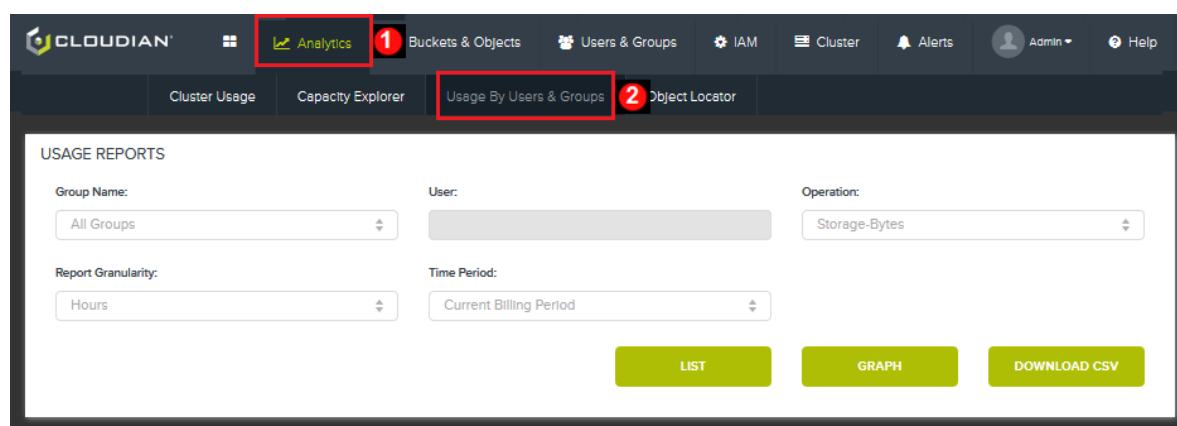
The measurements in the **Capacity Explorer** page are exclusively for HyperStore data disks -- on which S3 object data is being stored -- and do not address drives that are storing only the OS and the Cassandra and Redis databases (for system metadata). The HyperStore data disk utilization measured here is **raw** utilization and includes overhead for replication or erasure coding. For example a 1MiB object that's replicated three times in the system results in 3MiB of disk usage for the system.

IMPORTANT ! See ["Capacity Monitoring and Expansion"](#) (page 656) for guidance about capacity management and the importance of early planning for cluster expansions.

Note If there are any nodes down in the system, the used, free, and total capacity associated with the down node(s) will not be included in the totals shown in the Capacity Managed graphic.

6.3.3. Usage By Users & Groups

Path: **Analytics** → **Usage By Users & Group**



Supported tasks:

- ["Create a Usage Report"](#) (page 300)
- ["Review Usage Report Output"](#) (page 303)

Note For an overview of how the HyperStore system tracks service usage by groups and users, see ["Usage Reporting Feature Overview"](#) (page 252)

Note **Cloudbian HyperIQ** is a product that provides advanced analytics and visualization of HyperStore S3 Service usage by users and groups, as well as HyperStore system monitoring and alerting. For more information about HyperIQ contact your Cloudbian representative.

6.3.3.1. Create a Usage Report

In the CMC's **Usage By Users & Group** page you can generate service usage reports for individual users, for user groups, and for the system as a whole.

Note Cloudian HyperIQ is a product that provides advanced analytics and visualization of HyperStore S3 Service usage by users and groups, as well as HyperStore system monitoring and alerting. For more information about HyperIQ contact your Cloudian representative.

Usage reporting complies with Amazon S3 in that data storage and data transfer activity are always attributed to the bucket owner, regardless of who owns an individual object within the bucket, or who is submitting object-related requests. Also, if the account root user creates IAM users under their account, any activity associated with the IAM users' buckets counts toward the account root user's usage.

To create a usage report, choose your report filtering criteria:

Group Name

- ID of the Group to report on. For a system-wide report choose "All Groups".

User

- User ID, to optionally limit the report to a specific user's activity.

Operation

- *Storage-Bytes* — This report shows number of stored bytes. This is **net** bytes and excludes any storage policy overhead. For example, in the case of a 1MiB object that's protected by 3X replication, this counts as 1MiB toward Storage-Bytes -- not 3MiB.
- *Storage-Objects* — This report shows number of stored objects.
- *Data Transfer-In-Bytes* — This report shows data upload activity.
- *Data Transfer-Out-Bytes* — This report shows data download activity.
- *All Requests* — This report shows HTTP PUT, GET, and DELETE activity.
- *Get/Head Requests* — This report shows HTTP GET and HEAD activity. In List form this report is identical to the Data Transfer-Out-Bytes. In Graph form, Get/Head Requests graphs a request count while Data Transfer-Out-Bytes graphs number of bytes.
- *Put/Post Requests* — This report shows HTTP PUT and POST activity. In List form this report is identical to the Data Transfer-In-Bytes. In Graph form, Put/Post Requests graphs a request count while Data Transfer-In-Bytes graphs number of bytes.
- *Delete Requests* — This report shows HTTP DELETE activity.

Note By default only the Storage-Bytes and Storage-Objects reports are enabled. To enable the other types of reports, you must enable the [Track/Report Usage for Request Rates and Data Transfer Rates](#) setting on the CMC's **Configuration Settings** page.

Report Granularity

- *Hours* — Break the report period down into hourly intervals. The report will show only hours that have completed (to the top of the hour), not the currently in-progress hour.

- *Days* — Break the report period down into daily intervals. The report will show only days that have completed (midnight to midnight), not the currently in-progress day.
- *Months* — Break the report period down into monthly intervals. The report will show only months that have completed, not the currently in-progress month. For example, if today is June 25th and you set a Time Period from March 1st up through today, with Granularity of "Months", the report will show monthly usage data for the months of March, April, and May (and not June, since June hasn't completed yet).
- *Raw* — List every report-relevant transaction individually, with timestamp. For raw granularity, you must choose a custom Time Period that spans no more than 24 hours.

Time Period

- *Current Billing Period* — The current calendar month up to today's date.
- *Previous Billing Period* — The last completed calendar month.
- *Last Week* — The last completed calendar week (Monday to Sunday).
- *Last Month* — The last completed calendar month. This is equal to Previous Billing Period.
- *Custom Period* — This option opens a calendar tool in which you can specify a particular report begin date and report end date.

Region

- The service region for which to report usage activity. Choose "All" to show aggregate activity across all regions. This field displays only if your HyperStore system has multiple service regions.

Note "All" regions is a valid option only if you are generating a List report — not a Graph or CSV report. Multi-region Graph or CSV reports are not currently supported.

Traffic Type

- Choose "Normal" or "Whitelist". "Whitelist" refers to request traffic that originates from white-listed source IP addresses (traffic subject to special pricing), and is an option only if white-listing is enabled in the system. "Normal" refers to all other traffic.

Note The Traffic Type option does not apply to reports with Report Granularity "Raw", nor to reports with Storage-Bytes or Storage-Objects as the Operation. The Traffic Type field will not display if you choose those types of reports.

After specifying your usage report parameters, click:

- **List** to display a traditional tabular report.
- **Graph** to display a graphical report.

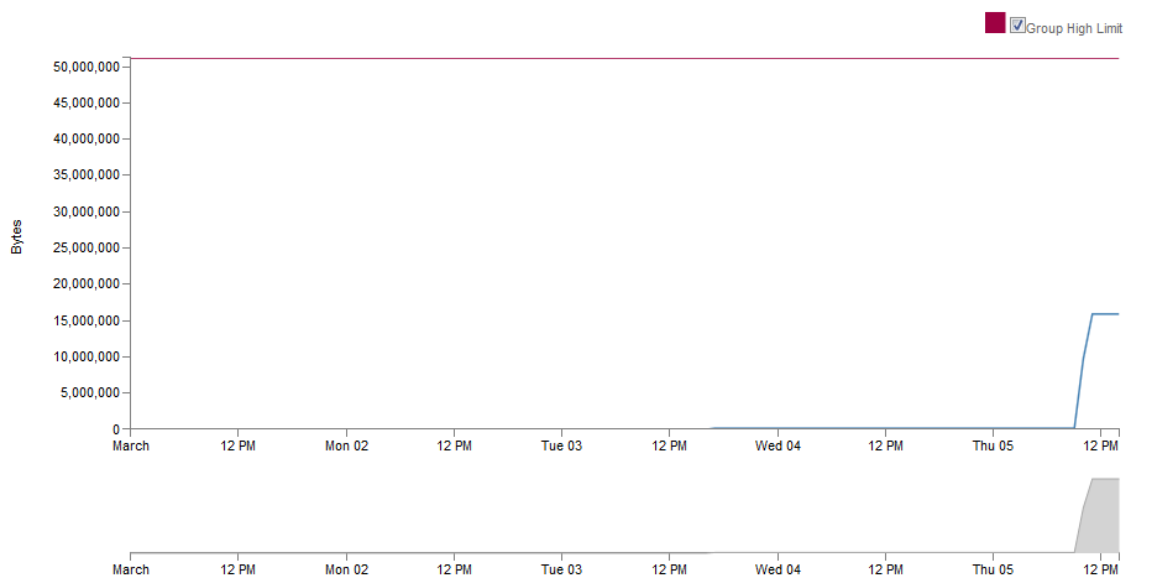
Note Graphs are not supported for reports for which the selected Operation category is "All Requests", reports for which the selected Report Granularity is "Raw", or reports for which the selected Region is "All". For more on graphing functionality see Manipulating Graph Reports.

- **Download CSV** to download a comma-separated value version of the report to your computer.

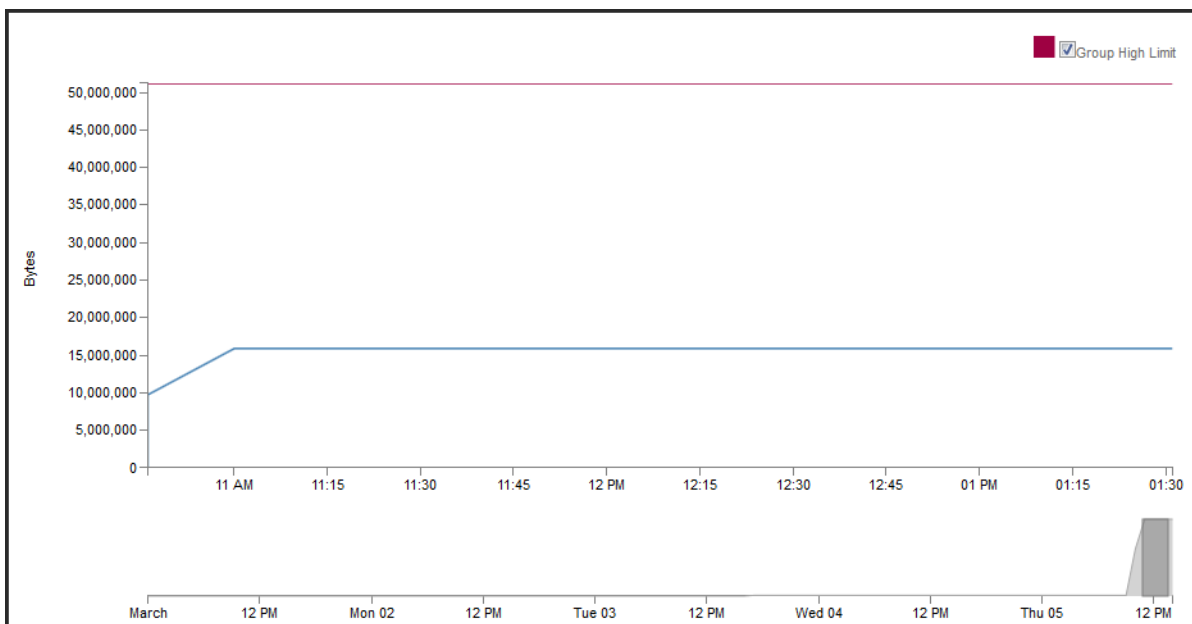
Note In CSV-formatted report output, all data size values are expressed in bytes (rather than KiBs, MiBs, or GiBs).

Manipulating Graph Reports

When you generate a graph report you can manipulate the graph display.



In the small graph at the bottom of the page click and drag horizontally to create a dark grey block. You can then click and drag the edges of the block to expand or contract the time period shown in the main graph. A narrower block provides a more granular view while a wider block provides a less granular view. You can also click the block's center and drag the block to shift the main graph to an earlier or later time interval (within the bounds of the Time Period that you selected when generating the graph).



For reports on usage for a single group, the graph will show a horizontal line that indicates the quality of service (QoS) limit for that group, for the service metric that's the focus of the report (storage bytes, for instance). For reports on usage for a single user, two horizontal lines display in the graph — one indicating the user's QoS limit and one indicating the user's group's QoS limit. You can uncheck the QoS display boxes to hide these lines.

Note Hiding the QoS lines will be desirable if the current usage indicated by the graph is just a small fraction of the QoS limit. When QoS lines are included, the graph's Y axis will necessarily scale up to include the QoS limit level, which can make it hard to view variation within the usage level if usage is confined to just a small portion of the Y axis. Without the QoS lines the Y axis will scale appropriately to the usage level.

The **metric used on a graph's Y-axis auto-scales** to units that are most appropriate for the particular quantities being conveyed. Specifically, for a given report the Y-axis may be expressed in terms of bytes, KiBs, MiBs, or GiBs, depending on the usage level during the full 30-day graphing interval. Pay attention to the Y-axis label to see what metric is being used.

If your **report is based on granular data but covers a long time period**, the labels on the X-axis will be less granular than the source data. For example, if you generate a report based on hourly granularity and a month-long reporting period, the X-axis labels will indicate days not hours. However, the graph content — the trend line itself — will be based on hourly data points. In a month with 30 days, 720 hourly data points (30 X 24) will go into determining and drawing the trend line. Note that if you use the click-and-drag controls (below the graph) to zoom in on a shorter period of time — such as a day or a portion of a day — then the X-axis labels will show hours rather than days.

6.3.3.2. Review Usage Report Output

This section clarifies the meaning of the data that you will see in your service usage reports. Below is an example of a "List" style report, with hourly granularity.

USAGE REPORTS

Group Name:

gr01

User:

user001

Operation:

Storage-Bytes

Report Granularity:

Hours

Time Period:

Current Billing Period

LIST

GRAPH

DOWNLOAD CSV

RESULTS

REGION	DATE/TIME	USER	GROUP	OPERATION	AVERAGE	MAXIMUM
us-east-1	Apr-21-2021 15:00 +0800	user001	gr01	Storage Bytes	3.2 GiB	4.6 GiB
us-east-1	Apr-21-2021 14:00 +0800	user001	gr01	Storage Bytes	565.0 MiB	565.0 MiB
us-east-1	Apr-21-2021 13:00 +0800	user001	gr01	Storage Bytes	565.0 MiB	565.0 MiB
us-east-1	Apr-21-2021 12:00 +0800	user001	gr01	Storage Bytes	274.0 MiB	1.0 GiB
us-east-1	Apr-21-2021 11:00 +0800	user001	gr01	Storage Bytes	340.9 MiB	1.0 GiB
us-east-1	Apr-21-2021 10:00 +0800	user001	gr01	Storage Bytes	2B	11B
us-east-1	Apr-21-2021 09:00 +0800	user001	gr01	Storage Bytes	565.0 MiB	565.0 MiB
us-east-1	Apr-21-2021 08:00 +0800	user001	gr01	Storage Bytes	565.0 MiB	565.0 MiB
us-east-1	Apr-21-2021 07:00 +0800	user001	gr01	Storage Bytes	565.0 MiB	565.0 MiB
us-east-1	Apr-21-2021 06:00 +0800	user001	gr01	Storage Bytes	565.0 MiB	565.0 MiB

10

NEXT

303

Starting from the left of "List" style reports, the first several columns have consistent meaning across the different types of usage reports:

Region

The service region in which the usage occurred.

Date/Time

- For reports with hourly granularity, this field displays each hour for which activity occurred during the reporting period. For example, in a row with Date/Time "Jan-14-2024 07:00 <UTC offset>" the reported activity is from 07:00 through 07:59.
- For reports with daily granularity, this field displays the day as, for example, "Jan-14-2024 <UTC offset>", and the reported activity is from that day, from midnight to midnight.
- For reports with monthly granularity, this field displays the month as, for example, "Jan-2024 <UTC offset>", and the corresponding activity is from that calendar month.
- For reports generated with granularity "Raw", this field displays the transaction timestamp.

Note Times are in the local time zone of your browser.

Note Usage reports show activity only for completed intervals, not in-progress ones. For example, an hourly report shows activity only for completed hours, not the currently in-progress hour. Likewise, a daily report shows activity for completed days, not the currently in-progress day.

User

- For a report on a specific user's activity, this field displays the user's ID. For system-wide or group-wide reports, this field displays an asterisk.

Group

- Group ID. For a system-wide report this field displays "ALL".

Operation

Operation field values will be one of the following:

- HTTP PUT/POST
- HTTP GET/HEAD
- HTTP DELETE
- Storage Bytes
- Storage Objects

Additional columns display depending on the **Operation** that you selected when you chose your report parameters in the upper part of the screen:

For Storage reports ("Storage-Bytes" or "Storage-Objects" from the **Operation** selection menu), these columns display on the right-hand side of the List report:

Average

- Average storage level during the time period specified in the Date/Time column (in bytes or

number of objects depending on the report type).

Maximum

- Maximum storage level during the time period specified in the Date/Time column (in bytes or number of objects depending on the report type).

Note In usage reports, Storage-Bytes are a measure of **net** bytes and exclude any storage policy overhead. For example, in the case of a 1MiB object that's protected by 3X replication, this counts as 1MiB toward Storage-Bytes -- not 3MiB.

For Data Transfer reports ("Data Transfer-In-Bytes" or "Data Transfer-Out-Bytes" from the **Operation** selection menu -- which in the report results will show as "HTTP PUT/POST" or "HTTP GET/HEAD" in the Operation column), these columns display on the right-hand side of the List report:

Data Transfer

- Amount of data transferred during the time period specified in the Date/Time column.

Average

- Average size of an individual data transfer during the time period specified in the Date/Time column. For example in the case of "Data Transfer-In-Bytes" reports, this would be the total data transferred in during the time period divided by the number of PUT and POST requests processed during the time period, to arrive at the average size of an inbound data transfer during the time period.

Maximum

- Maximum size of an individual data transfer during the time period specified in the Date/Time column. For example in the case of "Data Transfer-In-Bytes" reports, this would be the largest single PUT or POST processed during the time period.

For Requests reports ("Get/Head Requests", "Put/Post Requests", "Delete Requests", or "All Requests" from the **Operation** selection menu), this column displays on the right-hand side of the List report:

HTTP Requests

- Request count for the time period specified in the Date/Time column.

6.3.4. Object Locator

Path: **Analytics** → **Object Locator**

The screenshot shows the Cloudian Object Locator interface. The top navigation bar includes 'Analytics' (1) and 'Object Locator' (2), both highlighted with red boxes. Below the navigation bar, there are tabs for 'Cluster Usage', 'Capacity Explorer', 'Usage By Users & Groups', and 'Object Locator'. The 'Object Locator' tab is active. The main form is titled 'OBJECT LOCATOR' and contains three input fields: 'Bucket Name', 'Object Name', and 'Version:(optional)'. A green 'FIND' button is located at the bottom right of the form.

Supported task:

- View storage location information for an object

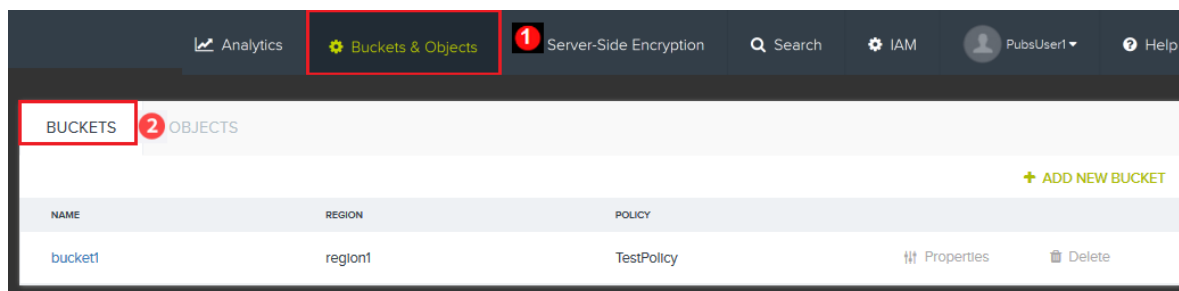
To view storage location information for an object:

1. Enter the bucket name. For example, *bucket1*. This field is case-sensitive.
2. Enter the full object name including "folder" path, if any. For example, *file1.txt* (for a file at the root level of the bucket) or *Videos/Vacation/Italy-2024.mpg*. This field is also case-sensitive.
3. Optionally, if versioning is enabled on the bucket that contains the object, enter the version ID of a particular version of the object. Version IDs are system-generated hexadecimal values (for example, *fe1be647-5f3b-e87f-b433-180373cf31f5*). If versioning has been used for the object but you do not specify a version ID in this field, location information will be retrieved for the most recent version of the object.
4. Click **Find**.

This function executes the *hsstool whereis* command. The results that display are the same as those that display if you run the *hsstool whereis* command on the object (on the **Cluster** → **Nodes** → **Advanced** page). For description of the result elements, see [hsstool whereis](#).

6.4. Buckets

Path: **Buckets & Objects** → **Buckets**



Supported tasks:

- ["Add a Bucket"](#) (page 307)
- ["Set Bucket Properties"](#) (page 309)
- ["Delete a Bucket"](#) (page 343)

Note System administrators do not have their own S3 account credentials and therefore cannot use the **Buckets & Objects** page for their own data storage purposes. However, system admins can access the **Buckets & Objects** page on behalf of regular service users, via the [Manage Users](#) page (if the [CMC is configured to allow administrator access to users' data](#)). System admins can also access the **Buckets & Objects** page by creating a regular user account for themselves (in the [Manage Users](#) page) and then logging into the CMC as a regular user.

6.4.1. Add a Bucket

A "bucket" is a logical container in which you can store data objects — comparable to a root folder in a conventional file system. You must create at least one bucket before you can store any data objects. Optionally you can have more than one bucket.

Note By default each user is allowed a maximum of 100 buckets. You can change this setting in the CMC's [Configuration Settings](#) page.

Note By default administrators cannot view or manage user's buckets and objects through the CMC. To change this and allow such access, use the configuration setting "[cmc.ui.admin.manageUsers.view.objectDataGrantees](#)" (page 739).

To create a bucket:

1. In the **Buckets** list view, click **Add New Bucket**. This opens the bucket creation interface.

2. Enter a bucket name. Using underscores in bucket names is not recommended.

Bucket naming restrictions

Your bucket name must meet these restrictions:

- Must be globally unique across the HyperStore service. If you choose a bucket name that's already been taken, an error message will display in the UI.
- Must be at least 3 and no more than 63 characters long.
- Only lower case letters (a-z), numbers (0-9), dash (-), period (.), and underscore (_) are allowed (although underscores are not recommended; see below). The use of the non-alphanumeric characters -- dash, period, or underscore -- is subject to certain restrictions.

A valid bucket name:

- Must start with a letter or a number.
- Must not end with a dash or a period.
- Must not contain two or more adjacent periods.
- Must not contain dashes next to periods (e.g., "my-.bucket.com" and "my-.bucket" are invalid).

- Must not be in the form of an IPv4 address (e.g "192.168.5.4").
 - Must not contain underscores if the bucket is being created in a non-default region of the S3 service. Underscores are allowed in bucket names for buckets created in the default service region. However, **if you use an underscore in a bucket name you will not be able to use 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.
3. From the "Region" drop-down list, select the service region in which you want the bucket to be created. If your system has only one service region, that will be the only region listed.
 4. From the "Storage Policy" drop-down list, select a storage policy to apply to the data that will be stored in this bucket. A storage policy is a method for protecting data against loss or corruption. The policies are pre-configured by system administrators. When you select a policy from the drop-down list a brief policy description displays.

If you do not select a storage policy the system default storage policy is automatically applied to the bucket. In the list of policies, the system default policy is the one listed first and prefixed by an asterisk (*).

Note After a bucket is created, its **storage policy assignment cannot be changed**. 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.

5. If Object Lock is supported in the system, you have the option to enable Object Lock on the new bucket. By default Object Lock is disabled when you create a new bucket. Object Lock can guard against accidental or malicious deleting of objects after the objects have been uploaded to the bucket. **If you enable Object Lock on the bucket:**
 - [Versioning](#) will automatically be enabled on the bucket as well. When versioning is enabled on a bucket, if you upload an object and then you later upload one or more modified versions of the same object (with the same object name), all the versions of the object are stored in the bucket.
 - If you create a bucket with Object Lock enabled, you **cannot disable Object Lock or suspend versioning** for the bucket (even if you never apply a default Object Lock configuration to the bucket).
 - After bucket creation you will be able to set a default Object Lock configuration for the bucket that will enforce a retention period on each version of each object that gets uploaded to the bucket. For details see ["Configure Object Lock Properties for a Bucket"](#) (page 337).
 - After bucket creation you will be able to set Object Lock attributes on individual objects that you have uploaded to the bucket, which will override the default Object Lock configuration for the bucket. For details see ["Set Object Lock Attributes on an Object"](#) (page 356).
 - This bucket will not be able to serve as the source bucket for [auto-tiering](#) or [cross-region replication](#).

Note

- For the **requirements** that your HyperStore system must meet in order for Object Lock to be used in the system, see ["Preparing the Object Lock Feature"](#) (page 224).
- Enabling Object Lock on a bucket is only supported **as you create the bucket**. You

cannot enable Object Lock on an already existing bucket, after you have completed the creation of the bucket.

- Enabling Object Lock on a bucket **does not by itself have the effect of locking objects that are subsequently uploaded** into that bucket. For uploaded objects to be locked, you must set a default Object Lock configuration for the bucket or set Object Lock attributes on individual objects that you have uploaded to the bucket.

6. Click **Create** to create the bucket.

The newly created bucket will then appear in the **Buckets** list view. If you enabled Object Lock on the bucket, a padlock icon will appear beside its name.

BUCKETS

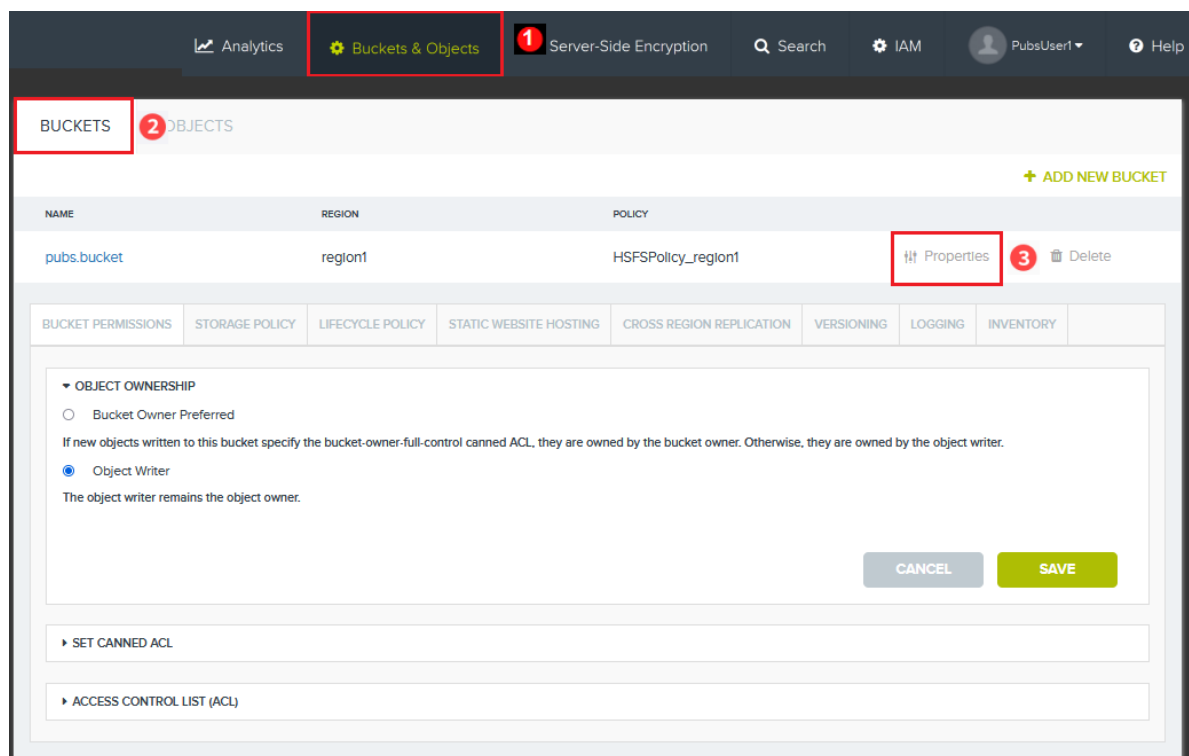
OBJECTS

+ ADD NEW BUCKET

NAME	REGION	POLICY		
 objectlockbucket2	ks-region1	testp	 Properties	 Delete
unlockedbucket	ks-region1	testp	 Properties	 Delete

6.4.2. Set Bucket Properties

Path: **Buckets & Objects** → **Buckets** → **Properties**



The screenshot shows the AWS S3 console interface. At the top, the navigation bar includes 'Analytics', 'Buckets & Objects' (highlighted with a red box and a red '1'), 'Server-Side Encryption', 'Search', 'IAM', 'PubsUser1', and 'Help'. Below the navigation bar, the 'BUCKETS' tab is selected (highlighted with a red box and a red '2'). The main content area displays a table of buckets. The first bucket, 'pubs.bucket', is highlighted. To its right, the 'Properties' link is highlighted with a red box and a red '3'. Below the table, there are tabs for 'BUCKET PERMISSIONS', 'STORAGE POLICY', 'LIFECYCLE POLICY', 'STATIC WEBSITE HOSTING', 'CROSS REGION REPLICATION', 'VERSIONING', 'LOGGING', and 'INVENTORY'. The 'OBJECT OWNERSHIP' section is expanded, showing two radio buttons: 'Bucket Owner Preferred' and 'Object Writer' (selected). Below this, there are sections for 'SET CANNED ACL' and 'ACCESS CONTROL LIST (ACL)'. At the bottom right, there are 'CANCEL' and 'SAVE' buttons.

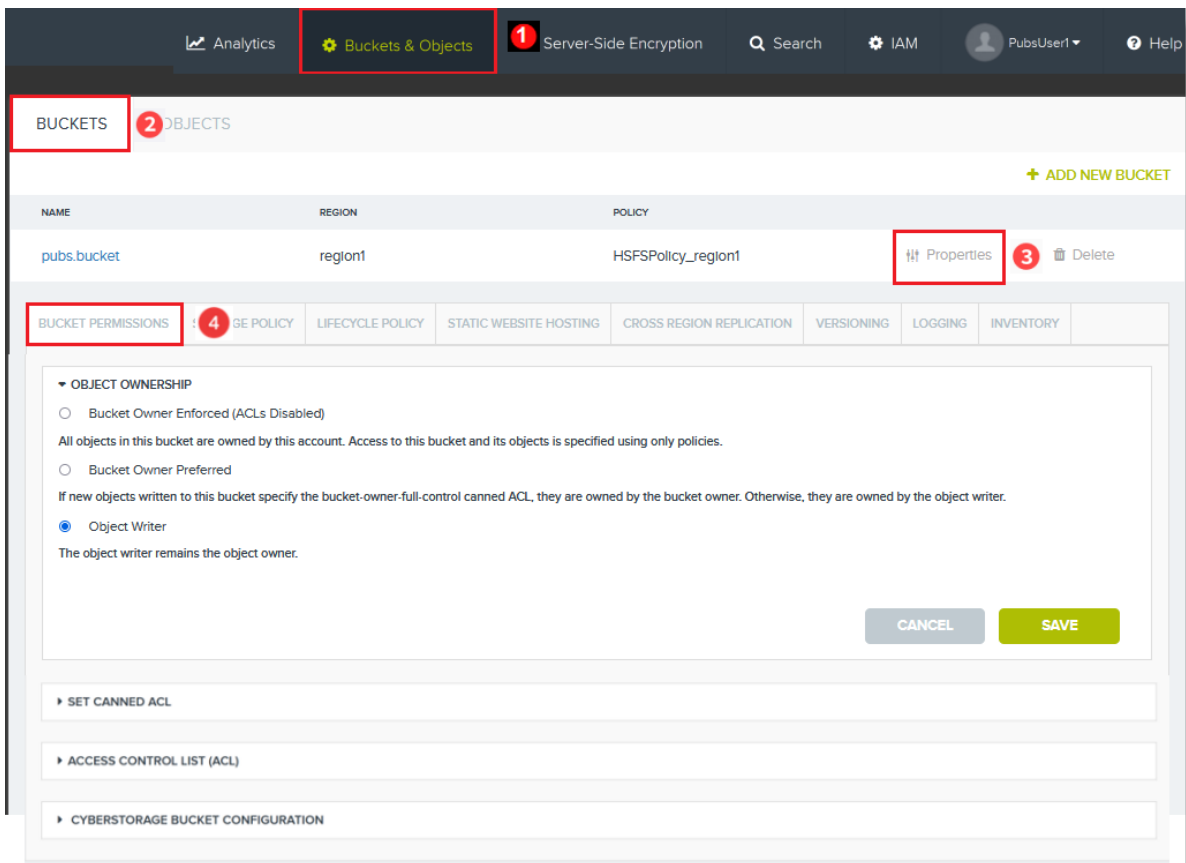
In the **Buckets** page click the **Properties** link for the bucket that you want to work with.

Supported tasks:

- ["Configure Bucket Permissions"](#) (page 310)
- ["View a Bucket's Storage Policy"](#) (page 315)
- ["Configure a Bucket Lifecycle Policy"](#) (page 315)
- ["Configure a Bucket as a Static Website"](#) (page 328)
- ["Configure Cross-Region Replication for a Bucket"](#) (page 331)
- ["Enable/Disable Versioning for a Bucket"](#) (page 333)
- ["Enable/Disable Bucket Logging"](#) (page 334)
- ["Configure Object Lock Properties for a Bucket"](#) (page 337)
- ["Configure Bucket Inventory Reporting"](#) (page 339)

6.4.2.1. Configure Bucket Permissions

Path: **Buckets & Objects** → **Buckets** → **Properties** → **Bucket Permissions**



[As is the case with Amazon S3](#), the HyperStore S3 Service supports several ways of managing permissions for S3 buckets and objects: IAM policies, bucket policies, and S3 ACLs.

- The CMC supports creating IAM users, groups, and policies as described in ["IAM"](#) (page 435).
- The CMC does not support creating bucket policies, but HyperStore's S3 API does support this, so you can use a third party S3 client application to create a bucket policy for a bucket if you wish.
- The CMC supports configuring S3 ACLs for a bucket. You can do so in the **Bucket Properties** interface's **Bucket Permissions** tab. In particular, on this tab you can:

- [Set object ownership](#) for objects uploaded to this bucket by users other than you, the bucket owner (users with permissions to upload to the bucket)
- [Set a "canned" ACL configuration](#) for the bucket
- [Set custom ACLs](#) for the bucket

This screen also supports "[Setting Cyberstorage Bucket Configuration](#)" (page 314)

Note that as the owner of a bucket you always have full bucket-level permissions, which consist of:

- Permission to view a list of the contents of the bucket
- Permission to write to the bucket and delete objects from the bucket
- Permission to view and change the bucket's ACL settings

Note You can use the CMC to configure ACL permissions on your bucket, but your grantees -- users to whom you grant ACL-based permissions -- cannot use the CMC to exercise those permissions. The CMC only allows a bucket's owner to access the bucket. The CMC does not allow users to access a bucket that they do not own, even if the bucket owner has granted them permissions on that bucket. However, most other S3 client applications allow users to access buckets to which they have been granted ACL permissions by the bucket owner. So, your grantees can use S3 applications other than the CMC to exercise ACL permissions that you grant them.

Object Ownership

Choose between:

- **Bucket Owner Enforced** -- This setting disables ACLs for the bucket and for objects uploaded to the bucket, so that all objects are owned by the bucket owner. Access to objects is controlled entirely by policies (bucket policies and IAM policies). *This is the recommended setting*, for optimal security.
- **Bucket Owner Preferred** -- If objects written to this bucket specify the *bucket-owner-full-control* canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.
- **Object Writer** -- The object writer (the account uploading the object to this bucket) always remains the object owner. The object writer remains the object owner even if the object specifies the *bucket-owner-full-control* canned ACL.

After making a selection, click **Save**.

Set Canned ACL

From the drop-down list, choose the canned ACL to assign to the bucket. You can only choose one.

Canned ACL	Description
Private	This bucket canned ACL grants no one any permissions on the bucket (so that only the bucket owner has permissions). This is the default.
Public Read	This bucket canned ACL grants "the public" -- any S3 application user, regardless of whether or not the user is a registered user of the Clodian HyperStore service -- permission to read a list of objects that are in the bucket.
Note The bucket read permission is not inherited by objects	

Canned ACL	Description
	within the bucket. Giving a grantee the bucket read permission does not give the grantee permission to read (open or download) objects in the bucket. It only gives the grantee permission to read a list of the bucket contents. By default an object can only be read by its owner (the user who uploaded the object to the bucket). An object owner can grant other users permission to read the object by setting ACL permissions on the object. If you own the object and you also own the bucket in which the object is stored, you can use the CMC's Object Properties interface to set per-object ACL permissions (see "Set Object Properties" (page 348)). To make large numbers of objects (or all the objects in a bucket) readable to users who don't own the objects, the most efficient way is to use IAM policies (see "IAM" (page 435)) or bucket policies (which are not supported by the CMC, but are supported by HyperStore's S3 Service API and by most third party S3 client applications.)
Public Read Write	This bucket canned ACL grants the public permission to read a list of files that are in the bucket, and also to upload files to the bucket or delete files from the bucket. See the Note in the "Public Read" description regarding the limits of the bucket read permission.
Authenticated Read	This bucket canned ACL grants all registered HyperStore users permission to read a list of files that are in the bucket. See the Note in the "Public Read" description regarding the limits of the bucket read permission.
Log Delivery Write	This canned ACL enables the HyperStore system to store the bucket's server access logs in the bucket.

After selecting a canned ACL for the bucket, click **Save**.

Note A bucket can have only one canned ACL attached to it at a time. If you apply one canned ACL to a bucket, and then you later apply a different canned ACL, the first canned ACL will be removed.

Access Control List (ACL)

To configure custom ACLs that grant bucket permissions to other users, follow the instructions below.

In the Grantee column are the different persons or groups of persons to whom you can grant permissions for the bucket.

Grantee	Description
Public	The general public, including persons who are not registered users of the Clodian HyperStore service. This is essentially any person with an S3 client application and knowledge of the HyperStore S3 service endpoint (URL) and the bucket name.
Authenticated Users	All registered users of the Clodian HyperStore service .
Log Delivery	Give this grantee the "Writable" permission if you want the HyperStore system to store the bucket's server access logs in the bucket. This is the only use for this grantee.
Specific group or user	If you want to grant permissions to a specific HyperStore user group or a specific user, click Add New. A new grantee row appears, with an empty grantee name box. In the name box do either of the following: • If the new grantee is a group, enter the group name followed by a vertical bar (with no space in between). For example, <i>Engineering </i>. This is to grant permissions to all users who belong to the group. • If the new grantee is a user, enter the group name followed by a vertical bar and then the user name (with no spaces in between). For example, <i>Engineering Balaji</i>. Note that this must be a HyperStore account root user -- it cannot be an IAM user. Note Group and user name matching are case-sensitive, so be sure to use the correct case when specifying the grantee. To configure permissions for multiple group or user grantees, click Add New for each one.

For each grantee you can select one or more of the following bucket permissions:

Permission	Description
Readable	With this bucket permission, the grantee can read a list of files that are in the bucket. Note The bucket permission "Readable" is not inherited by objects within the bucket. Giving a grantee the bucket permission "Readable" does not give the grantee permission to read (open or download) objects in the bucket. It only gives the grantee permission to read a list of the bucket contents. By default an object can only be read by its owner (the user who uploaded the object to the bucket). An object owner can grant other users permission to read the object by setting ACL permissions on the object. If you own the object and you also own the bucket in which the object is stored, you can use the CMC's Object Properties interface to set per-object ACL permissions (see).

Permission	Description
	To make large numbers of objects (or all the objects in a bucket) readable to users who don't own the objects, the most efficient way is to use IAM policies (see "IAM" (page 435)) or bucket policies (which are not supported by the CMC , but are supported by HyperStore's S3 Service API and by most third party S3 client applications.)
Writable	With this bucket permission, the grantee can upload objects to the bucket, replace existing objects in the bucket, and delete objects from the bucket.
ACP Readable	With this bucket permission, the grantee can view the current permission settings for the bucket.
ACP Writable	With this bucket permission, the grantee can change the permission settings for the bucket.

After making the permissions selections, click **Save**.

Editing Custom Bucket Permissions

After you've configured custom permissions for a bucket, you can subsequently modify the permissions configuration by returning to the **Bucket Permissions** tab for the bucket, selecting or deselecting permissions, and then clicking **Save**.

If you had previously added a group or user grantee and you now want to delete that grantee, you can do so by clicking **Delete** on the right of that grantee row and then clicking **Save**. Be sure to click **Save** after clicking **Delete** -- otherwise the grantee will not actually be deleted (the grantee row will disappear when you click **Delete**, but if you do not click **Save** the grantee will still have its permissions and if you leave the **Bucket Permissions** tab and then return to it, the grantee row will be there again).

Note If in your list of Grantees you see a row labeled **"Deleted User(s)"**, this means that one or more users to whom you previously granted permissions have been deleted from the system (they are no longer users of the storage service). You can delete this row by clicking **Delete** to the right of the row.

Setting Cyberstorage Bucket Configuration

In the current release of the object storage system, there is only one Cyberstorage bucket configuration setting: "Block Overwrite Configuration". This setting determines whether or not the system will block S3 *PutObjectCopy* operations that are attempting to copy an object over itself, with the same content except encrypted by SSE-C encryption. This is a potential attack vector for ransomware attacks, once an S3 credential has been stolen. The targeted objects would be encrypted in place, and with an SSE-C passphrase known only to the attackers.

System administrators set a system-wide default setting for this feature. If the "Block Overwrite Encryption" checkbox shows a gray checkmark, this means that by default the system is blocking these types of S3 requests (and unchecked means that by default the system is not blocking such requests).

If you want to have a different setting for your bucket than the system default setting, do the following:

1. Check the "Enable Cyberstorage Bucket Configuration" checkbox.
2. Click **Save**.

3. Make your desired change to the "Block Overwrite Encryption" checkbox (check it or uncheck it).
4. Click **Save** again.

6.4.2.2. View a Bucket's Storage Policy

Path: **Buckets & Objects** → **Buckets** → **Properties** → **Storage Policy**

The screenshot shows the AWS Management Console interface. The top navigation bar includes 'Analytics', 'Buckets & Objects' (highlighted with a red box and a red circle 1), 'Server-Side Encryption', 'Search', 'IAM', 'PubsUser1', and 'Help'. The main content area has a left sidebar with 'BUCKETS' (highlighted with a red box and a red circle 2) and 'OBJECTS'. The main panel shows a table of buckets with columns: NAME, REGION, POLICY, and Actions. A row for 'bucket1' in 'region1' with policy 'TestPolicy' is shown. The 'Properties' link in the Actions column is highlighted with a red box and a red circle 3. Below the bucket list, there are tabs for 'BUCKET PERMISSIONS', 'BUCKET CANNED ACL', 'STORAGE POLICY' (highlighted with a red box and a red circle 4), 'LIFECYCLE POLICY', 'STATIC WEBSITE HOSTING', 'CROSS REGION REPLICATION', 'VERSIONING', 'LOGGING', and 'INVENTORY'. The 'STORAGE POLICY' tab is active, showing a table with columns: NAME, DESCRIPTION, DATA DISTRIBUTION POLICY, NO OF REPLICAS, and EC K-M VALUE. The row for 'TestPolicy' shows '3X Replication', 'Single DC', '3', and 'N/A'.

When you first create a bucket, you assign it a "storage policy" — a method by which data in the bucket will be protected against loss or corruption. Subsequently you can select the **Storage Policy** tab of the **Bucket Properties** interface to view information about the storage policy that the bucket uses.

The display includes the storage policy name and description, whether data in the bucket is stored in one data center or multiple data centers, and whether data in the bucket is replicated or erasure coded.

You can control which user types can view this tab -- system admins, group admins, and/or regular users -- with the configuration setting "[cmc.ui.bucket.view.storagePolicyGrantees](#)" (page 740). By default all user types can view this tab.

Note You cannot change a bucket's storage policy assignment.

6.4.2.3. Configure a Bucket Lifecycle Policy

Path: **Buckets & Objects** → **Buckets** → **Properties** → **Lifecycle Policy**

The screenshot shows the AWS Management Console interface, similar to the previous one. The top navigation bar and sidebar are the same. The 'Properties' link in the bucket actions column is highlighted with a red box and a red circle 3. The 'LIFECYCLE POLICY' tab is highlighted with a red box and a red circle 4. Below the tabs, there is a section for 'LIFECYCLE POLICY' with an 'ADD NEW RULE' link.

Introduction

The HyperStore system supports bucket lifecycle policies. For each of your buckets, you can configure a lifecycle policy that consists of one or more rules for:

- **Auto-Tiering** -- Objects in the bucket can be automatically moved to a different S3 compatible object storage system on a schedule that you define.
- **Expiration** -- Objects in the bucket can be automatically deleted on a schedule that you define.

Both types of lifecycle rules can help to limit the amount of local storage space that the bucket will consume over time. Expiration lifecycle rules also help to limit the total number of objects that must be returned when you execute a "list objects" operation for the bucket (through the CMC or another S3 client application), so that good performance for this operation is maintained.

Note that:

- You can also create combination lifecycle rules such that objects are auto-tiered to a different storage system and then later are automatically deleted from that system.
- A lifecycle rule can apply to all objects in a bucket, or to a subset of objects as filtered by object name prefix and/or by object size. You also have the option of creating multiple lifecycle rules for a bucket, with different rules applying to different sub-sets of objects.
- **If the source bucket has [versioning](#) enabled:**
 - You can configure different schedules for current object versions and non-current object versions (object versions that have been superseded by a newer version). **It's important to consider how you want the system to handle non-current object versions**, particularly for a bucket for which you expect a large volume of objects and activity. By configuring an expiration rule for non-current versions you can prevent old non-current object versions from accumulating over time, consuming storage space and potentially impacting object listing operation performance.
 - For auto-tiering from a versioning-enabled source bucket, the tiering destination bucket must also have versioning enabled. Note that:
 - If you select to have HyperStore create the tiering destination bucket (this option is described in the instructions further below), HyperStore will enable versioning on the destination bucket after creating it.
 - If you select to use an existing bucket in the destination system as the tiering target (this option is also described in the instructions further below), HyperStore will enable versioning on the destination bucket if versioning is not already enabled.
 - You cannot use Azure, Google Cloud Storage, or Spectra BlackPearl as the tiering destination system if your source bucket in HyperStore has versioning enabled, because these storage systems do not support S3 compatible object versioning.

IMPORTANT ! After configuring auto-tiering from a versioning-enabled source bucket, **do not turn off versioning in the destination bucket**. If you do, then when object versions are tiered from the source bucket to the destination bucket, any older versions of the objects in the destination bucket will be overwritten (rather than being saved along with the newer object versions, as occurs when versioning is enabled). This will result in the irrecoverable loss of the older object versions.

- By default system configuration, the auto-tiering feature is disabled in the CMC. For instructions on enabling this feature in the CMC, and controlling which auto-tiering options are presented to users, see

"Preparing the Auto-Tiering Feature" (page 209). Even if you've enabled tiering in your system, tiering is not allowed for these types of source buckets:

- Buckets that have an underscore in their name.
- Buckets for which [object lock](#) is enabled.
- Buckets that are source buckets for [cross-region replication](#).

Creating a Bucket Lifecycle Rule

1. In the **Bucket Properties** interface, select the **Lifecycle Policy** tab and then click **Add New Rule**. The **Add New Bucket Lifecycle Rule** interface displays.

2. Enter a descriptive "Rule Name" for the rule you are creating. Spaces are allowed in the name.
3. Optionally enter the "Object Prefix" for the subset of objects for which you want to create a lifecycle rule. To have the lifecycle policy apply to all objects in the bucket, leave the "Object Prefix" field empty.

Note If you intend to use "Bridge Mode" (whereby objects are auto-tiered immediately after being uploaded to HyperStore), leave the "Object Prefix" field empty. Bridge Mode does not support filtering by prefix.

If entering an object prefix, the path should start from the root level of your bucket, and you do not need to include a leading forward slash ("/"). You should however include a trailing forward slash to demarcate the "folder" name from its "contents" (which will be subject to the lifecycle policy that you're configuring). For example:

Projects/archived/2024/

This object prefix would apply to all objects with names that start with *Projects/archived/2024/*, such as *Projects/archived/2024/01/JanuaryOverview.docx* and *Projects/archived/2024/02/FebruaryOverview.docx*.

4. Select:
 - The "Enable Tiering" checkbox, if you want objects to be **transitioned to a different S3 compatible storage system** on a schedule
 - The "Expire Objects" checkbox, if you want objects to be **deleted** on a schedule (and/or to configure deletion of incomplete multipart uploads and expired object delete markers)
 - Both checkboxes if you want to configure a rule that transitions objects to a different storage system and then later deletes them from that system. Note that if you configure a rule such that some objects become eligible for transition and deletion at the same time, deletion takes precedence (such objects will be deleted without being transitioned).
5. Configure the rule attributes:

Object Tiering Schedule and Other Tiering Details

OBJECT TIERING BUCKET LEVEL SETTING

DESTINATION

☐ AWS DEEP ARCHIVE

☐ AWS GLACIER

☒ AWS S3

☐ Azure

☐ Google

☐ Custom Endpoint

TIERING CREDENTIAL

Endpoint:

Access Key:

Secret Key:

Bucket Name:

☐ Retain Local Copy

☐ Bridge Mode (Proxy)

GET REQUEST HANDLING

☒ Stream

☐ Require Restore

☐ Cache (Stream & Restore)

Schedule: Current Version

☒ **CURRENT VERSION**

☐ DAYS AFTER CREATION DATE ☒ AFTER DATE:

Select the Current Version checkbox to set a tiering schedule for the current version of objects in the bucket. If your bucket does not use [versioning](#), then current versions are the only versions of objects in your bucket and this is the appropriate type of schedule to set. If your bucket does use versioning, select this checkbox to set a tiering schedule specifically for the **current version** of objects in the bucket (you can set a different schedule for non-current versions of objects as described in "Previous Version" below).

- If you want tiering of the object current versions to occur on a particular date, select the "After Date" radio button and enter the desired tiering date.
- If you want tiering of each object current version to occur a certain number of days after the object current version was created, choose the "Days After Creation Date" radio button and enter the number of days.
- If you want tiering of each object current version to occur a certain number of days after the object current version was last accessed (retrieved from local storage by an S3 *GetObject* operation), first go "Lifecycle Rule Bucket Level Setting" section toward the bottom of the interface and select the "Use Last Access Time" radio button. Then return to the Current Version scheduling section, select "Days After Last Access Time", and enter the desired number of days.

Note Whichever type of scheduling reference point you choose -- Creation Date or Last Access Time -- you must use this same type of reference point for all lifecycle rules for the bucket. You cannot configure some lifecycle actions to use Creation Date and others to use Last Access Time for the same bucket.

- If the bucket has versioning enabled and the current version of an object is a delete marker, the delete marker will be tiered according to your current version tiering schedule.

Schedule: Previous Version

☒ **PREVIOUS VERSION**

DAYS SINCE BECOMING NON-CURRENT

Note For important notes about tiering from a source bucket that has versioning enabled see the ["Introduction"](#) (page 316) to the bucket lifecycle configuration procedure.

This option is applicable only if the bucket uses [versioning](#). Select the Previous Version checkbox to set a tiering schedule for **non-current versions** of objects (object versions that have been superseded by a newer version). Tiering scheduling for non-current versions of objects is always based on the number of days since the objects became non-current (the number of days since being superseded by a newer version of the object). Enter the desired number of days.

For example, if you enter **365** here, then if version1 of an object is made non-current by the uploading of version2 of that object on July 13, 2023, then version1 of the object will be auto-tiered a year later (regardless of whether any additional versions of the object were uploaded during the intervening year).

As another example, if you enter **0** as the number of days, then if version1 of an object is made non-current by the uploading of version2 of that object, version1 becomes immediately eligible for auto-tiering (and will be tiered the next time that the system's automatic daily auto-tiering process runs).

Destination

The destination storage system that objects will be transitioned to.

Depending on system configuration (as described in ["Preparing the Auto-Tiering Feature"](#) (page 209)), you may or may not have multiple options to choose from in this section. The system configuration may be such that all auto-tiering goes to the same default destination, in which case you will see the destination name displayed here and there will be no choice for you to make.

Alternatively, the system configuration may be such that you can choose a tiering destination from among several options such as:

- AWS Deep Archive (Glacier Deep Archive)
- AWS Glacier (Glacier Flexible Retrieval)
- AWS S3
- Microsoft Azure
- Google Cloud Storage
- Spectra BlackPearl

Note The destinations and corresponding endpoint URLs that the interface displays here are configurable. For details see ["Configure Tiering Destinations"](#) (page 210).

The system configuration may also be such that a "Tier to Custom Endpoint" option displays, in addition to the destination types listed above. If you choose this Custom option, an editable "Endpoint" field displays in which you can enter the URL of an **S3-compliant** system you want to tier to (the field is populated with `https://s3.amazonaws.com` by default but is editable). For example this could be the URL of a different HyperStore region or system. When specifying a Custom Endpoint, note that:

- You cannot use a Custom Endpoint to tier to Azure or Spectra BlackPearl.
- You cannot use a Custom Endpoint to tier to AWS Glacier or AWS Deep Archive.
- In your Custom Endpoint be sure to include the `https://` part (or `http://` if the destination system does not support SSL).
- If your Custom Endpoint is AWS S3 in a region other than the AWS S3 default region (*us-east-1*) you must include the region in the endpoint, using proper AWS S3 endpoint formatting -- for example `https://s3.us-west-2.amazonaws.com`.

Regarding your selection of a tiering Destination, note that:

- After you Save your lifecycle policy, **you will not be able to change the tiering destination** for this bucket. If you create different auto-tiering rules for different subsets of objects within a single bucket, all such rules must use the same tiering destination system. Even if you subsequently delete all lifecycle rules for this bucket, for this bucket you will not be able to create new lifecycle rules that use a different tiering destination.
- Auto-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. If versioning is currently enabled for the bucket, these destinations will not display as options in the interface. If versioning was enabled for the bucket in the past but is no longer enabled, these destinations may display as options -- but do not use any of these destinations, or there will be errors when the system implements auto-tiering for the bucket and potentially loss of non-current object versions.
- 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. To change this limit consult with Cloudian Support.
- If your auto-tiering destination is a third party cloud storage service such as AWS, Google, or Azure you will incur charges from that provider per the terms of your service agreement with them.

Access Key and Secret Key

(or Account Name and Account Key if tiering to Azure)

For any tiering destination other than a system-default endpoint, you must enter your account security credentials for accessing the destination system. The local system will then use these account credentials when it transitions objects from the bucket to the destination account.

IMPORTANT ! If you subsequently change your security credentials at the tiering destination be sure to edit this tiering rule so that HyperStore has the correct current credentials for accessing the destination.

Bucket Name

(or Container Name if tiering to Azure)

Optionally, 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 have access privileges.
- 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 your bucket lifecycle configuration attempt will fail.

If you leave the Bucket Name field **empty**, 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>`

Note that after you Save your lifecycle policy, **you will not be able to change the tiering destination bucket**. If you create different auto-tiering rules for different subsets of objects within a single bucket, all such rules must use the same tiering destination bucket. **Even if you subsequently delete all lifecycle**

rules for this source bucket, for this source bucket you will not be able to create new lifecycle rules that use a different tiering destination bucket.

Note If you are tiering to a AWS Glacier or AWS Deep Archive destination, and you are having HyperStore create a tiering target bucket for you, the bucket will be created in AWS S3 and will be automatically configured to tier from AWS S3 to AWS Glacier or AWS Deep Archive. If you are tiering to a AWS Glacier or AWS Deep Archive destination, and you are specifying as the target an AWS S3 bucket that already exists, HyperStore will configure that AWS S3 bucket to tier to AWS Glacier or AWS Deep Archive if it is not already configured to do so. In either case, objects move from AWS Glacier or AWS Deep Archive within 24 hours after being tiered from your local source bucket to AWS S3 (since AWS executes its tiering process once a day).

Region

(Applicable only for tiering to Custom Endpoint)

If you are tiering to a Custom Endpoint and you are having the system create a destination bucket for you to tier to (meaning you left the "Bucket Name" field empty or you specified the name of a bucket that does not yet exist), in the "Region" field indicate the destination service region in which you want the bucket created.

Retain Local Copy

Select the "Retain Local Copy" checkbox if you want to keep a local copy of auto-tiered objects for a certain number of days after the objects have been auto-tiered. If you select the checkbox, then a field appears in which you can specify the number of days for which to retain the local copy.

For example, if you select this option and set the retention to 30 days, then each object in your bucket that gets auto-tiered to the destination system will also be retained locally for 30 days after the auto-tiering occurs. After the 30 days the local copy of the object will be automatically deleted.

There is not an option to retain a local copy "indefinitely", but if you wish you can enter a very large number of days.

If you do not select the "Retain Local Copy" checkbox, then by default the local copies of objects will be deleted immediately after the objects have been successfully tiered to the destination system.

Note Even after local copies of auto-tiered objects have been deleted, the system retains local metadata that enables the system include auto-tiered objects when listing the objects in a bucket and to retrieve a copy of the auto-tiered objects from the destination system upon your request.

Bridge Mode (Proxy)

(Applicable only for tiering AWS S3, Google, Azure, or Custom S3 destinations)

Select the "Bridge Mode (Proxy)" checkbox only if, starting from when you Save your auto-tiering rule, you want **objects newly uploaded to the bucket to be immediately transitioned** to the tiering destination system. In this mode HyperStore essentially acts as a proxy service, with uploaded objects only momentarily being held in storage before being automatically moved to the destination system.

If you select the "Bridge Mode (Proxy)" checkbox, then any schedule-based tiering configuration that you set in the object tiering "Schedule" section of the interface will be applied only to objects that are

already in the bucket at the time that you Save this auto-tiering rule. In other words, turning on Bridge Mode tiering for a bucket does not result in the immediate tiering of any objects that are already in the bucket at the time. Instead, those pre-existing objects are subject to the tiering schedule that you configure in the "Schedule" section of the bucket lifecycle interface (if any). By contrast, new objects -- uploaded after you turn Bridge Mode tiering on -- will be immediately tiered.

Note that:

- You cannot specify an Object Prefix in a tiering rule that uses Bridge Mode. Bridge Mode tiering cannot be filtered by Object Prefix.
- You **can** specify an Object Size in a tiering rule that uses Bridge Mode. Bridge Mode tiering can be filtered by Object Size, such that only newly uploaded objects that meet your configured size criteria are immediately transitioned to the tiering destination.

For more information on bridge mode, see ["Bridge Mode \(Proxy Tiering\)" \(page 206\)](#).

GET Request Handling

(Applicable only for tiering AWS S3, Google, Azure, or Custom S3 destinations)

If your tiering destination is AWS S3, Google, or Azure, or a Custom S3 destination, choose how you want the local HyperStore system to handle GET requests for objects that have been transitioned to the tiering destination system. Options are:

- **Stream** -- The local system GETs the object from the tiering destination system and immediately streams it through to the client without storing it in the local system. This is the default for tiering to AWS S3, Google, Azure, and Custom S3 destinations.
- **Require Restore** -- If you choose this option, the only way that S3 client applications will be allowed to access tiered objects is to execute a *RestoreObject* request in order to temporarily restore a copy of the object in local HyperStore storage. For information about how this works in the CMC's **Objects** interface, see ["Retrieve a Tiered Object" \(page 360\)](#).
- **Cache (Stream & Restore)** -- The local system GETs the object from the tiering destination system and immediately streams it through to the client, and also simultaneously 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 rule is configured:
 - If the tiering rule includes a "Days After..." configuration for "Current Version", 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 tiering rule 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.
 - In all other cases, the cached local copy is retained only until the next running of the system's auto-tiering / auto-expiration cron job (which by default runs daily at midnight). This includes for example tiering rules that use an "After Date..." configuration; tiering rules that only tier "Previous Version" objects and not "Current Version" objects; and "Bridge Mode" tiering rules.

With the "Cache (Stream & Restore)" method, during the period while the object is cached locally any subsequent GETs of the object can be served from local storage. After the cache period expires, the local copy is automatically deleted. 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 locally cached).

Note If the bucket uses [versioning](#) you should use either "Stream" or "Cache (Stream & Restore)" as the GET handling method. If you use Require Restore you will not be able to retrieve auto-tiered non-current object versions unless you first delete the current object version. For more information see ["Restoring a Tiered Object Version"](#) (page 362)

Note If you create multiple auto-tiering lifecycle rules for a single bucket, all such rules must use the same GET request handling method.

For objects tiered to AWS Glacier or AWS Deep Archive or Spectra BlackPearl, the "Require Restore" method is always used.

Object Expiration Schedule and Other Expiration Details

OBJECT EXPIRATION

SCHEDULE

☐ CURRENT VERSION
 ☐ PREVIOUS VERSION

CLEAN UP EXPIRED OBJECT DELETE MARKERS AND INCOMPLETE MULTIPART UPLOADS

☐ CLEAN UP INCOMPLETE MULTIPART UPLOADS
 ☐ CLEAN UP EXPIRED OBJECT DELETE MARKERS

Note If you are configuring a lifecycle rule that includes both auto-tiering **and** auto-expiration, be sure to configure the schedules so that the expiration doesn't occur until **after** the auto-tiering. With this type of rule, objects will be auto-tiered to a destination system in accordance with your auto-tiering schedule and then at a later date -- in accordance with your auto-expiration schedule -- the objects will be automatically deleted from the destination system . (and their metadata will be deleted from the local system).

Schedule: Current Version

☒ CURRENT VERSION

☐ 1 DAYS AFTER CREATION DATE
 ☒ AFTER DATE: Aug-02-2022 04:08 AM

Select the Current Version checkbox to set an expiration (deletion) schedule for the current version of objects in the bucket. The impact of deleting the current version of an object depends on whether the bucket uses [versioning](#) or not:

- If the bucket **does not use versioning**, then current versions are the only versions of objects in your bucket and this is the appropriate type of schedule to set. According to your defined

schedule, objects will be permanently deleted from storage.

- If the bucket **does use versioning**, then **schedule-based expiration of the current version of an object will not actually result in the deletion of any object data** from storage. Instead when the current version of the object reaches its expiration date a "delete marker" will be created for the object, so that the delete marker becomes the "current version" of the object and the object data itself becomes a non-current object version. (This is the same thing that happens if a user deletes the current version of an object in a bucket that uses versioning.) **For object data to actually be deleted from a versioning-enabled bucket by auto-expiration, you need to set a schedule for non-current versions** as described in the "Schedule: Previous Version" section below.

When setting a Current Version expiration schedule:

- If you want deletion of the objects to occur on a particular date, select "After Date" and enter the desired deletion date.
- If you want deletion of each object to occur a certain number of days after the object was created, choose the "Days After Creation Date" radio button and enter the number of days.
- If you want deletion of each object to occur a certain number of days after the object was last accessed (retrieved from local storage by an S3 GetObject operation), first go "Lifecycle Rule Bucket Level Setting" section toward the bottom of the interface and select the "Use Last Access Time" radio button. Then return to the Current Version scheduling section, select "Days After Last Access Time", and enter the desired number of days.

Note Whichever type of scheduling reference point you choose -- Creation Date or Last Access Time -- you must use this same type of reference point for all lifecycle rules for the bucket. You cannot configure some lifecycle actions to use Creation Date and others to use Last Access Time for the same bucket.

Schedule: Previous Version

☒ PREVIOUS VERSION

DAYS SINCE BECOMING NON-CURRENT

This option is applicable only if the bucket uses [versioning](#). Select the Previous Version checkbox to set an expiration (deletion) schedule for **non-current versions** of objects (object versions that have been superseded by a newer version or by a "delete marker"). When a non-current version of an object reaches its scheduled expiration, the non-current version is permanently deleted from storage.

Expiration scheduling for non-current versions of objects is always based on the number of days since the objects became non-current (the number of days since being superseded by a newer version or by a "delete marker"). Enter the desired number of days.

For example, if you enter **365** here, then if version1 of an object is made non-current by the uploading of version2 of that object on July 13, 2023, then version1 of the object will be auto-deleted a year later (regardless of whether any additional versions of the object were uploaded during the intervening year).

As another example, if you enter **0** as the number of days, then if version1 of an object is made non-current by the uploading of version2 of that object, version1 becomes immediately eligible for auto-deletion (and will be deleted the next time that the system's automatic daily auto-deletion process runs).

Note If the bucket uses "Object Lock", non-current object versions will not be deleted prior to the completion of the retention period defined in your object lock configuration. If a non-current object version's expiration date (based on your configured bucket lifecycle expiration schedule) falls before the end of its object lock retention period, the system will retain the non-current object version until the end of its object lock retention period and then will automatically delete the non-current object version shortly thereafter.

Clean Up Incomplete Multipart Uploads

When the CMC or other S3 client applications upload large objects into HyperStore, they use a method called multipart upload during which the large object is divided into multiple parts and each part uploaded separately. In some cases a system problem may result in some of the parts being successfully uploaded while others are not. In such cases an incomplete or partial object consumes some storage space in your bucket but the partial object is not readable and will not appear in your bucket contents list.

Select the "Clean Up Incomplete Multipart Uploads" option to have the system delete such incomplete objects a certain number of days after the multipart upload was initiated. You can specify the number of days. A small number of days such as 5 days is a reasonable setting.

Clean Up Expired Object Delete Markers

This option is applicable only if the bucket uses [versioning](#). If in a versioning-enabled bucket the current version of an object is deleted, a "delete marker" takes the place of that object version -- the delete marker becomes the current version -- while the object's older versions are retained in the system (and are retrievable), subject to an expiration rule for non-current versions if you've configured one). If all the older versions are subsequently deleted, that orphaned delete marker remains. Automatic cleanup (deletion) of these expired delete markers can become a part of your bucket lifecycle configuration in either of these ways:

- If you configure an Object Expiration schedule for Current Version or Previous Version, then clean up of expired object delete markers is automatically included in the bucket lifecycle processing. In this case the "Clean Up Expired Object Delete Markers" checkbox is disabled, since this action is automatically included.
- If you do **not** configure an Object Expiration schedule for Current Version or Previous Version, but you just want to have expired delete markers automatically cleaned up on a regular basis, then select the "Clean Up Expired Object Delete Markers" checkbox.

Use Creation Date/Time or Use Last Access Time

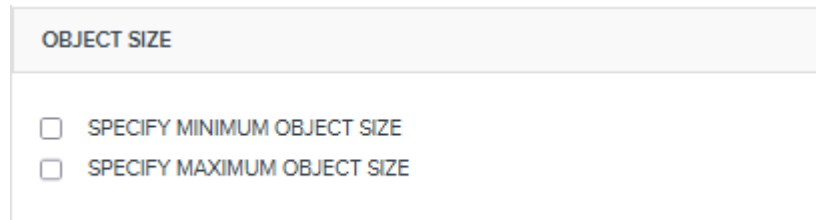
LIFECYCLE RULE BUCKET LEVEL SETTING

☒ Use Creation Date/Time
 ☐ Use Last Access Time

For information about this setting see the descriptions of configuring a Current Version schedule for tiering or expiration, above.

Note Once an object has been tiered to a destination site, any subsequent streaming GET of the tiered object will not affect the object's Last Access Time.

Object Size



OBJECT SIZE

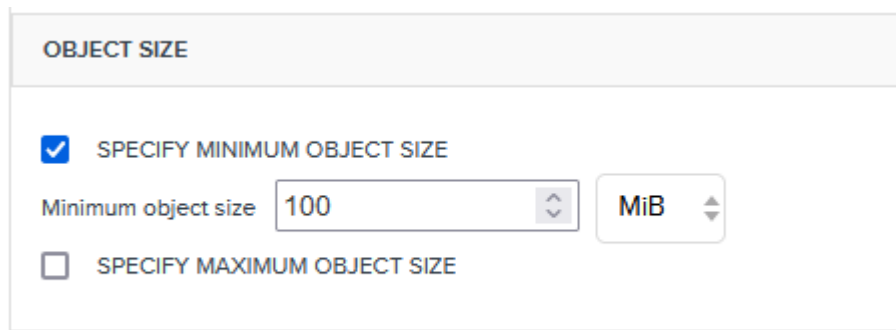
☐ SPECIFY MINIMUM OBJECT SIZE

☐ SPECIFY MAXIMUM OBJECT SIZE

In the Object Size section you can optionally specify a minimum object size and/or maximum object size to which the bucket lifecycle rule will apply:

- To have the rule apply only to objects above a certain size, select and configure just the Minimum Object Size.
- To have the rule apply only to objects below a certain size, select and configure just the Maximum Object Size.
- To have the rule apply only to objects within a size range, select and configure the Minimum Object Size and the Maximum Object Size.

When setting a minimum and/or maximum object size, use the drop-down list to choose a unit of measurement (bytes, KiB, MiB, or GiB). In the example below the lifecycle rule will apply only to objects larger than 100 MiB.



OBJECT SIZE

☒ SPECIFY MINIMUM OBJECT SIZE

Minimum object size

☐ SPECIFY MAXIMUM OBJECT SIZE

Note If the lifecycle rule includes both an object tiering schedule and an object expiration schedule, your object size filter will apply to both the tiering behavior and the expiration behavior.

6. Click **Save** to save the rule.

To confirm that your new rule has been saved in the system, select the **Lifecycle Policy** tab of the CMC's bucket properties interface. The new rule will display in a list along with any lifecycle rules you may have previously configured for the bucket.

IMPORTANT ! Once objects have been auto-tiered to the destination system, **do not overwrite or delete tiered objects directly through the destination system's interfaces** (such as the AWS Console in the case of tiering to AWS). Doing so will cause a discrepancy between the local metadata in HyperStore and the actual data in the destination bucket. If you want to overwrite or delete tiered objects, do so through the CMC or an S3 application accessing the HyperStore S3 Service. In the case of auto-tiering from one HyperStore region to another HyperStore region, any overwriting or deleting of objects should be done through the source bucket not the destination bucket.

Editing a Lifecycle Rule or Adding More Lifecycle Rules for a Bucket

To edit an existing bucket lifecycle rule, select the **Lifecycle Policy** tab of the CMC's bucket properties interface. This displays the current list of lifecycle rules for the bucket.

The screenshot shows the CMC bucket properties interface. At the top, there are tabs for 'BUCKETS' and 'OBJECTS'. Below the tabs, there is a table with columns: NAME, REGION, and POLICY. The first row shows 'bucket1', 'region1', and 'Replication-3X'. To the right of this row are links for 'Properties' and 'Delete'. Below the table, there is a section for 'LIFECYCLE POLICY' with a table of rules. The table has columns: ENABLE, RULE NAME, and ACTION. There are two rules: 'Delete Logs After 90 Days' and 'Auto-Tier Documents After 1 Year'. Each rule has an 'ENABLE' checkbox (checked) and an 'ACTION' column with 'edit' and 'delete' links. At the bottom of the rules section is a '+ ADD NEW RULE' button. At the bottom of the interface are 'CANCEL' and 'SAVE' buttons.

To edit a rule, to the right of the rule name click **edit**. This displays the rule details page where you can edit rule attributes.

For example, if you have an existing tiering rule and you have changed your security credentials at the tiering destination system (such as Amazon or Azure), edit the tiering rule to provide HyperStore with the updated security credentials so HyperStore can continue to access and tier objects to the destination system.

To add more lifecycle rules for the bucket, click **Add New Rule**.

Note these restrictions regarding adding or editing lifecycle rules for a bucket:

- If you add a new tiering rule for a bucket, the new rule must use the same tiering destination system and tiering destination bucket as any existing tiering rules for the bucket. You **cannot have multiple tiering destinations for the same source bucket**.
- If you edit an existing tiering rule for a bucket, you **cannot change the tiering destination system or tiering destination bucket** for the lifecycle rule. While other attributes of an existing rule are editable (such as the schedule and the security credentials), the tiering destination system and destination bucket are not editable.

Note also that if you configure multiple rules and those rules result in some objects becoming eligible for transition and deletion at the same time, deletion takes precedence (such objects will be deleted without being transitioned).

Disabling or Deleting a Lifecycle Rule

To disable or delete an existing bucket lifecycle rule, select the **Lifecycle Policy** tab of the CMC's bucket properties interface. This displays the current list of lifecycle rules for the bucket.

The screenshot shows the CMC interface with the 'OBJECTS' tab selected. Below the bucket name 'bucket1' and region 'region1', there is a table of bucket properties. The 'LIFECYCLE POLICY' tab is active, displaying a list of lifecycle rules. The rules are as follows:

ENABLE	RULE NAME	ACTION
☒	Delete Logs After 90 Days	edit delete
☒	Auto-Tier Documents After 1 Year	edit delete

At the bottom of the rules section, there is a '+ ADD NEW RULE' button. At the bottom right of the interface, there are 'CANCEL' and 'SAVE' buttons.

To disable a rule, to the left of the rule name deselect the **Enable** checkbox, then click **Save**. This causes HyperStore to stop implementing the rule. You can subsequently enable the rule again by selecting the **Enable** checkbox, and HyperStore will resume implementing the rule.

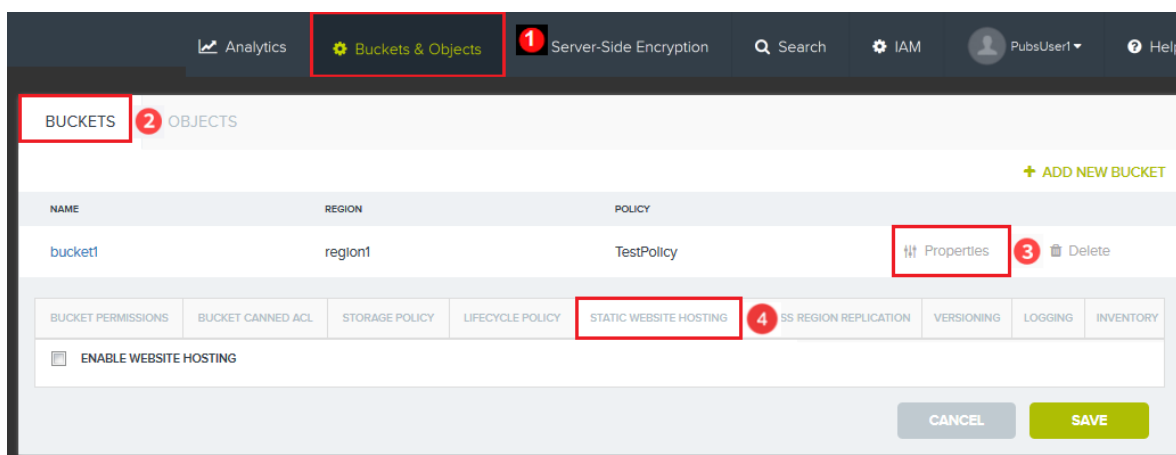
To delete a rule, to the right of the rule name click **delete**. This permanently deletes the rule, and causes HyperStore to stop implementing the rule.

Note that disabling or deleting a lifecycle rule doesn't change anything that has already been done as a result of the rule. So for example, if you disable or delete an auto-tiering rule, any objects that have already been transitioned to the tiering destination system will remain in that system. (For information about locally restoring objects that have been tiered to a destination system, see ["Retrieve a Tiered Object"](#) (page 360).)

Note Even if you delete the existing lifecycle rule(s) for a source bucket, for that source bucket **you cannot create new auto-tiering rules that use a different tiering destination endpoint or a different tiering destination bucket**. Throughout the life of a source bucket it can only have one tiering destination associated with it.

6.4.2.4. Configure a Bucket as a Static Website

Path: **Buckets & Objects** → **Buckets** → **Properties** → **Static Website Hosting**



You can configure a bucket as a website in order to make its content available to public users using regular web browsers (rather than a S3 client applications). Only **static** websites are supported. A static website is a website that serves the same content to all visitors -- there is no content personalization based on cookies or other mechanisms. A static site does not use server-side scripting.

IMPORTANT ! For the static website feature to work on users' buckets, on your DNS server you must create entries for these URIs:

```
s3-website-<region>.<your-domain>
*.s3-website-<region>.<your-domain>
```

For example:

```
s3-website-tokyo.enterprise.com IN A 10.1.1.1
*.s3-website-tokyo.enterprise.com IN A 10.1.1.1
```

For more information see ["DNS Set-Up" \(page 53\)](#).

To configure a bucket as a static website, in the **Bucket Properties** interface for the bucket select the **Static Website Hosting** tab. Then do the following:

1. Select the "Enable Website Hosting" checkbox.
2. By default, when you enable website hosting on a bucket, public read access is automatically granted for all objects in the bucket. A message will pop up to warn you of this behavior -- click **OK** to close the message. If you do not want this default configuration, deselect the "Grant automatic READ access to all objects" checkbox that appears in the **Static Website Hosting** interface. You can do this if you prefer to:
 - Grant public read permissions to individual objects in the bucket, one-by-one. You can do this through the CMC's ["Set Object Properties" \(page 348\)](#) function.

OR

 - Use the S3 API's "PUT Bucket Policy" method to grant public read access to some groups of objects but not others -- such as objects that start with a certain prefix (directory path). For this you would need to use a different S3 client application to do so, since the CMC does not currently support this capability.

The more typical static website set-up would be to accept the default behavior and grant automatic read access to all objects in the bucket.

3. Complete the "Index Document" field with the name of the file that should be served when site visitors access the root URL of your site. This is mandatory. This file is typically named "index.html". You must upload this index file into the **root level** of the bucket if you have not done so already.
4. Optionally, complete the "Error Document" field with the name of a file in which you've composed a custom error response message that you want the system to return when site visitors trigger an HTTP 4xx error (such as by requesting an object that does not exist). You might name this file "error.html", for instance. If you do specify a custom error response message file here, you must upload this file into the **root level** of the bucket if you have not done so already. If you do not specify a custom error response file the system will use a predefined response appropriate to the error type.
5. Click **Save**.

For a bucket configured as a static website, the root URL will be:

```
http://<bucketName>.<regionalS3WebsiteEndpoint>
```

Each Cloudian HyperStore service region has its own website endpoint, set by system configuration. To view the website endpoint(s) for your system, go to the CMC's [Cluster Information](#) page.

Note The HyperStore S3 Service allows static website requests that use HTTPS (TLS). HyperStore does not provide a mechanism for setting up TLS for static websites requests -- you would need to do so outside of HyperStore, by appropriately configuring your TLS certificates. But the HyperStore S3 Service does not prohibit HTTPS access to static websites.

Configure a "Vanity Domain" for an S3 Bucket

The HyperStore system supports the use of "vanity domains", whereby an S3 bucket can be accessed using just the bucket name as the URL, without the service provider's domain name. To use this feature:

- The bucket must be configured as a static website, as described above.
- The bucket name must exactly equal the desired vanity domain name. The bucket name must be DNS-compatible, and must use only lower case letters. For example:

Desired Vanity Domain Name	Required Bucket Name
baseball-stats.com	baseball-stats.com
japan.travel.org	japan.travel.org
never-ending-profits.biz	never-ending-profits.biz

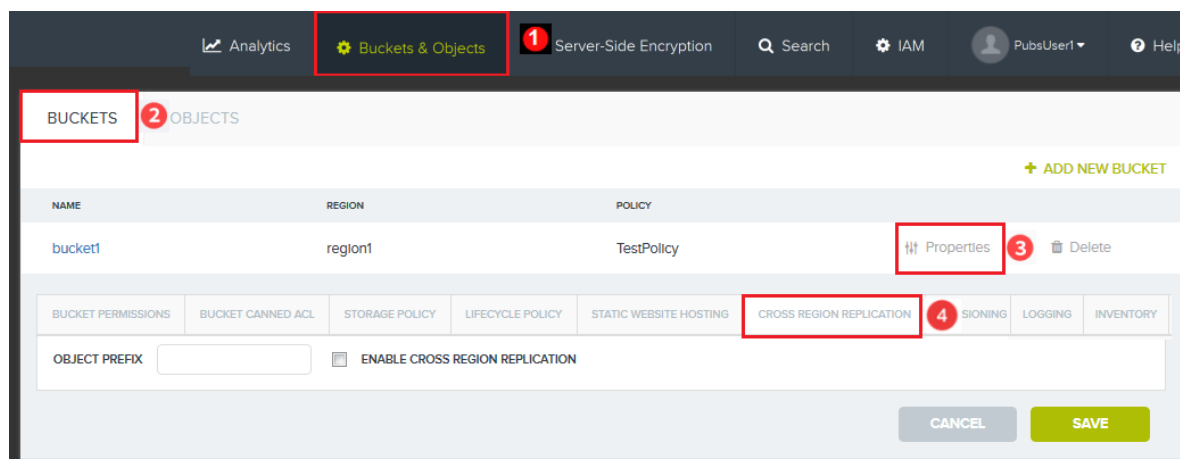
IMPORTANT ! The desired vanity domain must not already exist in the global DNS system or this feature will not work. It's up to end users to ensure that their proposed vanity domains do not yet exist on the web.

- In your DNS set-up, use CNAME to map the vanity domain to the bucket's fully qualified name. For example, if your HyperStore S3 static website endpoint URL is *s3-website-region1.enterprise.com* then you would map vanity domain *baseball-stats.com* to CNAME *baseball-stats.com.s3-website-region1.enterprise.com*.
- In your DNS set-up, use CNAME to map a wildcard of sub-domains of your S3 static website endpoint to the S3 static website endpoint itself. For example, if your HyperStore S3 static website endpoint is *s3-*

website-region1.enterprise.com then you would map wildcard **.s3-website-region1.enterprise.com* to CNAME *s3-website-region1.enterprise.com*.

6.4.2.5. Configure Cross-Region Replication for a Bucket

Path: **Buckets & Objects** → **Buckets** → **Properties** → **Cross-Region Replication**



You can configure a bucket so that any newly uploaded objects (objects uploaded **after** you configure this feature) are automatically replicated to your chosen destination bucket in a different service region within the same HyperStore system. This feature enables you to enhance the protection of your data by having it stored in two geographically dispersed service regions. The feature is also useful in cases where you want to have the same set of data stored in two different regions in order to minimize read latency for users in those regions.

The system also supports replicating to a destination bucket that's in the same service region as the source bucket, if you want to.

Depending on how system administrators have configured the system, the system may also support replication to a destination bucket in an external S3-compliant system such as Amazon Web Services or Cloudbian HyperStore. For general information about this feature see "[Cross-System Replication](#)" (page 216).

Object metadata — including any access permissions assigned to an object — is replicated to the destination bucket as well as the object data itself.

The cross-region replication feature does **not** replicate:

- Objects that were already in the source bucket before you configured the bucket for cross-region replication (except as noted in "[Handling of Pre-Existing Objects](#)" (page 218)).
- Objects that are encrypted with user-managed encryption keys.
- Object version deletions (deletions of specific object versions.)
- Objects that are themselves replicas from other source buckets.

Requirements and restrictions on using cross-region replication:

- Both the source bucket and the destination bucket must have [versioning](#) enabled in order to activate cross-region replication.
- Cross-region replication is not supported for a source bucket that has [object lock](#) enabled.
- Cross-region replication is not supported for a source bucket that has a bucket [lifecycle policy for](#)

[auto-tiering](#) enabled.

- Cross-region replication is not supported for a source bucket that has [KMIP encryption](#) enabled.

To configure a source bucket for replication, in the **Bucket Properties** interface for the bucket select the **Cross Region Replication** tab. Then do the following:

1. Select the "Enable Cross Region Replication" checkbox.
2. If you want all new objects (objects uploaded after you configure this feature) in the source bucket to be replicated, leave the "Prefix" field empty. If you only want to replicate objects whose full names (including path) start with a particular prefix, enter that prefix in the "Prefix" field. You do not need to include a leading forward slash at the start of the prefix. "Taxes/documents" is a valid example, or "pro-file/images/headshots".

If you want to replicate objects associated with multiple prefixes, enter a comma-separated list of prefixes, with no spaces between prefixes. For example "legal/docs,legal/audio,compliance/docs".

Note that if you are specifying multiple prefixes, all the replication must go to the same one destination bucket (you can't replicate from a single source bucket to multiple destination buckets). Note also that the source prefixes will be included in the full name of the object replicas in the destination bucket (for example, source object "legal/docs/briefing_04-17-2023" will be replicated as "legal/docs/briefing_04-17-2023" in the destination bucket).

3. Specify the "Destination Bucket" name. This must be a bucket that you own, within the HyperStore system.
4. (Applicable only if cross-system replication is enabled in your system and you are configuring cross-system replication for your bucket; otherwise skip ahead to Step 5). Specify the "Destination Endpoint". This is the external destination system's complete URL. When you complete this field, two additional fields will display for you to enter your "Secret Key" and "Access Key" (your security credentials for the external system, which will be needed to access and write to the destination bucket in the external system).

Note Microsoft Azure and Google Cloud Storage do not have native S3 support and are not valid destinations for cross-system replication. Note also that object ACLs will be replicated only if the destination is a Cloudian HyperStore system, and only for "canned" ACLs or permissions granted to S3 "predefined groups" ('Authenticated Users' or 'All Users / Public').

5. Click **Save**.

Editing a Source Bucket's Replication Configuration

After cross-region replication is enabled on a source bucket and has been active, you can optionally use the bucket properties' **Cross Region Replication** tab to edit a source bucket's replication configuration. For example you could change the destination bucket used for replication. After that change, new data uploaded to the source bucket would replicate to the new destination bucket rather than to the original destination bucket. However, after such a change, replicas already in the original destination bucket would remain there — they would not migrate to the new destination bucket.

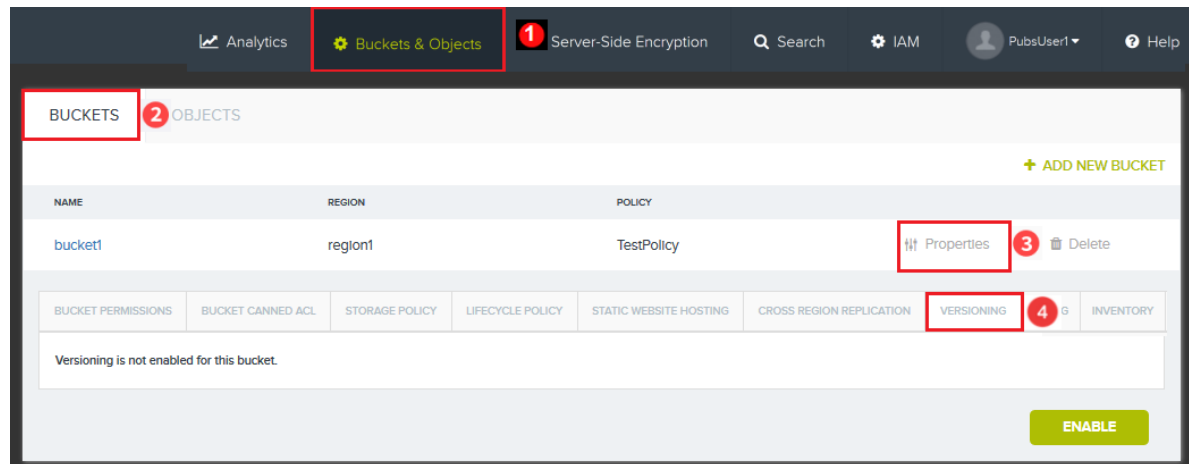
Disabling a Source Bucket's Replication Configuration

To disable cross-region replication for a source bucket, in the **Bucket Properties** interface for the bucket select the **Cross Region Replication** tab. Then de-select the "Enable Cross Region Replication" checkbox and click **Save**.

The system will no longer replicate newly uploaded objects from the source bucket to the destination bucket. However, any replicated objects already in the destination bucket -- from during the time period that you had cross-region replication enabled on the source bucket -- will remain there unless you delete them.

6.4.2.6. Enable/Disable Versioning for a Bucket

Path: **Buckets & Objects** → **Buckets** → **Properties** → **Versioning**



In the **Bucket Properties** interface you can use the **Versioning** tab to enable object versioning for a bucket, or to suspend versioning if you enabled it previously. By default versioning is not enabled.

When versioning is enabled on a bucket, each version of objects in the bucket is retained -- not just the current version but prior versions as well. For example, if you upload a newly created document called *GreatIdeas.docx*; and then later you revise the document and upload it again; and then later you revise it again and upload it again -- the system will store all three versions of the document (the initial version, the next revised version, and the current version).

Versioning protects you against losing an object due to accidentally overwriting it -- since the previous version is retained as well as the (accidental) new version. Also, versioning allows you access to all the past versions of an object throughout its history. For information about how this access appears in the CMC interface, see ["Download an Object" \(page 359\)](#).

By contrast, when versioning is not enabled, only the current (most recently uploaded) version of the object is retained in the system.

IMPORTANT ! If you enable versioning on your bucket, each version of an object will count toward your storage space consumption for purposes of usage reporting and usage-based billing.

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.

To enable versioning on a bucket, with the bucket properties **Versioning** tab selected, click the **Enable** button.

For information about downloading or deleting versions of an object see ["Download an Object" \(page 359\)](#) and ["Delete an Object" \(page 363\)](#).

Lifecycle Management for Versioned Buckets

HyperStore supports configuring lifecycle rules regarding current and non-current versions of objects in buckets for which versioning is enabled (all versions of an object other than the newest version are considered "non-current"). For example you could have non-current object versions auto-tiered to a different storage system -- or automatically deleted -- a certain number of days after they become non-current. For more information see ["Configure a Bucket Lifecycle Policy" \(page 315\)](#).

Suspending Versioning on a Bucket

If versioning is currently enabled on a bucket, you have the option of suspending it by selecting the **Versioning** tab in the CMC's bucket properties interface and then clicking **Suspend**. The behavior in a bucket for which versioning had been enabled and is then subsequently suspended is as follows:

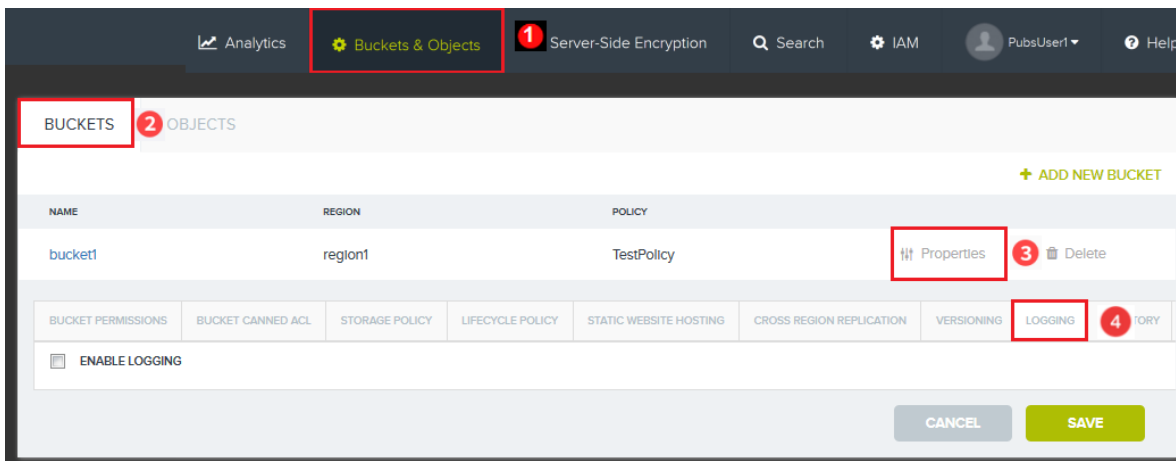
- For **new objects uploaded for the first time after versioning is suspended**, no versioning is applied. For example, if after suspending versioning you upload a new document *TerribleIdeas.docx*, and then later you revise that doc and upload the revised version, only the most current version will be retained.
- For **existing objects that had been uploaded during the period when versioning was enabled**:
 - All object versions that were already in the system are retained. In other words, when you suspend versioning this does not purge any existing object versions. (If you want to purge old object versions you can [delete them](#).)
 - Going forward, if you upload revised versions of objects that had been first uploaded during the period when versioning was enabled, what will be retained is:
 - All object versions from when versioning was enabled.
 - The most current of the object versions uploaded after versioning was suspended.

For example, if while versioning was enabled you upload versions 1, 2, and 3 of *GreatIdeas.docx*, and then after versioning is suspended you upload versions 4, 5, and 6 of the document, the system will retain versions 1, 2, 3, and 6.

Note For a bucket that has [object lock](#) enabled, versioning is automatically enabled and **cannot be suspended**.

6.4.2.7. Enable/Disable Bucket Logging

Path: **Buckets & Objects** → **Buckets** → **Properties** → **Logging**



In the **Bucket Properties** interface you can use the **Logging** tab to enable access logging for a bucket, or to disable access logging if you enabled it previously. By default access logging is not enabled.

When logging is enabled on a source bucket, every 10 minutes the HyperStore system generates an access log file for the bucket (if there has been access to the bucket during the past 10 minutes) and stores the log file in a specified destination bucket. The destination bucket can be the source bucket itself -- for instance you can have the system generate logs recording activity for *bucket1*, and store the log files in *bucket1* -- or the destination bucket can be a different bucket than the source bucket, with the restriction that the destination bucket must be in the same service region as the source bucket. Also, you must be the owner of the destination bucket.

For an overview of the bucket access log file content see ["Bucket Logging Content"](#) (page 336).

Before you enable logging for a source bucket, you must establish the necessary access rights on the destination bucket so that the system can write log files into it. You can do this through the CMC's **Bucket** page, by using the Properties interface for the destination bucket. You have two options for establishing the needed access rights on the destination bucket:

- Using the **Bucket Canned ACL** tab, set the canned ACL "Log Delivery Write" on the destination bucket.
- OR
- Using the **Bucket Permissions** tab, give the "Log Delivery" grantee permissions for "Writable" and "ACP Readable" on the destination bucket.

To enable logging for a source bucket, use the Properties interface for the source bucket, and go to the **Logging** tab. Then:

1. Select the "Enable Logging" checkbox. When you do this, entry fields will display.
2. Enter the name of the "Destination Bucket". As discussed above, this can be the source bucket itself or some other bucket that you own (in the same service region as the source bucket), and it must have the needed permissions already set on it.
3. In the "Target Prefix" field, optionally enter a text string that will be used as a prefix for all the bucket log file names for this source bucket. Absent a prefix, the log files will simply be named by a timestamp indicating the time of the log file generation. Using a prefix is recommended as it will make it easier for you to view and manage the log files (including possibly configuring lifecycle management for the log files). For example, if you enter "LogBucket1_" as the prefix, then all log files will be named as "LogBucket1_<timestamp>".

Note Using a comma in the Target Prefix is prohibited.

Note If you wish you can enable bucket logging on multiple source buckets (working through the Properties interface for each source bucket one at a time), and have all the logs written to the same destination bucket. In this case you could use prefix values to distinguish the logs from the different source buckets. For example, when enabling logging on *bucket1*, use "LogBucket1_" as the prefix, and when enabling logging on *bucket2*, use "LogBucket2_" as the prefix.

4. Click **Save**.

Once you have set up bucket logging, you may wish to use the [Bucket Lifecycle](#) feature to configure how the logs are handled as they age. For example you can configure it so that the log files are automatically deleted once they reach a certain age. To set up a bucket lifecycle rule for the logs, you must have used a Target Prefix

when you set up bucket logging (Step 3 above). You will need that prefix when you set up the bucket lifecycle rule.

Bucket Logging Content

The bucket access log entries generated by the HyperStore system have most but not all of the log entry fields that Amazon S3's bucket access logging feature does. Specifically, the HyperStore system's bucket logs contain the following fields:

- Bucket Owner (canonical user ID)
- Bucket
- Time
- Remote IP
- Requester
- Request ID
- Operation
- Key
- Request URI
- HTTP status
- Error Code
- Bytes Sent
- Object Size
- Total Time
- Turn Around Time (this will be the same as Total Time)
- Referrer
- User Agent
- Version Id

For details about the individual log fields see the Amazon documentation topic [Amazon S3 server access log format](#).

Editing or Disabling Bucket Logging

Once access logging is enabled for a source bucket, you can subsequently return to the bucket properties Logging tab for that bucket if you want to:

- Edit the bucket logging configuration (by changing the destination bucket or the log file name prefix)
- Disable logging for the bucket (by deselecting the "Enable Logging" checkbox)

Don't forget to click **Save**.

Note If you make any changes to the bucket logging configuration, this will affect how bucket logging is performed going forward. It will not retroactively impact log files that were previously generated.

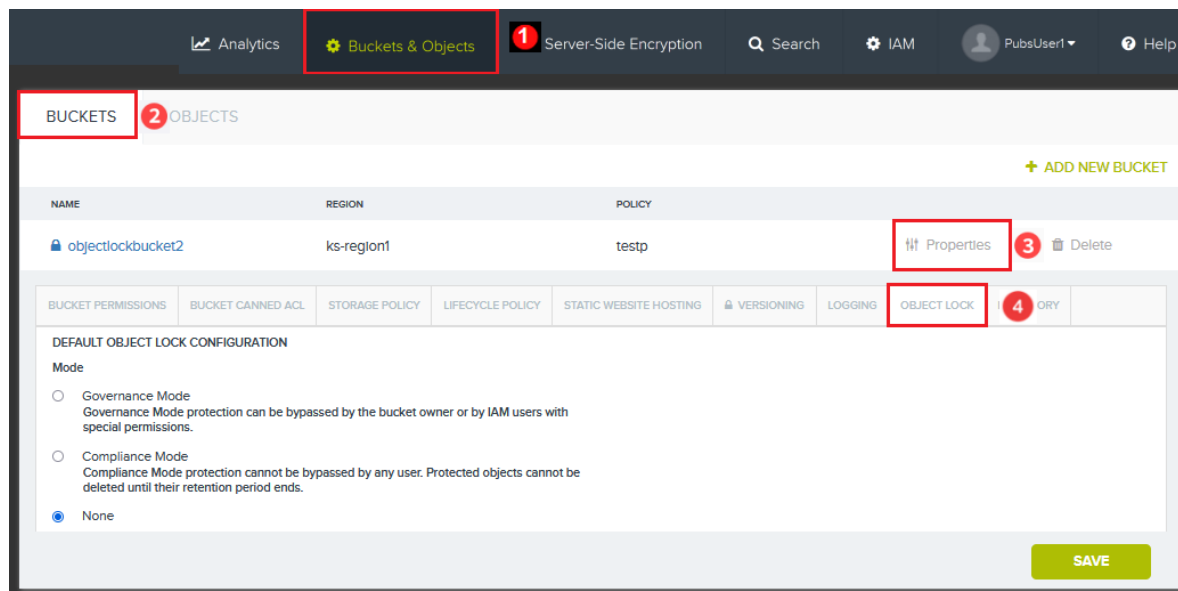
Masking Bucket Log Entry Fields for Requests From Internal Networks

If you wish you can configure the system so that for S3 requests originating from within your internal network(s) -- such as requests submitted through the CMC -- the Remote IP and User-Agent fields in bucket log entries

are masked. For more information see the configuration setting ["s3.bucketLogging.internalNetworks"](#) (page 815).

6.4.2.8. Configure Object Lock Properties for a Bucket

Path: **Buckets & Objects** → **Buckets** → **Properties** → **Object Lock**



For a [bucket that was created with Object Lock enabled](#), an **Object Lock** tab will appear in the bucket's **Properties** interface. Here you can set a default Object Lock configuration on the bucket that will by default be applied to all objects that are subsequently added to the bucket. The bucket's default Object Lock configuration:

- Does not apply to any objects that are already in the bucket at the time that you set the default Object Lock configuration. It applies only to objects uploaded **after** you set the default configuration.
- Can be overridden on a per-object basis, within limitations (as described in ["Set Object Lock Attributes on an Object"](#) (page 356)).

If you do not set a default Object Lock configuration on the bucket, then objects uploaded to the bucket will not be locked unless they have Object Lock attributes set on them on a per-object basis (as described in ["Set Object Lock Attributes on an Object"](#) (page 356)).

To set a default Object Lock configuration on the bucket:

1. Choose a lock mode -- either Governance Mode or Compliance Mode.
 - **Governance Mode:** With this mode, you as the bucket owner can remove the lock from a locked object before the end of its retention period, if you wish. Once you've removed the lock from an object, you can delete the object. This mode also allows you to shorten the retention period for individual locked objects if you wish. These per-object controls are described in ["Set Object Lock Attributes on an Object"](#) (page 356).
 - **Compliance Mode:** With this mode, you cannot remove the lock from a locked object before the end of its retention period. Nobody, including you as the bucket owner, can delete locked

objects through the S3 interface until the end of their retention period. Also with this mode, nobody can shorten the retention period for a locked object.

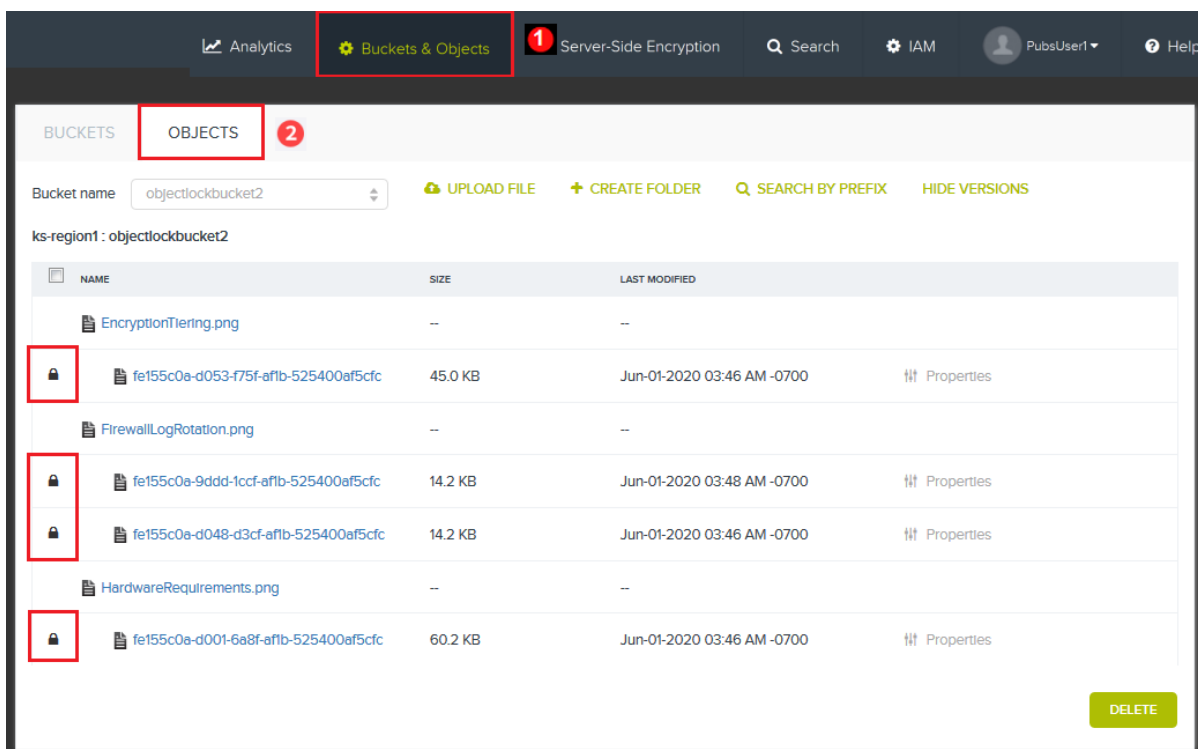
Note For a summary of protections that the S3 API provides to locked objects see "[Protections for Locked Objects](#)" (page 229).

- Specify a retention period in number of days or number of years.

The retention period will be applied to each object version starting upon its upload to the bucket. For example, with a retention period of one year, each object version will be locked for one year starting from the date that the object version is uploaded to the bucket.

- Click **Save**.

Once a default Object Lock configuration has been set for the bucket, any objects subsequently uploaded into the bucket will be locked in accordance with the terms of the default Object Lock configuration. In the bucket's object list, locked objects have a padlock icon beside the object name. Note that versioning is automatically enabled in a bucket that uses Object Lock and that each version of an object is locked.



Updating a Bucket's Default Object Lock Configuration

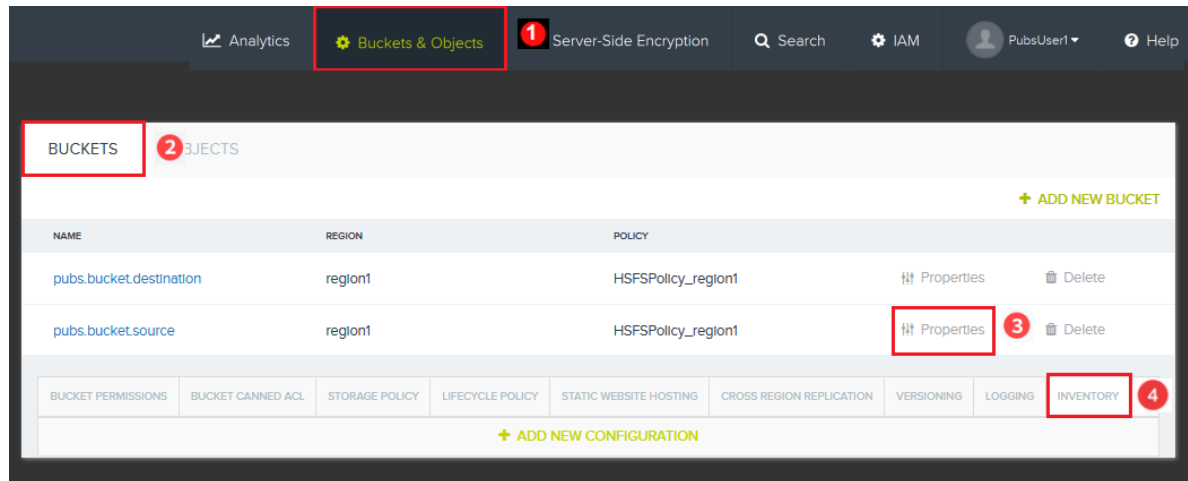
Once you've set a default Object Lock configuration on a bucket, you still have the ability to later change that default configuration if you wish (by once again using the **Object Lock** tab in the bucket's **Properties** interface). However, such a change will apply only to objects that are uploaded **after** you've made the change. Any objects uploaded while your original default Object Lock configuration was in place will continue to be controlled by the terms of that original Object Lock configuration.

Note If you've created a [file share](#) that uses the bucket, the bucket's default Object Lock configuration is loaded into the file share only when the file share is opened. Therefore if you want to change the

bucket's default Object Lock configuration you should first close the file share; then change the bucket's default Object Lock configuration; then re-open the file share.

6.4.2.9. Configure Bucket Inventory Reporting

Path: **Buckets & Objects** → **Buckets** → **Properties** → **Inventory**



In the **Bucket Properties** interface you can use the **Inventory** tab to configure recurring inventory reports for a bucket. You can configure inventory reporting attributes such as whether to list all objects in the source bucket or only objects whose names start with a specified prefix; whether to have the reports generated daily or weekly; and which types of metadata to report for each object. If you wish you can configure multiple recurring reports with different attributes -- for example, a daily report on objects whose names start with "Images/" and a weekly report on objects whose names start with "Videos/". You can configure as many as 1000 recurring reports per source bucket.

By default inventory reports are not generated. To have inventory reports generated for a source bucket you must configure one or more reports using the **Inventory** tab.

To add a recurring report configuration for a source bucket:

1. In the **Inventory** tab click **Add New Configuration**. This displays the **Add New Bucket Inventory Configuration** screen.

2. Configure the desired report attributes.

Inventory Configuration Name (mandatory)

Name of this recurring inventory report configuration. Maximum 64 characters, and only letters, numbers, underscores, dashes, and periods are allowed in the name.

Prefix (optional)

Optionally enter the "Prefix" for the subset of objects that you want to be listed in the inventory report. To have the report list **all objects in the bucket**, leave the "Prefix" field empty.

If entering an object prefix, the path should start from the root level of your bucket, and you do not need to include a leading forward slash ("/"). You should however include a trailing forward slash to demarcate the "folder" name from its "contents" (which will be subject to the report that you're configuring). For example:

Images/archived/2024/

This object prefix would apply to all objects with names that start with *Images/archived/2024/*, such as *Images/archived/2024/January/bird.png* and *Images/archived/2024/February/cat.jpg*

Object Versions: Current or All

Choose whether to have the report list only current object versions or all object versions. This distinction

matters only if you have [Versioning](#) enabled on the bucket.

Destination Bucket (mandatory) and Destination Prefix (optional)

For "Destination Bucket", enter the name of the bucket in which you want the system to store this recurring report. This can be the same as the source bucket -- for example, reports about a bucket named "MyBucket" can be stored into "MyBucket". Or, this can be any other bucket **that you own** in the same HyperStore system. (Because the destination must be a bucket that you own, you will see that "This Account" is selected above the destination bucket field and cannot be changed. You cannot use a destination bucket owned by a different user account.)

Optionally you can specify a Prefix for the system to use when it writes this report into the destination bucket -- for example "InventoryReports/". If you don't specify a Prefix the system will write this report into the root of the bucket.

Note Using a Prefix for the reports in the destination bucket can be helpful if you want to [create a lifecycle](#) on the destination bucket that deletes old reports. You can create a lifecycle rule that target objects that start with that Prefix.

Frequency: Daily or Weekly

Choose how often you want the system to generate this inventory report -- daily or weekly.

The system will continue generating this report at your chosen frequency unless you disable or delete this report configuration.

Output Format: CSV

The only supported report format is comma-separated values (CSV).

Status: Disable or Enable

You have the option of disabling a report if you want to pause the report generation without deleting the report configuration entirely. For example if you've created a daily report configuration, and at some future point you decide you want to pause the report for a week, you can edit the report configuration to set its Status to Disable -- and then after the week passes edit the report configuration again to set its Status back to Enable.

Additional Fields (optional)

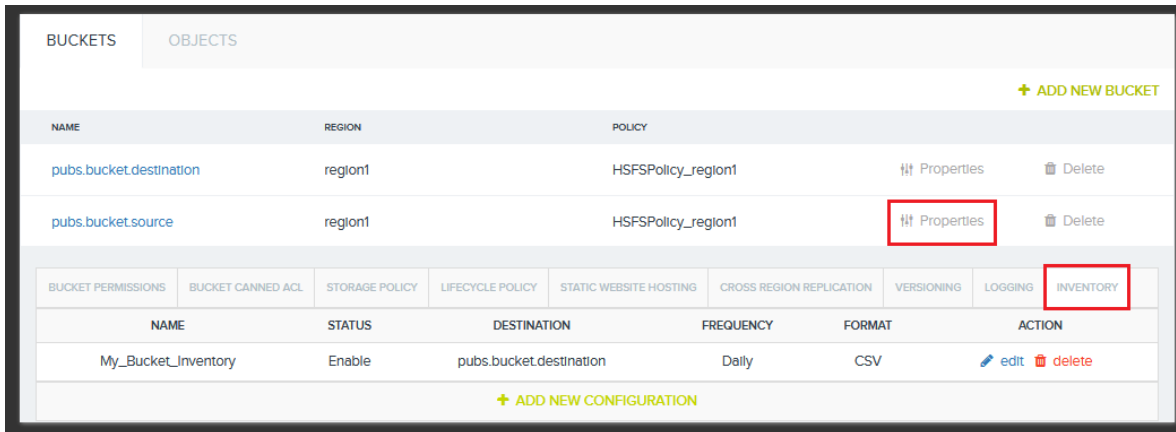
Use the checkboxes to select the object attributes that you want included in the per-object report entries (in addition to the bucket name and object name, which are always included.) Each entry in the report will be formatted as:

bucketname,objectname,attribute1,attribute2,etc.

Note The "Bucket key status" and "Intelligent-Tiering Access tier" attributes are not applicable to HyperStore and will always be empty if you include them in your reports.

3. Click **Save**.

You are then returned to the bucket list screen. If for the source bucket you select **Properties** and the **Inventory** tab you can see the completed inventory report configuration listed by name, along with some basic information about the report such as the destination bucket and whether the report is currently enabled.



Also from this view you can **Edit** the report configuration. If at some point in the future you no longer want the report to be generated, you can come to this view to **Delete** the report configuration.

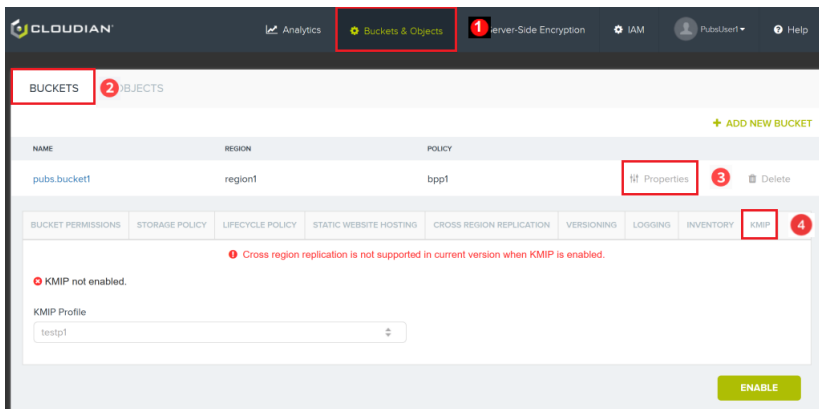
Note Editing or deleting the report configuration affects the system's behavior only from that time forward. It does not affect reports that have already been generated and saved into the destination bucket.

Report Output

- By system default, reports are generated at midnight. This is controlled by a [configurable cron job](#).
- In the destination bucket, reports will be written as .csv files. If you specified the use of a destination prefix in your report configuration, the report file names will start with that prefix.
- If there are a very large number of objects listed in a report, the report content will be partitioned into multiple .csv files.
- Each entry line in the report will be formatted as:
bucketname,objectname,attribute1,attribute2,etc.
- For each report there will also be a corresponding manifest file (with metadata about the report) and a checksum file.

6.4.2.10. Enable KMIP Encryption for a Bucket

Path: **Buckets & Objects** → **Buckets** → **Properties** → **KMIP**



In the **Bucket Properties** interface you can use the **KMIP** tab to check whether KMIP encryption is enabled for the bucket, and to enable KMIP encryption for the bucket if it is not already enabled. KMIP is a form of server-side encryption (encryption of "data at rest") in which the master encryption keys are generated and managed by an external Key Management System (KMS) that supports the Key Management Interoperability Protocol (KMIP).

KMIP-based server-side encryption will already be enabled for the bucket if:

- The bucket uses a [storage policy](#) for which the system administrator has pre-configured KMIP encryption as an attribute of the storage policy. In this case KMIP encryption is automatically enabled for the bucket when you create the bucket.

OR

- You have previously enabled KMIP encryption for the bucket, using the bucket properties **KMIP** tab.

If KMIP-based server-side encryption is not already enabled for the bucket, you can enable it by selecting a "KMIP Profile" from the drop-down list and then clicking **Enable**. A KMIP profile identifies a KMS and how the object storage system should connect to it. In the "KMIP Profile" drop-down list will be:

- KMIP profiles created by the system administrator, if any
- KMIP profiles created by your group administrator, if any
- KMIP profiles that you have created in the Console's **"KMIP Profiles"** (page 545) interface, if any

At least one KMIP profile must exist before you can enable KMIP for a bucket.

Note that:

- Once KMIP encryption is enabled for a bucket, **you cannot change the KMIP profile associated with the bucket**.
- KMIP encryption can be disabled for a bucket for which it is currently enabled, but only by using a third party S3 application to submit an S3 *DeleteBucketEncryption* request to the object storage system's S3 Service. The Console does not currently support disabling KMIP encryption for a bucket.
- A bucket with KMIP encryption enabled is not allowed to be a source bucket for [cross-region replication](#).

6.4.3. Delete a Bucket

To delete an **empty** bucket, go to the [Buckets](#) list view and click the **Delete** button on the far right of the bucket's display row.

If you want to delete a bucket that has objects in it, your options depend on how system administrators have configured the Console. Go to the [Buckets](#) list view and click the **Delete** button for the bucket:

- If the Console is configured to support purging all of a bucket's contents in one operation, a confirmation dialog will display to ask you whether you want to purge the bucket's contents. If you type PURGE (in all caps) in the text box, the system will initiate a purge of the bucket contents (as an asynchronous background operation) and you can delete the bucket by once again clicking the **Delete** button for the bucket.
- If your Console is not configured to support purging all of a bucket's contents in one operation, a dialog will inform you that there are objects in the bucket and that you must delete them before you can delete the bucket. You can then go to the [Objects](#) view and delete all the objects in the bucket (or alternatively you can use a third party S3 application to delete all the objects in the bucket). Once the bucket is

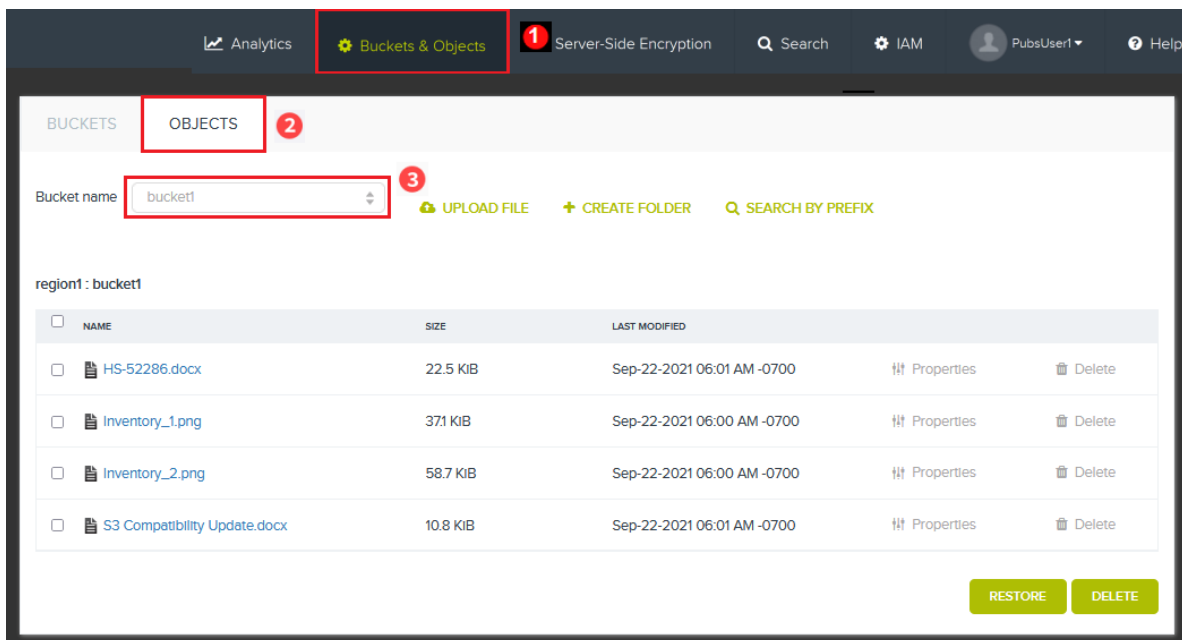
empty you can return to the [Buckets](#) list view and delete the bucket by clicking the **Delete** button for the bucket.

Note The Console does not ever support purging the contents of a bucket for which Object Lock is enabled.

Note If your user account has [file services enabled](#) and you have created one or more file shares, you cannot delete a bucket that is serving as the storage repository for a file share. You must [delete the file share](#) first and then you can delete the bucket.

6.5. Objects

Path: **Buckets & Objects** → **Objects** → <bucketname>



In the **Objects** interface you can select from the drop-down list of buckets that you own, and then work with objects in that bucket. Alternatively, clicking on a bucket name in the [Buckets](#) interface will bring you to the **Objects** interface for that bucket.

Note The **Objects** tab does not display until you've created a bucket.

If you have set the configuration setting **"s3.qos.bucketLevel"** (page 828) to "true" -- by default it is "false" -- then toward the top of the **Objects** interface you can see a count of the number of objects and number of bytes in the bucket. Note that:

- The bucket itself counts as one object (so, with an empty bucket the counter will indicate that there's one object). Also the bucket itself counts as several bytes of data, varying by the length of the bucket name.

- In some atypical circumstances the count of bytes in a bucket may temporarily become unavailable. This will be corrected by the usage repair cron jobs that run twice a day.

Supported tasks:

- ["Create or Delete a "Folder""](#) (page 345)
- ["Upload an Object"](#) (page 346)
- ["Set Object Properties"](#) (page 348)
- ["List or Search for Objects"](#) (page 357)
- ["Download an Object"](#) (page 359)
- ["Retrieve a Tiered Object"](#) (page 360)
- ["Delete an Object"](#) (page 363)

Note In an object-based storage system, each data entity that you store in a bucket is known as an "object". Within a bucket each object has a unique name. In most respects you can think of an object as being a file. For some distinctions between object storage and conventional file systems, see ["Note about 'folders' in an object storage system"](#) (page 345).

6.5.1. Create or Delete a "Folder"

In the [Objects](#) interface, first select the bucket that you want to work with and then click **Create Folder**. In the pop-up dialog that displays, enter a folder name and then click **OK**. The new folder will then display in the object list view.

For folder naming restrictions, see ["Object Naming Restrictions"](#) (page 348). These restrictions apply to unusual naming schemes and are of no concern if you are using a simple alphanumeric folder name.

Note The CMC does not support renaming existing folders.

To delete a folder:

To delete a folder click **Delete** at the right side of the folder's display row. You will be asked to confirm that you want to delete the folder.

IMPORTANT ! Deleting a folder will also delete all files and sub-folders within the folder.

6.5.1.1. Note about 'folders' in an object storage system

In an object-based storage system, each data entity stored in a bucket is known as an "object". Within a bucket, each object has its own unique name. In the storage layer -- the storage system back-end -- an object storage system does not have a hierarchical directory structure like a conventional file system. Instead it is a "flat" structure with all objects existing in the root of the bucket. However, objects can be named in such a way as to imply a hierarchy and allow a user interface such as the CMC to present a hierarchical view of the bucket contents. For example, in a bucket there could be four objects named:

- *Documents/resume.docx*
- *Documents/schedule.docx*

- *Images/birthday-party.png*
- *Images/sunset.png*

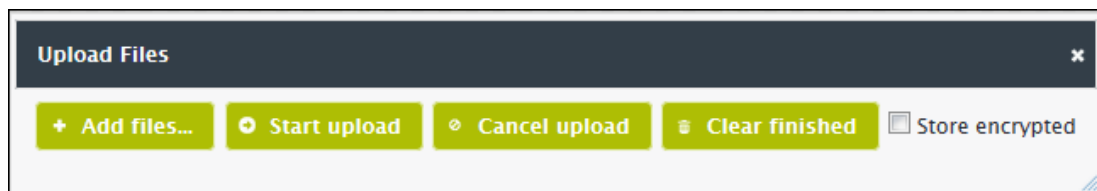
The CMC makes it easy to create such hierarchies by enabling you to create "folders". From the example above, through the CMC interface you could create a folder named "Documents", and then drill down into that folder and upload files named "resume.docx" and "schedule.docx". Behind the scenes, in the object storage layer, the objects are named as "Documents/resume.docx" and "Documents/schedule.docx". In the CMC interface, you can then retrieve a list of the contents of the "Documents" folder: the display will list the "resume.docx" and "schedule.docx" files.

Thus while the back-end is implemented differently, the CMC user experience is very similar to interacting with a regular file system.

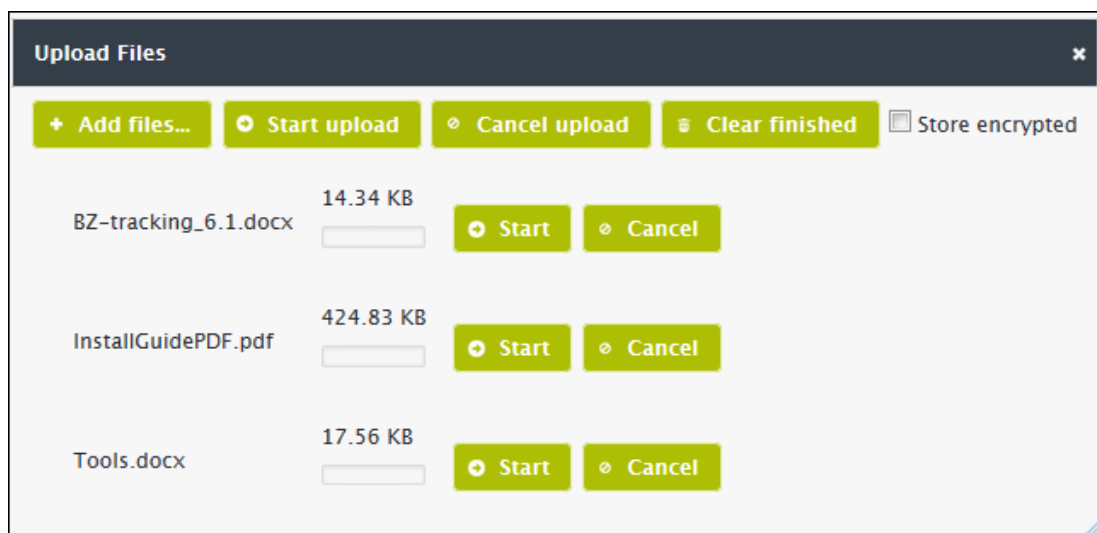
6.5.2. Upload an Object

Note If you haven't yet done so, you must [create a bucket](#) before you can upload any objects.

1. In the [Objects](#) interface, first select the bucket that you want to work with and then click **Upload File**. The **Upload Files** interface displays.



2. Click **Add Files**, then use the browse window that opens to browse to the file(s) on your computer. You can select multiple files from within the same directory on your computer by using <Ctrl>-click or <Shift>-click. After you've selected the files and clicked **Open** in the browse window, the list of selected files displays in the **Upload Files** interface.



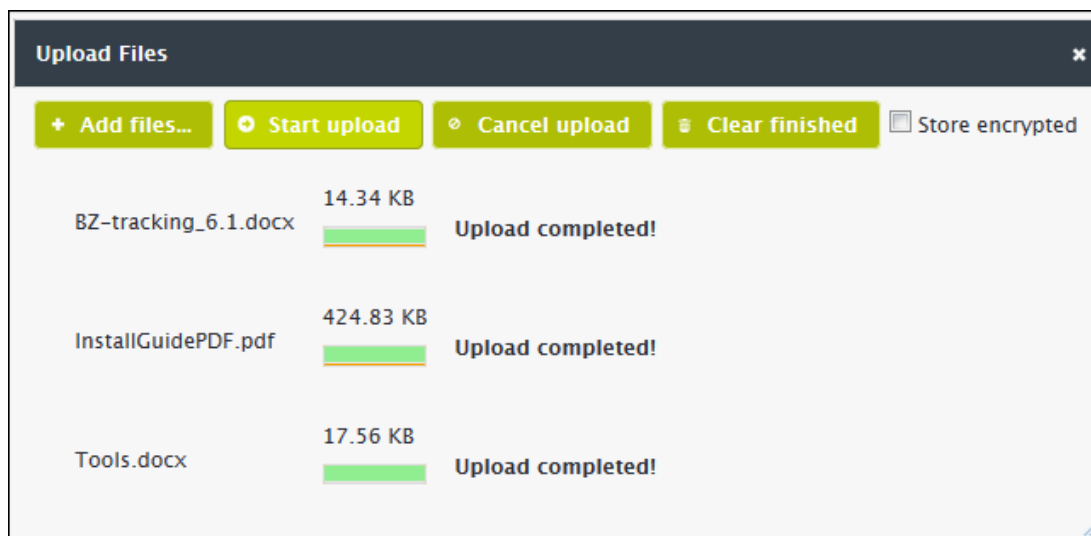
Note By default the maximum sized object that you can upload through the CMC interface is 5GB.

The maximum allowed S3 object name length — that is, the maximum length of the folder path plus file name, combined — is 1024 characters. For additional naming restrictions see ["Object Naming Restrictions"](#) (page 348). These additional restrictions pertain to unusual characters or character combinations and so are of no concern if you are using simple alphanumeric file names.

3. If you want the HyperStore system to store the file(s) in encrypted form for enhanced security, click the "Store encrypted" checkbox. If an encrypted file is subsequently downloaded, the system will automatically decrypt the file before transmitting it to the requesting client application.

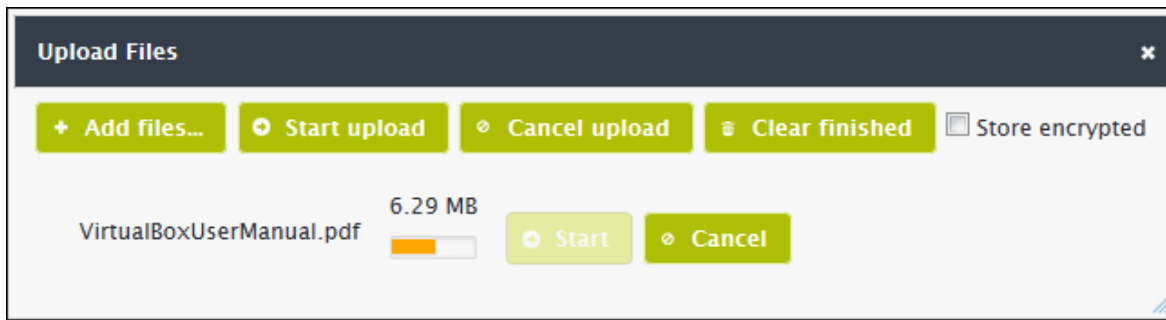
Note If you are uploading objects into a bucket that is using a storage policy that is configured for server-side encryption by default, or a bucket that is configured for server-side encryption by default, then the objects will be automatically encrypted by the system regardless of whether or not you select the "Store encrypted" checkbox.

4. To upload all the files click **Start Upload** at the top of the **Upload Files** interface. (Or if you want to upload files one-by-one, click **Start** to the right of individual file names). When all files have uploaded, a success message appears.



If you want to continue uploading more files, click **Clear Finished** to clear the **Upload Files** interface, then click **Add Files** to add more files to upload. If you're done uploading files, click the "X" in the upper right of the **Upload Files** interface to close it.

If you upload a large object you can monitor its upload progress by watching the progress bar beside the file name.



Objects that you have successfully uploaded appear in the **Objects** interface. If you chose to encrypt a file, a padlock icon displays to the left of the file name.

Note For a large object over a certain size threshold (16MB by default), the CMC will upload the object to the S3 Service in multiple separate parts, using multiple HTTP transactions. This is known as multipart upload (MPU). In such cases, a **Multipart Upload in Progress** section displays at the bottom of the **Objects** interface. This display indicates the time at which the multipart upload was initiated and the object name (the "Key"). If the object appears to be taking too long to upload, this may indicate that a problem has occurred while uploading the multiple parts. In this case you can use the **Abort** button to abort the incomplete MPU operation. By aborting the MPU operation, you remove from storage any object parts that had been uploaded before problems occurred with the operation. You can subsequently try again to upload the object, starting over from Step 1 of the procedure above.

6.5.2.1. Object Naming Restrictions

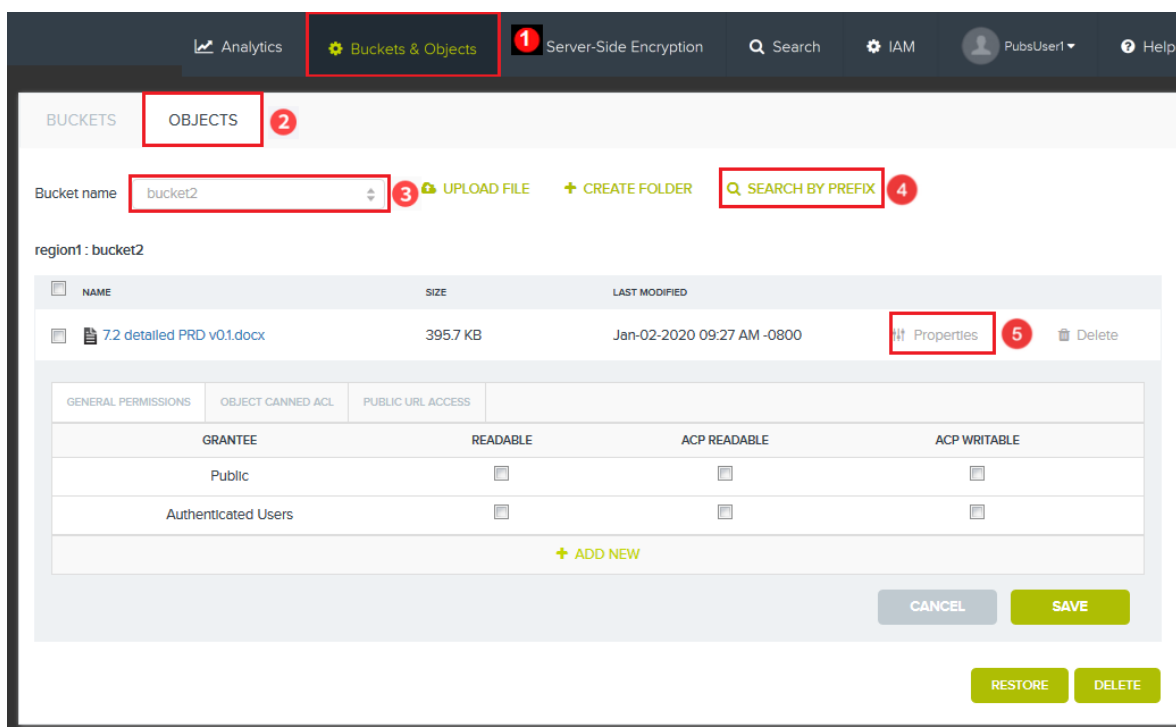
The following control characters cannot be used anywhere in an object name and will result in a 400 Bad Request response: 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A ("\\n"), 0x0B, 0x0C, 0x0D ("\\r"), 0x0E, 0x0F, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1A, 0x1B,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

6.5.3. Set Object Properties

Path: **Buckets & Objects** → **Objects** → <bucketname> → <find object> → **Properties**



In the [Objects](#) interface, first select the bucket that you want to work with. Then in the object list that displays for that bucket, [find the object](#) for which you want to set properties. Then click **Properties** in the object's display row. This opens the properties interface for that object.

Supported tasks:

- ["Set Custom S3 Permissions on an Object"](#) (page 349)
- ["Set "Canned" S3 Permissions on an Object"](#) (page 352)
- ["Set Public URL Permissions on an Object"](#) (page 354)
- ["Set Object Lock Attributes on an Object"](#) (page 356)

Note For custom and canned S3 permissions, the CMC supports only setting permissions on individual objects, one-by-one. If you want to make **all** objects within a bucket -- or a group of objects such as those with a common prefix (directory path) -- readable to the public or to specified individuals or groups, the best way to accomplish this is through creating a bucket policy. The S3 API supports this (see [PUT Bucket policy](#)), and some S3 client applications may support this, but the CMC currently does not.

6.5.3.1. Set Custom S3 Permissions on an Object

Path: **Buckets & Objects** → **Objects** → <bucketname> → <find object> → **Properties** → **General Per-**
missions

region1 : bucket2

NAME	SIZE	LAST MODIFIED		
7.2 detailed PRD v0.1.docx	395.7 KB	Jan-02-2020 09:27 AM -0800	Properties 1	Delete

GENERAL PERMISSIONS **2**

OBJECT CANNED ACL

PUBLIC URL ACCESS

GRANTEE	READABLE	ACP READABLE	ACP WRITABLE
Public	☐	☐	☐
Authenticated Users	☐	☐	☐

+ ADD NEW

CANCEL SAVE

[As is the case with Amazon S3](#), the HyperStore S3 Service supports three different ways of managing permissions for S3 buckets and objects: IAM policies, bucket policies, and S3 ACLs.

- The CMC supports creating IAM users, groups, and policies as described in ["IAM"](#) (page 435).
- The CMC does not support creating bucket policies, but HyperStore's S3 API does support this, so you can use a third party S3 client application to create a bucket policy for a bucket if you wish.
- The CMC supports configuring ACLs for an object. You can do so in the ["Set Object Properties"](#) (page 348) interface, by using either:
 - The [Object Canned ACL tab](#), to select from a small set of pre-defined ACL packages for an object.
 - The **General Permissions** tab, for more granular, customizable configuring of ACLs. This is described in the instructions below.

As the owner of an object you always have full permissions to the object, which consist of:

- Permission to read the object (view or download it)
- Permission to view and change the object's ACL settings

To configure custom ACLs that grant object permissions to other users, follow the instructions below.

Note You can use the CMC to configure ACL permissions on your object, but your grantees -- users to whom you grant ACL-based permissions -- cannot use the CMC to exercise those permissions. The CMC only allows a bucket's owner to access the bucket and its contents. The CMC does not allow users to access a bucket that they do not own, even if the bucket owner has granted them permissions on that bucket and its contents. However, most other S3 client applications allow users to access buckets and objects to which they have been granted ACL permissions by the bucket owner. So, your grantees can use S3 applications other than the CMC to exercise ACL permissions that you grant them.

In the Grantee column are the different persons or groups of persons to whom you can grant permissions for the object.

Grantee	Description
Public	The general public, including persons who are not registered users of the Cloudian HyperStore service. This is essentially any person with an S3 client application and knowledge of the HyperStore S3 service endpoint (URL) and the bucket name.

Grantee	Description
Authenticated Users	All registered users of the Cloudian HyperStore service .
Specific group or user	If you want to grant permissions to a specific HyperStore user group or a specific user, click Add New. A new grantee row appears, with an empty grantee name box. In the name box do either of the following: • If the new grantee is a group, enter the group name followed by a vertical bar (with no space in between). For example, <i>Engineering</i>. This is to grant permissions to all users who belong to the group. • If the new grantee is a user, enter the group name followed by a vertical bar and then the user name (with no spaces in between). For example, <i>Engineering Balaji</i>. Note that this must be a HyperStore account root user -- it cannot be an IAM user. Note Group and user name matching are case-sensitive, so be sure to use the correct case when specifying the grantee. To configure permissions for multiple group or user grantees, click Add New for each one.

For each grantee you can select one or more of the following object permissions:

Permission	Description
Readable	With this object permission, the grantee can read (open or download) the object.
ACP Readable	With this object permission, the grantee can view the current permission settings for the object.
ACP Writable	With this object permission, the grantee can change the permission settings for the object.

Note Write permissions for uploading, overwriting, and deleting objects are set at the bucket properties level, not the object properties level. See "[Set Bucket Properties](#)" (page 309).

After making the permissions selections, click **Save**.

Setting Custom Permissions on an Object Version

If you have [versioning](#) enabled on your bucket, and you want to set permissions for a version of an object, in the **Objects** interface click **Show Versions**. Under each object name, the interface will then list all stored versions of that object, ordered from the current (most recently uploaded) version to the oldest version. Each version is identified by a system-generated alphanumeric identifier, and for each version the upload timestamp is shown (the date and time that particular version of the object was uploaded to the system). To set custom permissions for an object version, in the CMC's **Objects** interface click **Properties** in the object version's display row. This opens the properties interface for that object version. Then follow the instructions above.

Note When you set permissions for a version of a versioned object, the permissions apply **only to that version of the object** -- not to any past or future versions of the object. For example, if you set permissions for the current version of a versioned object, and then you later upload a revised version of the object, the permissions that you set previously will not apply to the revised version of the object. If you want there to be permissions on the revised version of the object, you must explicitly set the permissions on that object version after you upload it. This behavior is compliant with Amazon S3.

Editing Custom Object Permissions

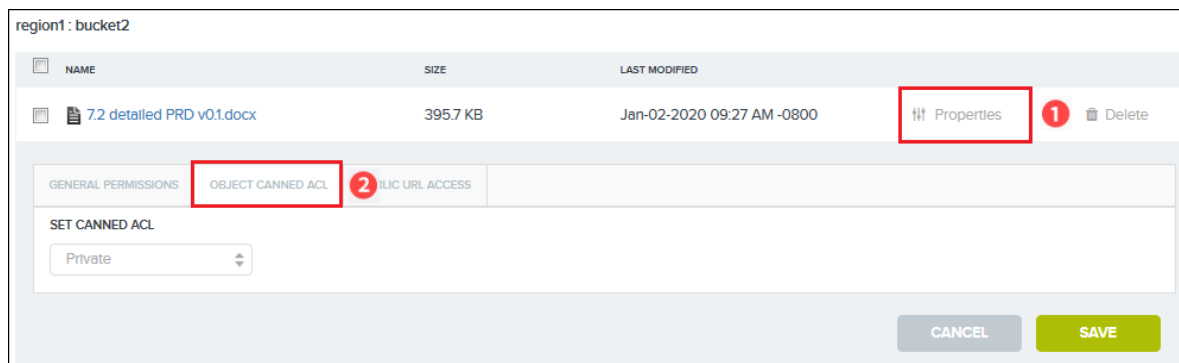
After you've configured custom S3 permissions for an object, you can subsequently modify the permissions configuration by returning to the **General Permissions** tab for the object, selecting or deselecting permissions, and then clicking **Save**.

If you had previously added a group or user grantee and you now want to delete that grantee, you can do so by clicking **Delete** on the right of that grantee row and then clicking **Save**. Be sure to click **Save** after clicking **Delete** -- otherwise the grantee will not actually be deleted (the grantee row will disappear when you click **Delete**, but if you do not click **Save** the grantee will still have its permissions and if you leave the **General Permissions** tab and then return to it, the grantee row will be there again).

Note If in your list of Grantees you see a row labeled **"Deleted User(s)"**, this means that one or more users to whom you previously granted permissions have been deleted from the system (they are no longer users of the storage service). You can delete this row by clicking **Delete** to the right of the row.

6.5.3.2. Set "Canned" S3 Permissions on an Object

Path: **Buckets & Objects** → **Objects** → <bucketname> → <find object> → **Properties** → **Object Canned ACL**



As is the case with Amazon S3, the HyperStore S3 Service supports three different ways of managing permissions for S3 buckets and objects: IAM policies, bucket policies, and S3 ACLs.

- The CMC supports creating IAM users, groups, and policies as described in ["IAM" \(page 435\)](#).
- The CMC does not support creating bucket policies, but HyperStore's S3 API does support this, so you can use a third party S3 client application to create a bucket policy for a bucket if you wish.
- The CMC supports configuring ACLs for an object. You can do so in the ["Set Object Properties" \(page 348\)](#) interface, by using either:

- The **Object Canned ACL** tab, to select from a small set of pre-defined ACL packages for an object. This is described in the instructions below.
- The [General Permissions tab](#), for more granular, customizable configuring of ACLs.

As the owner of an object you always have full permissions to the object, which consist of:

- Permission to read the object (view or download it)
- Permission to view and change the object's ACL settings

To set canned ACLs that grant object permissions to other users, follow the instructions below.

Note You can use the CMC to configure ACL permissions on your object, but your grantees -- users to whom you grant ACL-based permissions -- cannot use the CMC to exercise those permissions. The CMC only allows a bucket's owner to access the bucket and its contents. The CMC does not allow users to access a bucket that they do not own, even if the bucket owner has granted them permissions on that bucket and its contents. However, most other S3 client applications allow users to access buckets and objects to which they have been granted ACL permissions by the bucket owner. So, your grantees can use S3 applications other than the CMC to exercise ACL permissions that you grant them.

From the drop-down list, choose the canned ACL to assign to the object. You can only choose one.

Canned ACL	Description
Private	This object canned ACL grants no one any permissions on the object (so that only the object owner has permissions on the object). This is the default.
Public Read	This object canned ACL grants "the public" -- any S3 application user, regardless of whether or not the user is a registered user of the Cloudean HyperStore service -- permission to read (open or download) the object.
Authenticated Read	This bucket canned ACL grants all registered HyperStore users permission to read a list of files that are in the bucket.
Bucket Owner Read	This bucket canned ACL grants the bucket owner read permission. If you are the bucket owner and the object owner, you do not need to select this permission -- as object owner you already have full permissions on the object (including read permission).
Bucket Owner Full Control	This bucket canned ACL grants the bucket owner full permissions (permission to read the object; to read the object's permissions; and to change the object's permissions). If you are the bucket owner and the object owner, you do not need to select this permission -- as object owner you already have full permissions on the object.

Note Write permissions for uploading, overwriting, and deleting objects are set at the bucket properties level, not the object properties level. See ["Set Bucket Properties"](#) (page 309).

After selecting a canned ACL for the object, click **Save**.

Note An object can have only one canned ACL attached to it at a time. If you apply one canned ACL to an object, and then you later apply a different canned ACL, the first canned ACL will be removed.

Setting Canned Permissions on an Object Version

If you have [versioning](#) enabled on your bucket, and you want to set permissions for a version of an object, in the **Objects** interface click **Show Versions**. Under each object name, the interface will then list all stored versions of that object, ordered from the current (most recently uploaded) version to the oldest version. Each version is identified by a system-generated alphanumeric identifier, and for each version the upload timestamp is shown (the date and time that particular version of the object was uploaded to the system). To set canned permissions for an object version, in the CMC's **Objects** interface click **Properties** in the object version's display row. This opens the properties interface for that object version. Then follow the instructions above.

Note When you set permissions for a version of a versioned object, the permissions apply **only to that version of the object** -- not to any past or future versions of the object. For example, if you set permissions for the current version of a versioned object, and then you later upload a revised version of the object, the permissions that you set previously will not apply to the revised version of the object. If you want there to be permissions on the revised version of the object, you must explicitly set the permissions on that object version after you upload it. This behavior is compliant with Amazon S3.

6.5.3.3. Set Public URL Permissions on an Object

Path: **Buckets & Objects** → **Objects** → <bucketname> → <find object> → **Properties** → **Public URL Access**

Setting public URL permissions on an object enables members of the public to access the object by using a standard web browser (rather than an S3 client application). The system will generate for the object a web URL that you can provide to whoever you want to have access to the object.

Note that setting public URL permissions is not supported for the following types of objects:

- Objects in a bucket that resides in a non-default service region within a multi-region object storage system.
- Objects that have been server-side encrypted with a user-provided encryption key. (By contrast, public URL permissions are supported for files that have been encrypted with a system-generated encryption key, which is the more typical encryption method.)

To set public URL permissions on an object, in the **"Set Object Properties"** (page 348) interface for the object select the **Public URL Access** tab.

1. Click the "Enable Public URL Access" checkbox.
2. Configure the public URL access for the object:

- In the "Maximum Downloads" field, specify a maximum number of times that the object can be downloaded in total (the maximum total downloads that you want to allow, by all users combined). If you want to allow an unlimited number of downloads, set this to "-1" (negative one). Note that after enabling public URL access for the object, you can subsequently return to this interface and view the number of times the object has been downloaded to date (the "Current Downloads" field).
- In the "Expiration Date/Time" field, choose an expiration date/time at which the public URL for the object will expire. After this time the URL will no longer be valid and the object will not be accessible via the URL. If you click in the field, a calendaring tool displays to make it easy to choose the date and time.
- If you want HTTP access to the object to be SSL-secured, select the "Secure URL (HTTPS)" checkbox.

3. Click **Apply** and the system-generated public URL for the object will display in the interface.

NAME	SIZE	LAST MODIFIED		
6.2_AutoTiering_CustomEndpoint.png	31.3 KB	Apr-07-2017 12:35 PM -0700	Properties	Delete

GENERAL PERMISSIONS

OBJECT CANNED ACL

PUBLIC URL ACCESS

☒ **ENABLE PUBLIC URL ACCESS**

Maximum Downloads:
Current Downloads: 0

Expiration Date/Time:
☐ Secure URL (https)

URL: http://s3-region1.mycloudianhyperstore.com/bucket1/6.2_AutoTiering_CustomEndpoint.png?AWSAccessKeyId=5cb6eee4356624d883ef&Expires=1493535600&Signature=9wke%2FoCdGhCNJxsDMACIFVTsxf%3D&x-amz-pt=MDAxZTE1NDExNDkxNTk0NDk2Mzk2

MAIL TO CLOSE APPLY

The public URL for an object has the following format:

```
http[s]://<S3Endpoint>/<bucketName>/<objectName>?AWSAccessKeyId=<accessKeyOfObjectOwner>
&Expires=<expiryTimeUnixSecs>&Signature=<signatureString>&x-amz-pt=<uniqueID>
```

The CMC interface makes it easy for you to share this public URL with others. Once the public URL is displayed in the interface, click **Mail To** to open your default email client with the public URL pre-populated into the message body of a new email message. Just add recipients and optionally edit the subject line (which by default is "Public URL Access Link") and then send the email message.

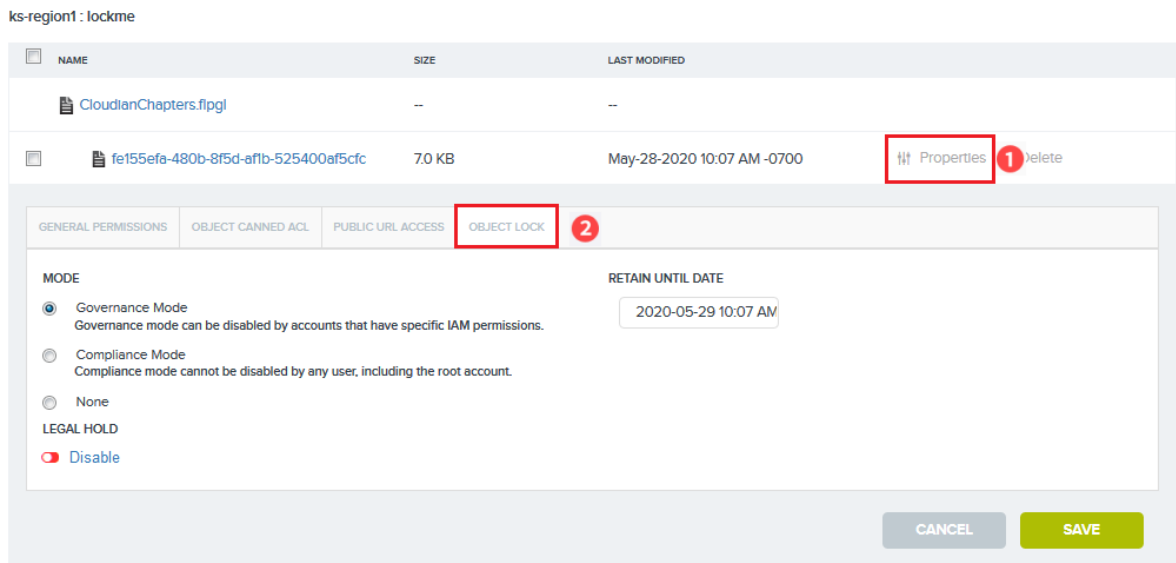
Note For the **Mail To** feature to work, your browser must be configured to be able to open your default web-based email client.

If you want, you can make changes to an object's public URL settings at a later point in time. For example, you can change the maximum allowed downloads or the expiration time. Note that in the "Expiration Date/Time" field the calendaring tool provides a **Now** button in case you want to terminate public URL access for the object immediately. Be sure to click **Apply** to save your changes.

Note that if you change the expiration date/time of a public URL and click **Apply** this will result in the generation of a **new URL**. This is because the expiration times is part of the URL -- so changing the expiration time necessarily results in a new URL. By contrast, changing the maximum allowed downloads does not result in a new URL.

6.5.3.4. Set Object Lock Attributes on an Object

Path: **Buckets & Objects** → **Objects** → <bucketname> → <find object> → **Properties** → **Object Lock**



For an object that has been uploaded to a bucket that has Object Lock enabled, an **Object Lock** tab will appear in the object's **Properties** interface. Here you can set Object Lock attributes on the object. For an object for which there are multiple versions in the bucket, each object version has its own **Properties** interface including an **Object Lock** tab, and object lock attributes can be configured separately for each object version.

Note To access the **Properties** interfaces for older versions of an object, toggle the object list display to **Show Versions** if it is not already showing all object versions. If the object list is toggled to **Hide Versions** you can only access the **Properties** interface for the current version of each object.

Within the limits described below, Object Lock attributes that you apply to an individual object version will override any [default Object Lock attributes](#) that had been automatically applied to the object version at its upload time as a result of the bucket's default Object Lock configuration.

To set Object Lock attributes on an object version:

1. In the **Object Lock** tab of the object version's **Properties** interface, make your desired changes to the object version's current lock attributes:
 - **Lock Mode.** Changing the lock mode for an object is allowed only if the current lock mode is Governance or None. You cannot change the mode if the current mode is Compliance. For more information on Governance Mode and Compliance Mode see ["Configure Object Lock Properties for a Bucket"](#) (page 337).
 - **Retain Until Date.**
 - If the object's current mode is Governance, or if the object's current mode is None and you are changing the mode to Governance or Compliance, you can set the Retain Until Date to whatever date you desire.
 - If the object's current mode is Compliance, you can change the Retain Until Date to a later date than its current value. You cannot change the Retain Until Date to a sooner

date than its current value (you cannot shorten the object's retention period if the current mode is Compliance).

- **Legal Hold.** Legal Hold prevents the object from being deleted through the S3 interface for an indefinite period of time, until you explicitly remove the Legal Hold from the object. When the time comes that you want to remove the Legal Hold, you can do so in the **Object Lock** tab of the object's **Properties** interface.

Note For a summary of protections that the S3 API provides to locked objects see ["Protections for Locked Objects"](#) (page 229).

2. Click **Save**.

Note Your change applies only to a specific version of the object:

- If you had the object list view toggled to **Hide Versions** rather than **Show Versions**, and you opened an object's **Properties** interface and changed the object's lock properties, your change applies only to the current version of the object, not to any older versions of the object.
- Your change does not apply to future versions of the object. If one or more additional versions of the object are uploaded in the future, those versions will inherit the bucket's default Object Lock configuration.

6.5.4. List or Search for Objects

Subjects covered in this section:

- *Introduction (directly below)*
- ["Objects That Indicate a Metadata Error"](#) (page 358)
- ["List Object Versions Created During a Specified Time Range"](#) (page 359)

NAME	SIZE	LAST MODIFIED	ACTIONS
FullListOfCommands.png	265.4 KiB	Feb-25-2025 01:37 AM +0800	Properties Delete
GetDefaultBucketThreshold.png	154.2 KiB	Feb-25-2025 01:37 AM +0800	Properties Delete
GetQoSMeterStatus.png	169.3 KiB	Feb-25-2025 01:37 AM +0800	Properties Delete
GetQoSRunTimeInterval.png	156.2 KiB	Feb-25-2025 01:37 AM +0800	Properties Delete
GetRestrictedBuckets.png	156.3 KiB	Feb-25-2025 01:37 AM +0800	Properties Delete
GetThresholdForBucket.png	164.0 KiB	Feb-25-2025 01:37 AM +0800	Properties Delete
InsightEngine_SetupGuide_v2.1.x.pdf	644.6 KiB	Feb-26-2025 08:21 PM +0800	Properties Delete
QoSStatus.png	150.0 KiB	Feb-25-2025 01:37 AM +0800	Properties Delete
RemoveThresholdForBucket.png	164.0 KiB	Feb-25-2025 01:37 AM +0800	Properties Delete
ReportBucketQoSStatus.png	167.7 KiB	Feb-25-2025 01:37 AM +0800	Properties Delete

In the **Objects** interface the objects in your bucket will be listed alphabetically. If you have many objects the listing will span across multiple pages, with 10 objects per page (by default). Your **"folders"**, if any, are also listed within this alphabetical listing. You can browse through the objects and folders list by clicking the page numbers at the bottom of the **Objects** interface.

If you have [versioning](#) enabled on your bucket, and you want to list the versions of an object, in the **Objects** interface click **Show Versions**. When versions are shown, under each object name the interface lists all stored versions of that object, ordered from the current (most recently uploaded) version to the oldest version. Each version is identified by a system-generated alphanumeric identifier, and for each version the upload timestamp is shown (the date and time that particular version of the object was uploaded to the system). See also the option to ["List Object Versions Created During a Specified Time Range"](#) (page 359)

You can search for an object or objects by clicking **Search by Prefix**. In the pop-up dialog that displays, enter either:

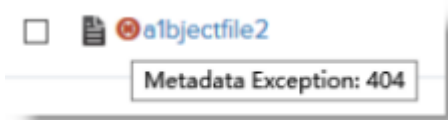
- An object name prefix, including folder path (if applicable) -- to retrieve all objects that start with that prefix. Examples:
 - *Video/* -- Retrieves all objects that start with prefix *Video/* (or in common file system terminology, all files under the *Video* "folder")
 - *Video/2024/InstallerDemo* -- Retrieves all objects that start with prefix *Video/2024/InstallerDemo* (or in common file system terminology, all files that are under the *Video/2024* "folder" and have file names starting with the string *InstallerDemo*)
- A full object name, including folder path (if applicable) -- to retrieve one specific object. Example:
 - *Video/2024/InstallerDemo_2024-05-22.mpg*

For background information about how "folders" are implemented in an object storage system like HyperStore, see ["Note about 'folders' in an object storage system"](#) (page 345).

Note The search matching is **case sensitive**. Be sure to use the correct case when entering your search term.

6.5.4.1. Objects That Indicate a Metadata Error

On rare occasions an object in the display list may have a red "X" beside its name, and if you hold your cursor over it a "Metadata Exception: <number>" error message appears.



This indicates that there was a problem retrieving the object's metadata. You will not be able to download such an object or edit its properties.

A "403" error means that you don't have read permissions for this object -- such as if another user (with permissions that you granted) uploaded an object to your bucket using a third party application, without granting read permissions to you as the bucket owner.

A "404" error means that the system cannot locate the object's metadata:

- If you previously deleted this object, delete the object again and then it should no longer appear in the object list.
- If you did not previously delete this object, and if you have a copy of the object on your computer, upload the object again and that should fix the error.
- If you did not previously delete this object, and if you do not have another copy of the object, contact Support.

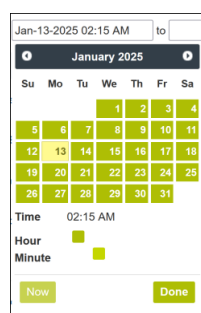
6.5.4.2. List Object Versions Created During a Specified Time Range

If you wish you can list just the object versions that were created during a time period that you specify. This feature is supported only for buckets that have [versioning](#) enabled, and the interface for specifying a filtering date-time range displays only if you select **Show Versions** in the upper right of the **Objects** interface.

Filter by Date Range: to  

IMPORTANT ! When objects are uploaded to the HyperStore system, they are assigned creation stamps indicating the current time **in the time zone in which the system is located**. These are the timestamps that display in the "Last Modified" column in the **Objects** list view. **If you and your browser are in a different time zone** than the HyperStore system, then when specifying your desired filtering date-time range you should specify start and end times as they would be in the time zone in which the HyperStore system is located. For example if the HyperStore system is located in a time zone two hours ahead of where you are located, and you want to see a list of object versions that were created between 12:00PM January 15th and 12:00PM January 16 your time, specify the filter start and end times as 2:00PM January 15th and 2:00PM January 16 (the corresponding time range in the location of the HyperStore system).

To set a filtering range, first click in the left field and use the calendar that pops up to specify the starting date-time for the filtering range; and then do likewise in the right field to specify the ending date-time for the filtering range. The Hour and Minute controls below the calendar work as sliders -- slide the green squares to set the desired time.



Once you've set a date-time range, the icons to the right of the date-time range fields work as follows:

	Clear your selected date-times (to start over and choose different date-times).
	Proceed with filtering the object version list based on the date-time range that you've entered in the fields.

6.5.5. Download an Object

In the [Objects](#) interface, first select the bucket that you want to work with. Then in the object list that displays for that bucket, [find the object](#) that you want to download. To download the object, click on the object name and then choose whether to directly open the object or to save it to your computer.

Note In your object list display, certain object names may have beside them a left or right-pointing arrow icon that indicates that the object has been subject to auto-tiering. For information about the icons and about retrieving such objects, see ["Retrieve a Tiered Object" \(page 360\)](#).

6.5.5.1. Downloading an Object Version

If you have [versioning](#) enabled on your bucket, and you want to download a particular version of an object, in the [Objects](#) interface click **Show Versions**. When versions are shown, under each object name the interface lists all stored versions of that object, ordered from the current (most recently uploaded) version to the oldest version. Each version is identified by a system-generated alphanumeric identifier, and for each version the upload timestamp is shown (the date and time that particular version of the object was uploaded to the system). To download a version of an object, click the alphanumeric identifier.

6.5.6. Retrieve a Tiered Object

Topics covered in this section:

- ["Downloading or Restoring a Tiered Object" \(page 360\)](#)
- ["Icons for Tiered or Restored Objects" \(page 361\)](#)
- ["Restoring a Tiered Object Version" \(page 362\)](#)
- ["Deleting a Tiered Object" \(page 362\)](#)
- ["Deleting a Bucket That's Configured for Tiering" \(page 363\)](#)

6.5.6.1. Downloading or Restoring a Tiered Object

In the [Objects](#) interface, first select the bucket that you want to work with. If the bucket that you're [browsing or searching](#) through has been configured for auto-tiering, any objects that have been auto-tiered will be marked with a [special icon](#). Because these objects are currently stored in a remote tiered storage system (such as Amazon, Azure, or Google), you may not be able to directly download these objects. Whether they are directly downloadable or not depends on where they are stored — for example, objects in Amazon Glacier are never directly downloadable — and on the user-defined bucket lifecycle configuration that governed the auto-tiering of the objects (specifically, whether the bucket lifecycle configuration supports "streaming" GETs of tiered objects rather than requiring a restore operation).

You can try directly **downloading** an auto-tiered object by clicking on its object name. If direct download (streaming) is not supported for the object, a response message will indicate that you need to temporarily restore a local copy of the object rather than directly downloading it.

To **restore** one or more auto-tiered objects:

1. Click the checkbox(es) to the left of the object name(s).
2. Click the **Restore** button.
3. In the dialog that pops up, use the "Restore" number of days field to enter the number of days for which you want the restored object(s) to be locally available in your.

Note For an object that has been tiered to Glacier, the AWS "Standard" retrieval type is used to temporarily retrieve the object into AWS S3 (for one day) so that it then can be restored from AWS S3 to your HyperStore system for your specified number of days.

4. Click the **OK** button.

Note that restore does not happen instantly — it can take up to six hours before auto-tiered objects are restored to your local HyperStore system (or up to nine hours for objects that have been auto-tiered to Amazon Glacier). In the interim, the object is marked with a [special icon](#) that indicates that it's in the process of being restored. During this stage you cannot download the object.

After an object has been restored (as indicated by a different [special icon](#)), you can download it in the normal way.

Note For special considerations for versioned objects, see ["Restoring a Tiered Object Version"](#) (page 362).

Note The Restore "number of days" duration is implemented specifically as follows: Starting from the day and time at which you submit the restore request, the restore duration will end at the **first midnight after your specified number of days have passed**. For example: If on the 15th of a month, in the morning, you submit an object restore request with a specified restore period of 3 days, then -- with 3 days having passed by the morning of the 18th -- the object's restore duration will end at 00:00 (midnight) of the 19th.

Note In the case of objects that have been tiered to Spectra BlackPearl, if the tape on which an object resides has been ejected the restore attempt will fail and an error will be written to the S3 application log. The logged error message will indicate the ID of the tape that needs to be reinserted in order to restore the object.

6.5.6.2. Icons for Tiered or Restored Objects

In the [Objects](#) interface, the document icon to the left of object names will indicate if an object is transitioned (tiered), restoring, or restored. The icons feature right-pointing or left-pointing red or blue arrows.

Icon	Meaning
	Object has been transitioned to a remote tiered storage system.
	A copy of the transitioned object is in the process of being restored to the local HyperStore system. Download of the object is not yet supported.
	A copy of the transitioned object has been restored to the local HyperStore system. Download is supported.

6.5.6.3. Restoring a Tiered Object Version

If you have **versioning configured on your bucket** as well as auto-tiering, then it may be that certain versions of your objects have been auto-tiered to an external destination system. A common configuration is to have object versions get auto-tiered after they become non-current. For example, the first version of an object would get auto-tiered after a second version of that same object is uploaded to the local system; and the second version would get auto-tiered after a third version of the object is locally uploaded.

If you have your bucket configured for auto-tiering of non-current object versions, retrieving object versions works like this:

- In the **Objects** interface, click **Show Versions** so that all object versions display. If you are using auto-tiering of non-current object versions, typically the newest version of each object will be in your local bucket and older versions of each object will have been auto-tiered to the external destination system (as indicated by the [special icon](#) next to transitioned object versions). You can retrieve the **current object version** (stored locally) in the usual way, by simply clicking on the object version name.
- If your bucket's auto-tiering configuration allows for streaming GETs of auto-tiered objects (which is the default configuration), you can retrieve auto-tiered non-current object versions by clicking on the version name of the non-current object version that you want to retrieve.
- If your bucket's auto-tiering configuration does not allow for streaming GETs of auto-tiered objects, the way to locally retrieve an auto-tiered non-current object version is to restore it by clicking the **Restore** button to the right of the object version. In the number of days field that appears, enter the number of days for which you want the restored object version to be locally available in your HyperStore system.

Note that restore does not happen instantly — it can take up to six hours before an auto-tiered object version is restored to your local system (or up to nine hours for object versions that have been auto-tiered to Amazon Glacier). In the interim, the object version is marked with a [special icon](#) that indicates that it's in the process of being restored. During this stage you cannot download the object version.

After an object version has been restored (as indicated by a different [special icon](#)), you can download it in the normal way.

6.5.6.4. Deleting a Tiered Object

If you want to **delete** an object that has been auto-tiered, you can do so by deleting the object through the **Objects** interface, just like deleting any other object. You do not need to restore the object first. To delete the object, click **Delete** at the right side of the object's display row. Then click **OK** in the confirmation dialog.

By default system behavior, this deletes the object's local metadata (and therefore the object will no longer be accessible through the local HyperStore system) and it also deletes the tiered object from the tiering destination system. For information about changing this default behavior see "[s3.tiering.forwardDeletes](#)" (page 836).

IMPORTANT ! Do not overwrite or delete tiered objects directly through the destination system's interfaces. Doing so will cause a discrepancy between the local metadata in HyperStore and the actual data in the destination bucket. If you want to overwrite or delete tiered objects, do so through the CMC or an S3 application accessing the HyperStore S3 Service . In the case of auto-tiering from one HyperStore region to another HyperStore region, any overwriting or deleting of objects should be done through the source bucket not the destination bucket.

6.5.6.5. Deleting a Bucket That's Configured for Tiering

As with any bucket, you can only delete a bucket that's configured for tiering if the bucket is empty. "Empty" here means that you've deleted all objects from it, including objects that have been tiered. After deleting all objects in the bucket, you can delete the bucket by going to the [Buckets](#) list view and clicking the **Delete** button on the far right of the bucket's display row.

If you delete an empty local bucket that has been configured for auto-tiering, by default the system will also automatically delete the tiering destination bucket. For information about changing this default behavior see "[s3.tiering.forwardDeletes](#)" (page 836).

6.5.7. Delete an Object

In the [Objects](#) interface, first select the bucket that you want to work with. Then in the object list that displays for that bucket, [find the object](#) that you want to delete. To delete the object, click **Delete** at the right side of the object's display row. After you click **OK** in the confirmation dialog, the object is deleted.

To delete multiple objects from storage, click on the checkboxes to the left of the object names, and then click the **Delete** button at the bottom right of the **Objects** interface. After you click **OK** in the confirmation dialog, the objects are deleted.

Note Deleting an object through the CMC interface (which on the back end calls the S3 *DeleteObject* API method or the S3 *DeleteMultipleObjects* API method) 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. (The frequency with which the batch processing job runs is configurable by the setting "[s3.batchOperations.delete.queuePollInterval](#)" (page 814).)

6.5.7.1. Deleting an Object Version

If you have [versioning](#) enabled on your bucket, and you want to delete a particular version of an object, in the [Objects](#) interface click **Show Versions**. When versions are shown, under each object name the interface lists all stored versions of that object, ordered from the current (most recently uploaded) version to the oldest version.. Each version is identified by a system-generated alphanumeric identifier, and for each version the upload timestamp is shown (the date and time that particular version of the object was uploaded to the system). To delete a version of an object, click **Delete** at the right side of the object version's display row. After you click **OK** in the confirmation dialog, the object version is deleted.

Note Deleting any one version of any object -- including the newest version -- does not delete the other versions of the object.

If you delete a versioned object when the **Objects** interface is in "Hide Versions" mode, it will appear that the object has been entirely deleted. But in fact only the newest version is deleted, and all other versions of the object remain in the system. You can see this if you click **Show Versions**.

To delete multiple versions of an object, click on the checkboxes to the left of the object versions, and then click the **Delete** button at the bottom right of the **Objects** interface. After you click **OK** in the confirmation dialog, the object versions are deleted.

6.5.7.2. Deleting an Auto-Tiered Object

See ["Deleting a Tiered Object" \(page 362\)](#).

6.5.8. Object Naming Restrictions

The following control characters cannot be used anywhere in an object name (a folder name or file 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

Also, an object name may result in 400 Bad Request or be stored as a different name if the supplied object name:

- Consists only of "."
- Consists only of ".."
- Contains a combination of "." and "/", or ".." and "/"

Examples of character sequences that will result in 400 Bad Request:

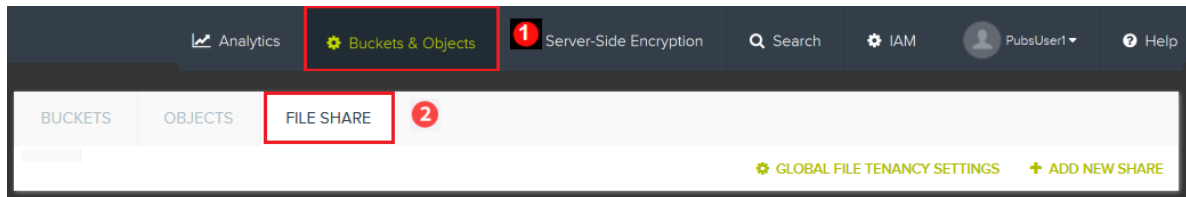
- .
- ..
- ./
- ../
- ./.
- ./..
- ../.
- ../..
- ../a
- ../a/
- a/././b

Examples of character sequences that will be stored as a different name:

Supplied Object Name	Stored As
./a	a
./a/	a/
a//b	a/b
a/./b	a/b
a/./b	b
a/././b	a/././b

6.6. File Share

Path: **Buckets & Objects** → **File Share**



Note This CMC functionality is available only to group admins for whom file services are enabled or regular users for whom file services are enabled. This functionality is not available to system admins.

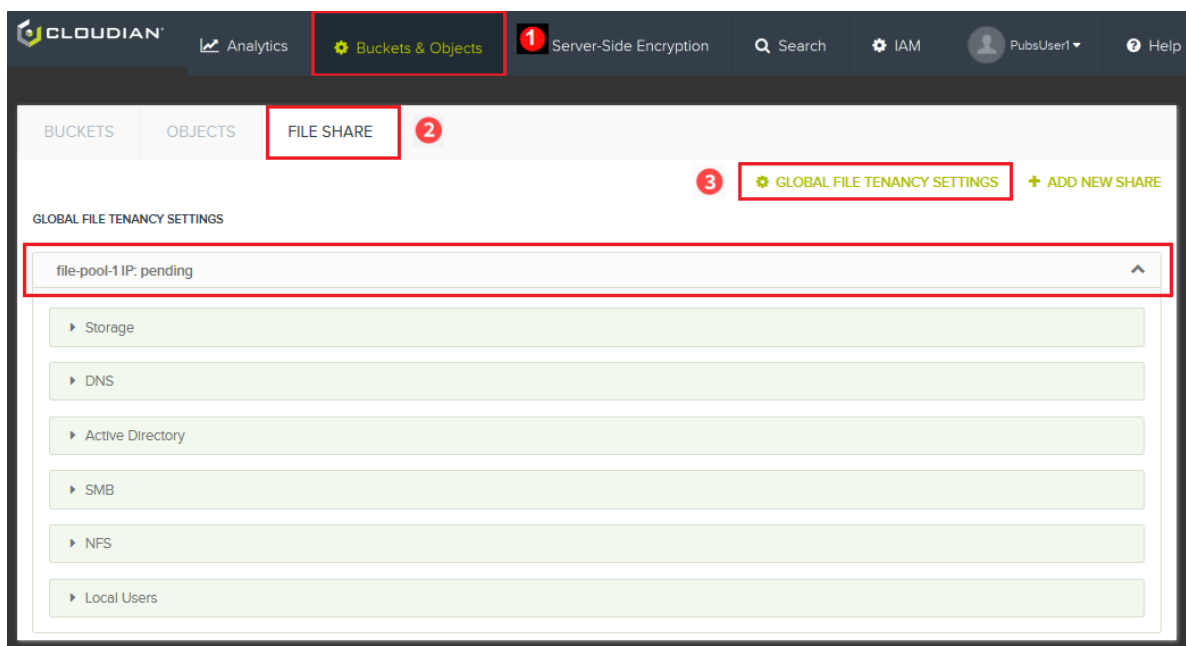
Supported tasks:

- ["Set Global Properties for File Shares"](#) (page 365)
- ["Add a File Share"](#) (page 368)
- ["Set File Share Custom Properties"](#) (page 373)
- ["Open a File Share and Provide Its Endpoint To Users"](#) (page 378)
- ["Edit or Delete a File Share"](#) (page 383)
- ["View File Share Usage Statistics"](#) (page 382)

For an overview of the file share set-up workflow see ["Set Global Properties for File Shares"](#) (page 365).

6.6.1. Set Global Properties for File Shares

Path: **Buckets & Objects** → **File Share** → **Global File Tenancy Settings**



Note This CMC functionality is available only to group admins for whom file services are enabled or regular users for whom file services are enabled. This functionality is not available to system admins.

Subjects covered in this section:

- *Introduction (directly below)*
- ["High Level Workflow for Setting Up File Shares" \(page 366\)](#)
- ["Configuring Global File Tenancy Settings" \(page 366\)](#)

If the object storage system administrator has enabled the file sharing feature for your user account, you can use the functionality in the **Buckets & Objects** → **File Share** tab to set up file shares for end users who use Server Message Block (SMB) client applications or Network File System (NFS) client applications. Each file share that you set up will use one of your object storage buckets for back-end data storage. When your SMB or NFS users upload a file to a file share, the File Service -- the part of the system that services SMB and NFS requests -- will archive the data to the corresponding object storage bucket as well as caching the file in the File Service front-end cache. When your SMB or NFS users request a file that they had previously uploaded, the File Service will return the file from the front-end cache, or if the file is no longer present in the cache (due to cache space management policies) the File Service will retrieve the data from the object storage bucket and return it to the SMB or NFS client as well as re-caching the file. Optionally you can have objects directly uploaded to the bucket through S3 applications be visible and accessible to SMB or NFS clients using the bucket's corresponding file share, if you enable the "bi-modal access" feature for the file share.

6.6.1.1. High Level Workflow for Setting Up File Shares

The general workflow for setting up file shares is:

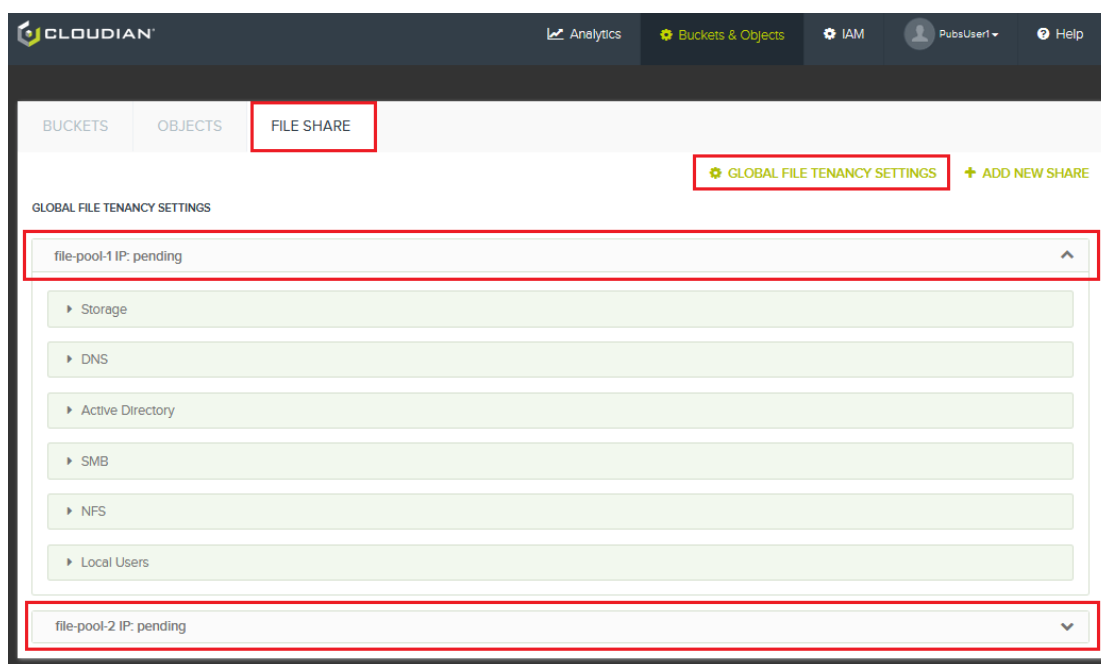
1. If you haven't already done so, for each file share that you want to set up, create an object storage bucket that you will use as the back-end storage for the file share (each file share will require its own corresponding object storage bucket that is not used by any other file share) and enable versioning on each of those buckets. Versioning is mandatory for buckets that will be used as file share storage back-ends. For details about this task see ["Add a Bucket" \(page 307\)](#) and ["Enable/Disable Versioning for a Bucket" \(page 333\)](#).
2. Configure "global file tenancy settings" which will be used as the default configuration settings for all file shares that you create. This step is described further below.
3. Create a file share and select which of your buckets the file share will use as its storage back-end. For details on this task see ["Add a File Share" \(page 368\)](#).
4. Configure custom settings specifically for the file share (including S3 security credentials and the option to enable bi-modal access for the share). For details on this task see ["Set File Share Custom Properties" \(page 373\)](#).
5. Open the file share and provide the file share endpoint to your users. For details on this task see ["Open a File Share and Provide Its Endpoint To Users" \(page 378\)](#).

You can then repeat Steps 3, 4, and 5 for each file share that you want to set up.

6.6.1.2. Configuring Global File Tenancy Settings

To access the global settings to apply as defaults for your file shares, in the **Buckets & Objects** → **File Share** tab click **Global File Tenancy Settings**. Note that:

- When you look through these settings for the first time and see default values for many of them, these are default values that have been established by the system administrator. If you wish you can edit such settings here and establish global default values for your file shares that are different than the default values initially set by the system administrator.
- The configuration values for your Global File Tenancy Settings become the default configuration values for each file share that you will create. For each share that you create you will have the option to set custom configuration values for the share. For any settings that you do not customize for a share, the share will use the global default values. Note that some types of global settings, such as Active Directory settings and the Local User settings, can not be customized for individual shares.
- If the system administrator has granted you access to multiple File Service pools, you can configure the default values for each File Service pool independently, and each pool's default values will serve as the defaults for file shares that you publish to that pool. (Click a pool name to display the configurable settings for that pool.)



Note also that if you make edits to your Global File Tenancy Settings **after** you have created file shares and customized some settings specifically for those file shares, then:

- For settings for which an existing file share is using the global default value, your change to the global setting will propagate to the share (the share will start using the updated setting value)
- For settings that you've customized for the share -- rather than using the global default value -- your change to the global setting will not affect the share (the share will continue to use the customized setting value)

For individual setting descriptions see ["File Services Configuration Setting Matrix" \(page 384\)](#). This setting description matrix also indicates the level(s) at which you can edit each setting (the global level applied to all your shares, and/or the share-specific level).

IMPORTANT ! Although the CMC does not prevent you from having both SMB and NFS enabled for the same file share, the **File Service does not support multi-protocol write access to the same file share** -- and potential conflicts or corruptions may occur if the same file share is concurrently written to

by SMB clients and NFS clients. Therefore, if you have both SMB and NFS enabled on the same file share, then you must implement a safe access policy such as SMB write access, NFS read-only.

6.6.2. Add a File Share

Path: **Buckets & Objects** → **File Share** → **Add New Share**

Note This CMC functionality is available only to group admins for whom file services are enabled or regular users for whom file services are enabled. This functionality is not available to system admins.

Subjects covered in this section:

- *Adding a File Share (directly below)*
- *"Geo-Distributed File Shares" (page 370)*
- *"WORM File Shares (Object Lock)" (page 372)*

If the object storage system administrator has enabled the file sharing feature for your user account, you can use the functionality in the **Buckets & Objects** → **File Share** tab to set up file shares for users who use Server Message Block (SMB) client applications or Network File System (NFS) client applications. For an overview of the file share set-up workflow see ["Set Global Properties for File Shares" \(page 365\)](#). Part of the workflow is described in detail below: adding a file share.

Before adding a file share, decide which of your buckets you will use as the storage repository for the file share. The bucket:

- **Should be a new bucket** that does not yet have any content in it. If you do not currently have any new, unused buckets, create a new bucket to use as the storage repository for the file share. (Note that the **File Share** tab will not display if you have not yet created any buckets.)
- **Must have [Versioning](#) enabled.** As SMB or NFS file share clients modify files over time, the older versions of the files are retained in the object storage bucket as well as the current versions. (This enables system administrators to take 'snapshots' of the file share database as time goes on and, if needed, to revert to a recent earlier version of the file share database.)

Also before adding a file share, be aware that **all bucket usage associated with the file share will be attributed to your user account**, as the bucket owner.

To add a file share:

1. In the **File Share** tab click **Add New Share**.
2. Enter a name for the new share (maximum 248 characters, and the only special characters allowed are "-" or "_").
3. From the Bucket Name drop-down list, choose one of your buckets to serve as the storage back-end for the file share.
4. Under "Target File Service Pool" select a File Service pool or pools to which to publish the file share. Depending on how system administrators have configured the system and your File Service access privileges:

- There may be only one File Service pool displayed -- in which case that pool will automatically be selected.

OR

- There may be multiple File Service pools displayed, from the same service region that your selected bucket is in -- in which case:
 - You can leave the "Geo-Distribution" option disabled and select just one of the pools, to have the file share published to just that File Service pool and its end users.

The screenshot shows the 'ADD NEW SHARE' form. The 'Share Name' field contains 'pubs-standard-share'. The 'Bucket Name' dropdown shows 'pubsuser1.bucket1'. The 'Geo-Distribution' toggle is set to 'DISABLED'. Under 'Target File Service Pool', 'File-Pool-1' is selected with a radio button, and 'File-Pool-2' is unselected.

OR

- You can enable the "Geo-Distribution" option and then select multiple File Service pools, to create a geo-distributed file share that is automatically kept in sync as it is used by the end users of the pools that you've selected. For more information about geo-distributed file shares see ["Geo-Distributed File Shares" \(page 370\)](#).

The screenshot shows the 'ADD NEW SHARE' form. The 'Share Name' field contains 'pubs-distributed-share'. The 'Bucket Name' dropdown shows 'pubsuser1.bucket1'. The 'Geo-Distribution' toggle is set to 'ENABLED'. Under 'Target File Service Pool', both 'File-Pool-1' and 'File-Pool-2' are selected with checkboxes.

Note In an object storage system that has multiple service regions, the "Target File Service Pool" section will only show File Service pools that are in the same region as your selected bucket. You cannot publish a file share to a File Service pool that is in a different service region than the bucket that you are using for back-end storage. If you want to create a "geo-distributed" file share that spans across two service regions, see the instructions in ["Special Case: Geo-Distributed File Shares Spanning Two Service Regions" \(page 371\)](#).

5. Click **Create**.

The share will take a few moments to create, after which you should see a success message.

BUCKETS OBJECTS FILE SHARE

GLOBAL FILE TENANCY SETTINGS + ADD NEW SHARE

ADD NEW SHARE

Share Name: pubs-standard-share

Bucket Name: pubsuser1.bucket1

Geo-Distribution: ☒ DISABLED

Target File Service Pool

☒ File-Pool-1 ☐ File-Pool-2

New share created

CANCEL CREATE

You can then refresh your screen and see the new share in your share list.

BUCKETS OBJECTS FILE SHARE

GLOBAL FILE TENANCY SETTINGS + ADD NEW SHARE

SHARE NAME	STATUS	BUCKET	
pubs-standard-share	☒ Closed(CLOSED)	pubsuser1.bucket1	Properties Delete

Next configure any settings that you want to apply specifically to this new file share, as described in ["Set File Share Custom Properties"](#) (page 373).

6.6.2.1. Geo-Distributed File Shares

Depending on how administrators have configured the object storage system, and the File Service access privileges that administrators have granted you, you may be able to create "geo-distributed" file shares (as described above in Step 4 of the procedure for adding a share). A geo-distributed file share is published to two or more File Service pools, typically in different locations and servicing different end users.

For example you could publish the file share to two different pools named "DC1 File Service" and "DC2 File Service", with your bucket named "my.bucket" serving as the back-end storage for the geo-distributed file share. When end users of the "DC1 File Service" pool create, update, or delete a file, that change becomes visible not only to "DC1 File Service" end users but also to "DC2 File Service" end users. Likewise, when end users of the "DC2 File Service" pool create, update, or delete a file, that change becomes visible not only to "DC2 File Service" end users but also to "DC File Service" end users.

This synchronization of the file share view across different File Service pools is rapid but not instantaneous. There will be a brief delay before a new change is tiered from the originating File Service pool to the back-end object storage bucket (typically 20 seconds but may vary depending on how administrators have configured the system); and another brief delay as the file share metadata synchronization to the other File Service pool is processed. The file share views from the two different File Service pools are "eventually consistent".

Note In the uncommon event of near-simultaneous conflicting changes made by users of the different File Service pools -- for example a user of "DC1 File Service" and a user of "DC2 File Service" make changes to the same file, nearly simultaneously -- the latest version tiered to the back-end object storage bucket "wins" and becomes the current version visible to users of both File Service pools. Meanwhile the version that "lost", by having been superseded by the slightly later winning version, still exists as a non-current object version in the bucket (since versioning is enabled on the bucket).

Converting a Standard File Share to a Geo-Distributed File Share

If you know from the outset that you want a share that you are creating to be a geo-distributed share, you can create it that way as described in procedure for adding a share.

Alternatively, if you've created a standard, not geo-distributed share, and then subsequently you decide that you want to convert it to a geo-distributed share, you can do so as follows:

1. [Close](#) the existing standard share.
2. Follow the procedure for adding a new share, but with these distinctions:
 - As the "Share Name" type the **name of the existing standard share** (must be exactly the same name)
 - From the "Bucket Name" drop-down list select the bucket that is being used by the existing standard share
 - Set the "Geo-Distribution" toggle to Enabled
 - As the "Target File Service Pool" select **only the additional File Service pool** to which you want to publish the share (not the pool to which the share has already been published)
3. Click **Create**. Wait a few minutes, then refresh the screen.

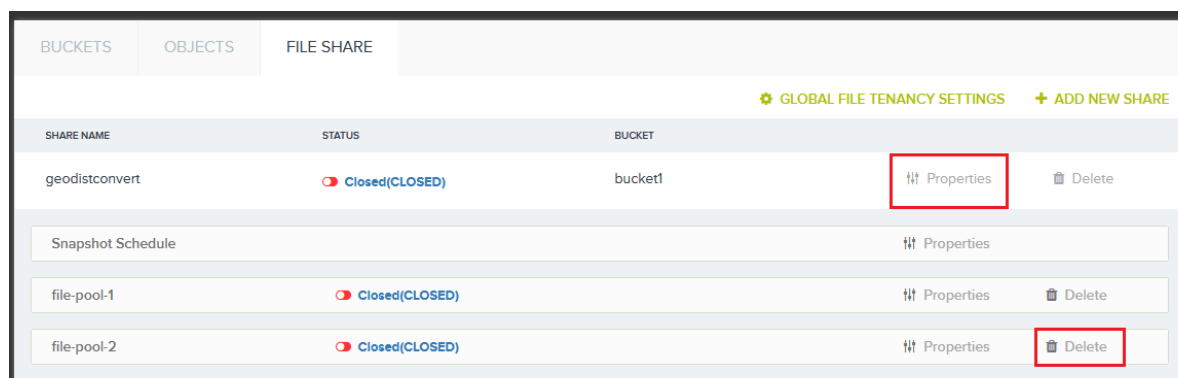
This converts the standard share into a geo-distributed share.

If you wish you can then ["Set File Share Custom Properties"](#) (page 373) for the share in the additional File Service pool to which you've published it.

Next you can [open the share and obtain the front-end IP address](#) for the share in the additional File Service pool to which you've published it, so that you can provide that share access information to the end users of that pool. (Note that the share will have a different front-end IP address in each File Service pool.)

Converting a Geo-Distributed File Share to a Standard File Share

To convert a geo-distributed share to a standard share, first close the share then go to the share Properties and delete the share from just the File Service pool to which you no longer want to publish the share. Then reopen the share.



BUCKETS	OBJECTS	FILE SHARE		
		GLOBAL FILE TENANCY SETTINGS + ADD NEW SHARE		
SHARE NAME	STATUS	BUCKET		
geodistconvert	● Closed(CLOSED)	bucket1	Properties	Delete
Snapshot Schedule			Properties	
file-pool-1	● Closed(CLOSED)		Properties	Delete
file-pool-2	● Closed(CLOSED)		Properties	Delete

Special Case: Geo-Distributed File Shares Spanning Two Service Regions

If your object storage system is deployed across multiple service regions -- typically these would correspond to different geographic regions, potentially very far from each other -- then each time you create a bucket in the Console you choose which service region the bucket will be based in. In some cases system administrators will have set up File Service pools in each of the system's multiple service regions. If this is true of your system, then you have the option of creating a geo-distributed file share that is deployed to two different service regions

-- although technically what you would be doing is creating two separate file shares, backed by two separate buckets, and the system will keep the two geographically distributed file shares in sync.

Note these restrictions on this feature:

- **Geo-distributed file shares spanning more than two service regions are not supported.**
- **Buckets that have Object Lock enabled cannot be used** for geo-distributed file shares spanning two service regions.

To create geo-distributed file shares across two service regions follow the procedure below, using the Console. For convenience, in the instructions the two regions will be called *Region1* and *Region2*, the two buckets will be called *bucket1* and *bucket2*, and the two file shares will be called *Share1* and *Share2*.

1. Create *bucket1* in *Region1*, and enable versioning on the bucket. (For more details on how to take these actions see ["Add a Bucket"](#) (page 307) and ["Enable/Disable Versioning for a Bucket"](#) (page 333).)
2. Create *bucket2* in *Region2*, and enable versioning on the bucket.
3. Enable Cross-Region Replication on *bucket1* as a source bucket, and make the destination bucket *bucket2*. (For more details on how to take this action see ["Configure Cross-Region Replication for a Bucket"](#) (page 331).)
4. Enable Cross-Region Replication on *bucket2* as a source bucket, and make the destination bucket *bucket1*.
5. Add a file share *Share1* using *bucket1* as the back-end storage for the share and selecting as the Target File Service Pool a pool based in *Region1* (note that when you choose *bucket1* as the back-end storage bucket, only pools based in *Region1* will display as options in the Target File Service Pool section of the interface). (For more details on how to take this action see ["Add a File Share"](#) (page 368).)
6. Add a file share *Share2* using *bucket2* as the back-end storage for the share and selecting as the Target File Service Pool a pool based in *Region2*.

Next configure any settings that you want to apply specifically to these new file shares, as described in ["Set File Share Custom Properties"](#) (page 373).

Note In most respects, with file shares geo-distributed across two service regions the share content is kept in sync between the participating File Service pools just like for a geo-distributed share within a single region. However there is one exception in the case of ACL-only changes made to a file (changes made to a file's permissions rather than to the file's content). After a created file has been synced across regions (so that the file is available to File Service users in both regions) any subsequent ACL-only changes made to the file by users within the same region that the file was created will be synced to the file's ACL in the other region -- but ACL-only changes made to the file by users in the region that the file did not originate from will **not** be synced to the file's ACL in the region in which the file originated.

6.6.2.2. WORM File Shares (Object Lock)

If you wish you can create a file share that uses a bucket on which you have enabled [Object Lock](#). The result will be a file share that operates in WORM mode (Write Once Read Many). The main behaviors of a WORM file share backed by an Object Lock enabled bucket are:

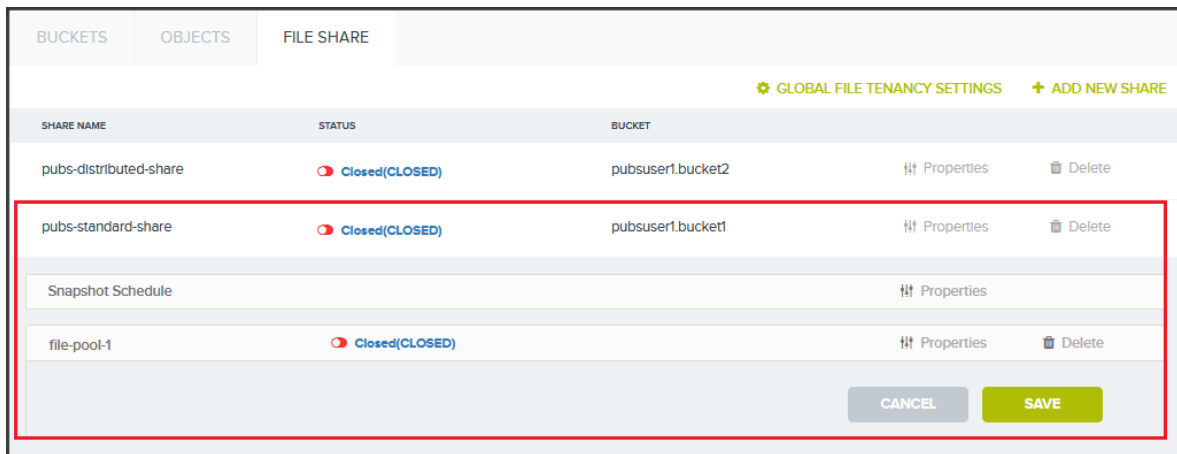
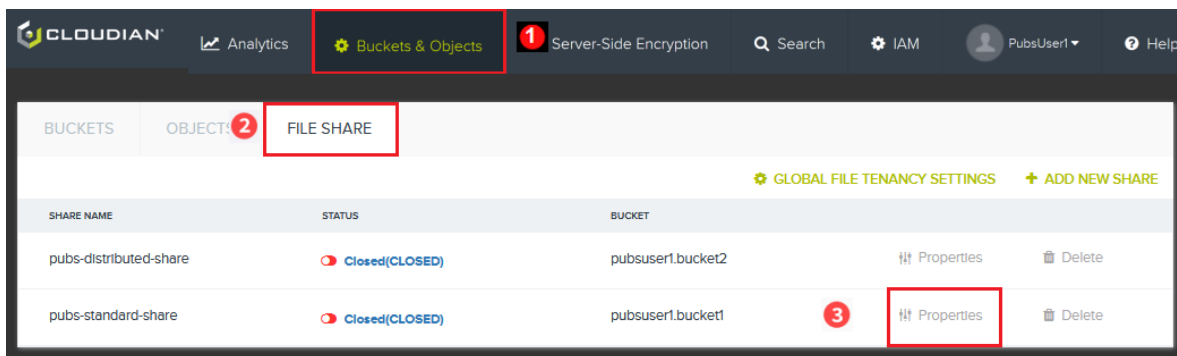
- Files created through the File Services interface (SMB or NFS front end) will automatically take on the retention properties defined in the bucket's default Object Lock configuration. For example if the back-

end bucket's default Object Lock configuration specifies 30 days retention then each new file created through the File Services interface will have WORM protection for 30 days. Also, as a file is archived into object storage, the resulting object will have the appropriate S3 Object Lock headers indicating not only the retention date but also the S3 Object Lock "mode" (Governance mode or Compliance mode; the mode is relevant only for direct S3 access to the archived object and does not impact how the file is managed by the File Services front end).

- When a file is being created in a WORM file share, the file moves into WORM mode after 40 seconds of inactivity by the client application (40 seconds with no change to the file's *mtime*).
- During its retention period, a WORM protected file cannot be deleted or modified through the File Services interface. After the file's retention period ends, the file can be deleted through the File Services interface but not modified through the File Services interface.
- File retention attributes cannot be set through the File Services interface. Any and all retention related configuration occurs exclusively on the S3 side.
- 'Legal Hold' is not supported at the file system level.
- The File Service detects and loads a bucket's default Object Lock configuration only when a share is being opened. So before opening the share for the first time be sure that you've [set the bucket's default Object Lock configuration](#). And subsequently if you want to change the default Object Lock configuration for a bucket to which a file share is attached, you must close the share, make your change to the bucket's default Object Lock configuration, then re-open the share.
- If you enable [bi-modal access](#) on the file share (as well as enabling Object Lock on the bucket):
 - Objects created in the bucket directly via the S3 interface will become available through the file share and will have whatever retention attributes they were assigned through the S3 interface. The retention attributes could come from the bucket's default Object Lock configuration or from object-specific Object Lock attributes specified in the S3 upload operation for the object.
 - Although a WORM protected file in a share cannot be modified or deleted through the File Services interface during the file's retention period, the **file could be overwritten or deleted as a result of S3 activity** that gets synced to the file system. Suppose a WORM protected file is still within its retention period, and its corresponding archived object resides in the Object Lock enabled bucket. Through the S3 interface, a new version of the object could be uploaded to the bucket (while the original, locked object version is retained since versioning is enabled on the bucket); and the new "version" of the object could even be a delete marker. Because the bi-modal access feature automatically keeps the file share in sync with the back-end bucket contents, the file in the share will be updated to match the latest object version in the bucket -- including that the file will be deleted from the share if the latest object version is a delete marker. If you want stricter WORM protection of files in the file share do not enable bi-modal access on the file share. (Note though that disabling bi-modal access is not allowed for [geo-distributed shares](#).)

6.6.3. Set File Share Custom Properties

Path: **Buckets & Objects** → **File Share** → **Properties**



Note This CMC functionality is available only to group admins for whom file services are enabled or regular users for whom file services are enabled. This functionality is not available to system admins.

Subjects covered in this section:

- *Introduction (directly below)*
- ["File Share Properties Customizable Per File Service Pool" \(page 375\)](#)
- ["Snapshot Schedule" \(page 376\)](#)

To configure custom settings for a file share, from the **File Share** tab (**Buckets & Objects** → **File Share**) click **Properties** to the right of the share in your share list. If it's a new share that you just created, you will need to refresh your browser to see the newly created share in your share list.

Note After setting the file share custom properties described below, you can proceed to ["Open a File Share and Provide Its Endpoint To Users" \(page 378\)](#). Note that while most share properties can and should be reviewed and configured prior to opening the file share, configuring a share snapshot schedule is not allowed until after you open the file share.

6.6.3.1. File Share Properties Customizable Per File Service Pool

The screenshot shows a web interface for managing file shares. At the top, there are tabs for 'BUCKETS', 'OBJECTS', and 'FILE SHARE'. Below these are links for 'GLOBAL FILE TENANCY SETTINGS' and '+ ADD NEW SHARE'. A table lists file shares with columns for 'SHARE NAME', 'STATUS', and 'BUCKET'. Two shares are listed: 'pubs-distributed-share' and 'pubs-standard-share'. The 'pubs-standard-share' row is highlighted with a red box and a red circle with the number '1' next to its 'Properties' link. Below the table, there is a section for 'file-pool-1' with a red circle with the number '2' next to its 'Properties' link. This section contains a 'Snapshot Schedule' table, a 'Storage' tab, and a 'Reclaim Delay' section. The 'Storage' tab is active, showing 'Archiving Delay' set to 20 and 'Enable read range' set to 'Enabled'. The 'Reclaim Delay' section shows '0 d 0 h 5 m 0 s'. At the bottom right of the configuration panel are 'CANCEL' and 'SAVE' buttons.

When you're configuring custom settings for a file share, most settings can be set independently for each File Service pool to which you have published the share:

- In the most common case -- a standard file share -- the share is published to just one File Service pool.
- In the case of a [geo-distributed share](#), the share is published to multiple File Service pools and you can configure file share custom properties for each target pool independently.

For each target File Service pool to which the file share is published, the configurable file share properties are presented in several tabs:

- ["Storage, S3, SMB, and NFS"](#) (page 375)
- ["Usage"](#) (page 376)
- ["Bi-Modal Support"](#) (page 376)

Storage, S3, SMB, and NFS

The **Storage, SMB, and NFS tabs** show the same settings that are configurable as [Global File Tenancy Settings](#). Using these share properties tabs is necessary only if you want to configure the share with setting values different than the default values that you established with your Global File Tenancy Settings. Throughout these tabs, the default values that display in the interface will be those that you established when you configured your Global File Tenancy Settings. Any edits that you make in these tabs will override the global default values, for this share only.

The **S3 tab** is for specifying the S3 access key that the File Service should use when archiving data from the file share to the S3 bucket that you've associated with the share. If you have only one [S3 access key](#), that key will automatically populate into the S3 tab. If you have multiple S3 access keys, in the S3 tab you can select one access key from a drop-down list that will display all of your access keys. (Note that the S3 access key cannot be set at a global level for all your shares; it must be set in each share's custom properties.)

After making edits click **Save**.

For setting descriptions see ["File Services Configuration Setting Matrix"](#) (page 384).

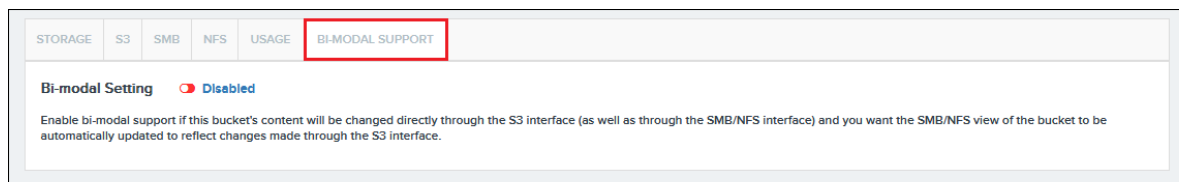
Note If you customize a setting for a file share, and then later you change the Global File Tenancy Setting value for that setting, your share-specific custom configuration will persist -- it will not be overridden by the new Global File Tenancy Setting value.

IMPORTANT ! Although the CMC does not prevent you from having both SMB and NFS enabled for the same file share, the **File Service does not support multi-protocol write access to the same file share** -- and potential conflicts or corruptions may occur if the same file share is concurrently written to by SMB clients and NFS clients. Therefore, if you have both SMB and NFS enabled on the same file share, then you must implement a safe access policy such as SMB write access, NFS read-only.

Usage

Unlike the other properties tabs, the **Usage** tab is for file share usage statistics -- not file share configuration properties. For information about the **Usage** tab see ["View File Share Usage Statistics"](#) (page 382).

Bi-Modal Support



Enabling bi-modal support for a share is appropriate if you expect that S3 client applications -- either the CMC or a third party S3 client application -- will use the S3 interface to make changes to the content of the bucket that backs this file share, and you want those changes to be reflected in the view that SMB or NFS client applications have of the file share. For example, if S3 clients will add objects to the bucket and you want SMB or NFS client applications to be able to list and access those objects (as files in the file share), enable bi-modal support for the share. Similarly, if S3 clients will remove objects from the bucket and you want this to result in SMB or NFS client applications no longer seeing that content in their view of the file share, enable bi-modal support for the share.

If access to the share contents will only be by SMB or NFS client applications -- and S3 clients will not make any changes to the content of the bucket that backs the file share -- then there is no need to enable bi-modal support for the share.

Note that:

- In a **standard file share** -- published to just one File Service pool -- **bi-modal support is disabled by default**. You can enable it if you wish.
- In a [geo-distributed share](#) -- published to multiple File Service pools -- **bi-modal support is enabled by default and cannot be disabled**. Bi-modal support is a prerequisite for geo-distributed shares.

6.6.3.2. Snapshot Schedule

Note

* You cannot create a snapshot schedule until after you have [opened the file share](#).

* For a geo-distributed share, snapshots are configured at the file share level -- they are not configured separately per File Service pool.

A file share "snapshot" is a point-in-time image of a file share. A snapshot freezes and preserves pointers to the object data in your bucket as it exists at a particular moment in time. Leveraging the bucket versioning feature, a file share snapshot gives SMB and NFS client application users the ability to browse the file share as it existed in the past, and download objects as they existed in the past. This can be valuable for file share users in the event that they inadvertently delete a file, or make changes to a file and then later want to retrieve an older version of the file.

Note that a snapshot does not create any additional object data in your bucket -- rather, it is a type of metadata that gives file share users point-in-time access to past versions of objects (which are retained in the bucket as a result of versioning being enabled on the bucket).

Using a file share's **Snapshot Schedule** panel you can configure hourly, daily, and/or monthly schedules according to which the system will automatically create snapshots of the file share.

For each schedule, you specify a maximum number of snapshots for the system to retain as part of that schedule. Once the maximum number of snapshots for that schedule is reached, then each time a new snapshot is created the oldest snapshot from that schedule will be automatically deleted. Each schedule you create appears in a list in the **Snapshot Schedule** tab, and at any time you can edit a schedule or delete it (if you no longer want snapshots to be taken on that schedule).

Once you've created one or more snapshot schedules, and snapshots have been created in accord with the schedule(s), you can view a list of snapshots in the **Snapshots** tab.

NAME	CREATION TIME	STATE
2023-09-28_1500+0000.Hourly	9/28/2023, 11:00:00 AM	CREATED Delete
2023-09-28_1400+0000.Hourly	9/28/2023, 10:00:00 AM	CREATED Delete
2023-09-28_1300+0000.Hourly	9/28/2023, 9:00:00 AM	CREATED Delete
2023-09-28_1200+0000.Hourly	9/28/2023, 8:00:00 AM	CREATED Delete

Each snapshot is named as `<timestamp>.<schedule-name>`, where the latter segment is the name that you assigned to the snapshot schedule that resulted in the creation of the snapshot. Optionally you can delete a snapshot that's listed in the **Snapshots** tab, bearing in mind that if you do so your file share end users will no longer be able to use that snapshot. Keep in mind also that older snapshots will eventually be automatically deleted by the system in accord with your configured maximum number of snapshots to retain for each schedule.

Note that you cannot use the **Snapshot** tab to view the content of a snapshot. This tab is simply for viewing a list of snapshots (and for deleting a snapshot if you wish).

In the case of a **geo-distributed share**, in the unlikely event that the snapshot schedule execution becomes out of sync across the multiple File Service pools to which the share is published, in the **Snapshot Schedule** tab an exclamation point will display beside the schedule name and a **Synchronize Snapshot Schedule** button appears in the upper part of the tab. Click this button to bring the schedule back in sync.

SNAPSHOT

SNAPSHOT SCHEDULE

↻ SYNCHRONIZE SNAPSHOT SCHEDULE

+ ADD NEW SNAPSHOT SCHEDULE

NAME	SCHEDULE	MAX SNAPSHOTS	
test ⚠	hourly	10	Edit Delete

How File Share End Users Can Access Snapshots

Once snapshots have been created for a file share, end users of the file share can access the snapshots in this hidden folder under the file share root:

```
.snapshot
```

Each existing snapshot will be listed under that hidden folder. End users can browse through the contents of a snapshot, and download files through the snapshot if they wish.

Note For information about accessing a hidden folder see your SMB or NFS client application documentation.

6.6.4. Open a File Share and Provide Its Endpoint To Users

Note This CMC functionality is available only to group admins for whom file services are enabled or regular users for whom file services are enabled. This functionality is not available to system admins.

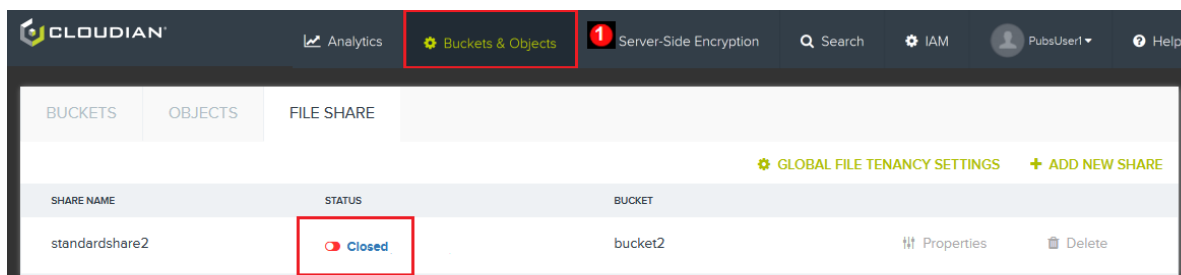
After you've [configured Global File Tenancy Settings](#) for your file shares, and you've [created a file share](#) and (if desired) [customized its settings](#), you can open the file share and then provide the file share's service endpoint to your SMB or NFS client users.

Subjects covered in this section:

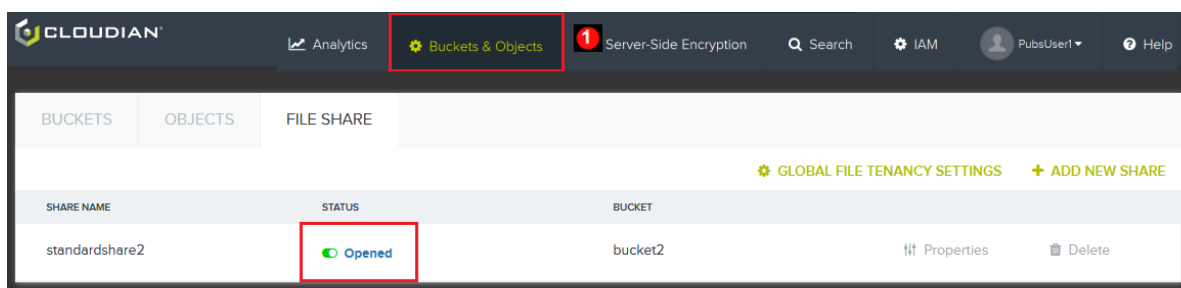
- ["Opening a File Share" \(page 379\)](#)
- ["Determining a File Share's Endpoint" \(page 380\)](#)
- ["Setting an ACL on the Share Root" \(page 381\)](#)
- ["Steps to Take If You Change Your S3 Access Key" \(page 381\)](#)

6.6.4.1. Opening a File Share

When you create a file share, the share is initially in a "closed" state (inaccessible to clients). This closed status is indicated for the file share in the file share list view.



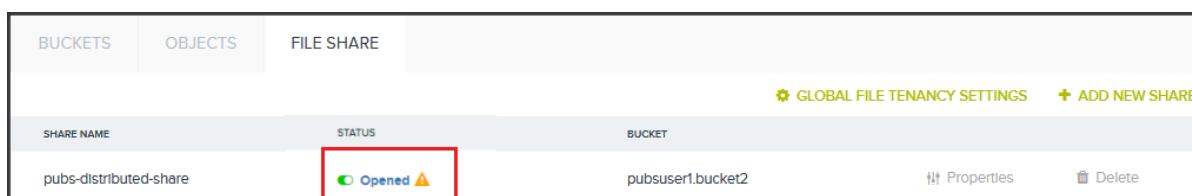
To open the file share, click and drag the slider to the right (or click on the right side of the slider) so that the status shows as Opened.



Note If after a few minutes the Status display is still showing "Opening" rather than "Opened" try refreshing your browser. In the unlikely event that a share appears to be stuck in the "Opening" state (it has been in this state for more than several minutes), you can close the share and then try to open it again.

In the case of a **geo-distributed share**, there will be an open/close slider at the share level (next to the file share name), and then -- below that in the interface -- there will also be open/close sliders for the share in each File Service pool to which the share is published. Although on rare occasions you may want to temporarily close (or open) a share in just one File Service pool, more typically the way to open or close a geo-distributed file share is to do it with the top-level file share controls -- using the open/close slider next to the file share name. This will open or close the file share in all File Service pools to which the share is published.

In the event that a geo-distributed file share is open in one File Service pool and closed in a different File Service pool, in the top-level file share row the Status will show an exclamation mark icon that indicates that the open/closed status for the share is out of sync across the File Service pools to which the share is published.



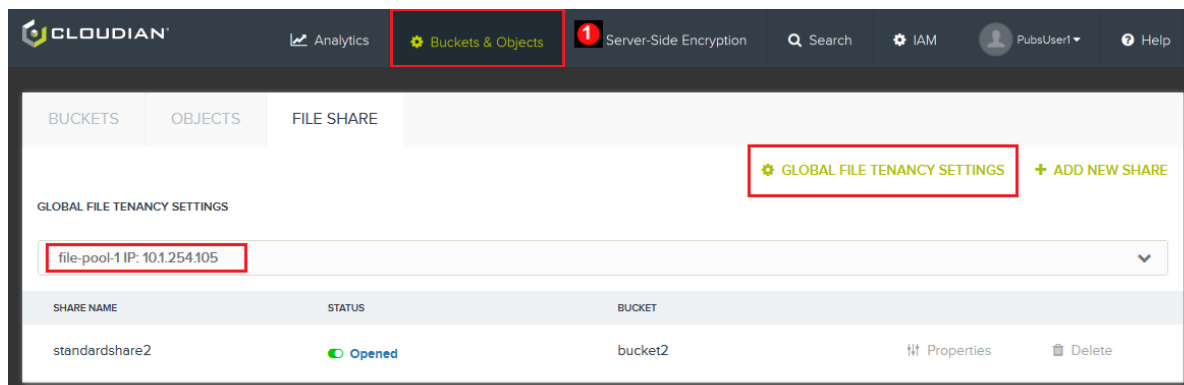
Closing a File Share

If a circumstance arises where you need to close a file share to make it inaccessible to clients, you can use the slider control to change the file share's status to Closed.

For information about opening and closing a geo-distributed file share, see the ["Opening a File Share" \(page 379\)](#) section above.

6.6.4.2. Determining a File Share's Endpoint

Once you have opened at least one file share, an IP address displays next to the File Service pool's name in the **File Share** tab's **Global File Tenancy Settings** section.



This is the front-end sharing IP address for all of your file shares in that File Service pool. This IP is unique to your user account (if any other users of the object storage system are authorized to create and manage file shares, those other object storage system users will have different IP addresses for their file shares). The File Service endpoint for each of your file shares will be a combination of the front-end sharing IP address and the share name, and the format of the endpoint will depend on whether the client access is via SMB or NFS.

For **SMB clients** the access point is:

```
\\<front-end IP address>\<share name>
```

For example:

```
\\10.10.3.48\pubs-bucket1-share
```

NFS clients on Linux can mount the file share with the following command:

```
mount -t nfs <front-end IP address>:<share name> <MOUNT_POINT>
```

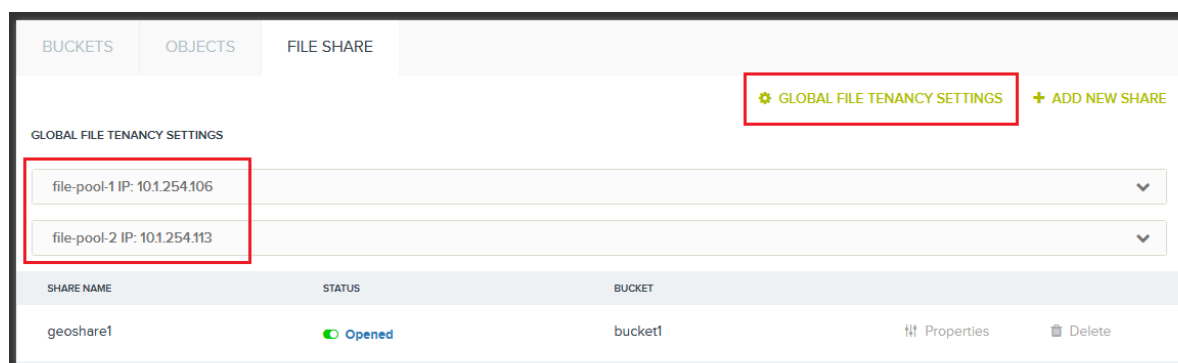
where **<MOUNT_POINT>** is a mount point that has already been created on the client machine.

For example:

```
mount -t nfs 10.10.3.48:/pubs-bucket2-share /mnt/pubs2
```

File Share Endpoints If You've Published File Shares to Multiple File Service Pools

If you have published file shares to more than one File Service pool, there will be an front-end sharing IP address for your shares in each pool.



The formats for SMB and NFS endpoints are the same as described above, except that the clients of different File Service pools will use different front-end IP addresses.

6.6.4.3. Setting an ACL on the Share Root

For a Windows (SMB) share, **before informing your end users of the share endpoint** you should set the ACL that you desire for the share root. To do this, use a Windows client connected to the share. Note that:

- By default when you create a new share the share has the 'Everyone' ACL. Using a Windows client you can set a more restrictive ACL on the share root.
- Because the File Service's share snapshots and shadow copy features create read-only content in each share, when you are setting your desired share root ACL you will encounter a small number of error messages indicating that the ACL cannot be applied to read-only content. You can click through these error messages to continue applying the ACL to the share root.

Because of this consequence of the snapshots and shadow copy features, it's best to set the share root ACL before making the share available to end users. If you try to set a root ACL after the share has been in use, you encounter many more of these "read-only" error messages that you will need to click through (since the snapshots and shadow copy features will have created more read-only content as the share gets used).

Once the share has been in use, you can use a Windows client to set ACLs on specific directories and/or files if you wish.

- Along with setting a root ACL it's advisable to use a Windows client to set a default file ownership for the share. This is particularly true if you have the [bi-modal feature](#) enabled for the share. If you do not set a default file ownership for the share, then all files created via the S3 interface (and made available to SMB clients through the bi-modal feature) will be owned by *root*.

Note The File Service does not support setting an ACL on the share root for NFS shares. For NFS shares you can restrict access by using host (FQDN) or IP address filtering (which you can [set at a global level](#) for all of your NFS shares or [set for a particular NFS share](#)).

6.6.4.4. Steps to Take If You Change Your S3 Access Key

If at any time you [change your S3 access key](#), you must do the following so that the File Service can use your new S3 access key to continue to archive each file share's data to the S3 bucket that you've associated with the share:

1. For each of your file shares, in the S3 tab of the [file share's custom properties configuration](#) update the S3 access key setting to indicate the key that the File Service should use when archiving data to the S3 bucket. The S3 tab will show a drop-down list of all of your S3 access keys.
2. For each of your file shares that is currently open, **close the share and then reopen it**.

6.6.5. View File Share Usage Statistics

Note This CMC functionality is available only to group admins for whom file services are enabled or regular users for whom file services are enabled. This functionality is not available to system admins.

Subjects covered in this section:

- *Viewing File Share Usage Statistics (immediately below)*
- ["Usage Statistics for Geo-Distributed File Shares" \(page 383\)](#)

To view high level usage statistics for a file share click **Properties** to the right of the share in your share list, then click **Properties** for the File Service pool to which the share is published, then select the **Usage** tab.

The screenshot shows the CMC interface with the following elements:

- Share List:** A table with columns: SHARE NAME, STATUS, BUCKET, and actions. It lists 'pubs-standard-share' (Status: Opened, Bucket: pubuser1.bucket1) and 'file-pool-1' (Status: Opened).
- Properties Dialog:** A modal window for 'file-pool-1' is open, showing tabs for STORAGE, S3, SMB, **USAGE** (highlighted with a red box and number 3), and BI-MODAL SUPPORT.
- Usage Statistics:** The USAGE tab displays the following data:

Category	Value
Total files:	6
Total size:	1.55MiB
Total files archived:	6
Total size archived:	1.55MiB
Total files in cache:	6
Total size in cache:	1.55MiB
Total files to archive:	0

In the **Usage** tab you can view the following file share usage statistics:

- "Total files" and "Total size"-- Total number of files in the share, and the total size of those files. These figures are based on files that have been archived to the object storage bucket, plus files that have been ingested by the file service front end and registered in the share database but have not yet been archived.
- "Total files archived" and "Total size archived" -- Total number of files in the object storage bucket that services as the back-end archive storage for this share, and the total size of those files. Note that the counts for total number of archived files and total size of archived files exclude non-current file versions (which are retained in the object storage bucket because versioning must be enabled on the bucket). So for example if in the object storage bucket there is the current version of a given file as well as two older versions of that file, only the current version of the file counts toward the total number of archived files and total archived files size.
- "Total files cache" and "Total size in cache" -- Total number of files in the file service front end cache, and the total size of those files.
- "Total files to archive" -- Total number of files that have been ingested by the file service front end and registered in the share database but have not yet been archived. By typical configuration, files are archived just seconds or minutes after being uploaded to the file share by clients, so these statistics should typically be small in relation to the total number and size of files in the share.

Note Archiving behavior and cache retention behavior are configurable in the **Storage** tab of the [Global File Tenancy Settings](#) and the [file share custom settings](#).

6.6.5.1. Usage Statistics for Geo-Distributed File Shares

In the case of a [geo-distributed share](#) there is a **Usage** tab for each File Service pool to which the file share is published. From the different pools' **Usage** tabs:

- The "Total files archived" and "Total size archived" metrics are for the file share as a whole and so the values reported by the different File Service pools should be very similar -- but not necessarily identical, since the synchronization between File Service pools is rapid but not instantaneous (i.e. the different pools' view of the file share will be "eventually consistent").
- The "Total files" and "Total size" metrics may differ to a limited degree since these metrics include not only archived files but also files that have not yet been archived (and this latter quantity will differ between File Service pools).
- The "Total files in cache", "Total size in cache", and "Total files to cache" metrics may differ substantially between the File Service pools -- for example if one File Service pool is heavily used and the other File Service pool is lightly used, the pools will report very different numbers for cache utilization and for in-progress archiving.

6.6.6. Edit or Delete a File Share

Note This CMC functionality is available only to group admins for whom file services are enabled or regular users for whom file services are enabled. This functionality is not available to system admins.

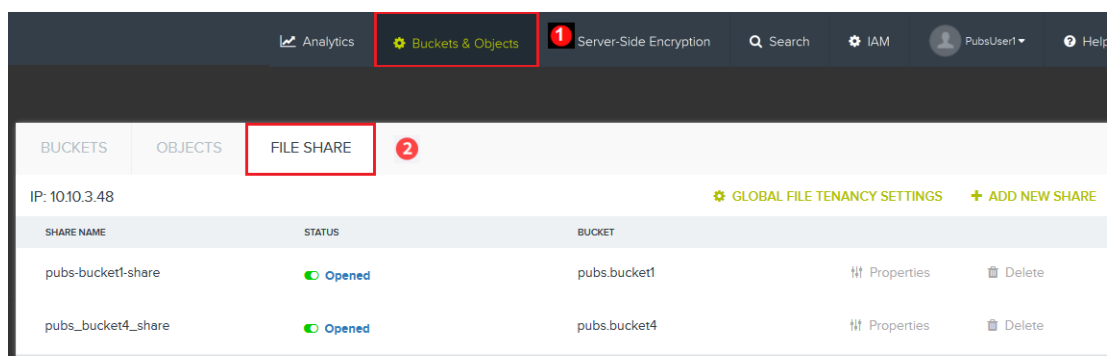
Subjects covered in this section:

- ["Editing a File Share's Settings" \(page 383\)](#)
- ["Deleting a File Share" \(page 384\)](#)

6.6.6.1. Editing a File Share's Settings

To edit a file share's configuration settings:

1. Go to **Buckets & Objects** -> **File Share** to view the list of your existing file shares.



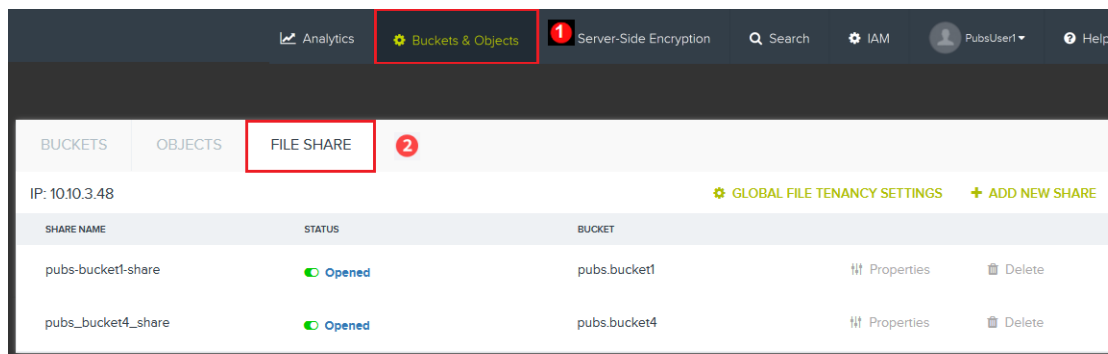
2. In the row for the file share that you want to edit, click **Properties** to display the configuration property tabs for the file share.
3. Edit the share properties as desired. For more information see ["Set File Share Custom Properties" \(page 373\)](#).

6.6.6.2. Deleting a File Share

If you delete a file share, the share's index and metadata will be deleted from the File Service and SMB/NFS clients will no longer be able to access the share. Once you delete a share, **this action cannot be undone**.

To delete a file share:

1. Go to **Buckets & Objects -> File Share** to view the list of your existing file shares.



2. In the row for the file share that you want to delete, close the share if it is not already closed (slide the control bar left to the **Closed** position). You cannot delete an open file share. It may take a few minutes for the share to close.
3. In the row for the closed file share that you want to delete, click **Delete**. You will be prompted to confirm that you want to delete the share. When you confirm, the file share will be deleted.

After you've deleted a file share, data that had been archived from the share will still exist in the share's back-end object storage bucket. If you wish you can then [delete this data](#) from the bucket. If you delete all data from the bucket you can then [delete the bucket](#) if you wish.

In the case of a **geo-distributed share**, to completely delete the share click the **Delete** button in the top-level file share row -- the row in which the file share name appears -- just as described in the steps above. This will delete the file share in all File Service pools to which the share was published. (If instead you click a **Delete** button that's in the row for just one of the File Service pools to which the share is published, that will convert the share to a standard -- not geo-distributed -- share, as described in ["Converting a Geo-Distributed File Share to a Standard File Share" \(page 371\)](#).)

6.6.7. File Services Configuration Setting Matrix

Most file services configuration settings can be set at three different levels of granularity:

Level	Purpose	Editable By	Location in CMC
System	Becomes the default file service settings for all users for whom file services are enabled	System administrators	Cluster -> Cluster Config -> File Ser-

Level	Purpose	Editable By	Location in CMC
			vice Set-tings
User	For a specific user with file services enabled, edits at this level override defaults inherited from the system level to become the default file service settings for that user's file shares.	System administrators on behalf of the user	Manage Users -> retrieve user -> File Set-ting
		User	Buckets & Objects -> File Share -> Global File Ten-ancy Set-tings
File share	For a specific file share, edits at this level override defaults inherited from the system and user levels to become the settings for that file share.	User	Buckets & Objects -> File Share -> Prop-erties for the share.

Note that:

- If you have multiple File Service pools, each File Service pool will have its own independent configurable settings at the system level and at the user level.
- User-level settings can be set by system administrators and also by the users for whom file services are enabled. Setting values set by the users override values set by the system administrator.
- Some settings can only be set at a particular level -- for example Active Directory settings are only at the user level and S3 access credentials are only at the file share level.

The table below shows the settings and the levels at which they can be set. Click on a setting name for a description of that setting.

	Editable by System Administrator		Editable by User with File Services Enabled	
	System Level	User Level	User Level	File Share Level
General Settings				
Waterlevel	✓	X	X	X
Storage Settings				
Read Part Size	✓	✓	✓	X
Archiving delay	✓	✓	✓	✓
Reclaim delay	✓	✓	✓	✓

	Editable by System Administrator		Editable by User with File Services Enabled	
	System Level	User Level	User Level	File Share Level
Enable read range	✓	✓	✓	✓
DNS Settings (overview)				
DNS domain(s)	✓	✓	✓	X
DNS search(es)	✓	✓	✓	X
DNS server IP(s)	✓	✓	✓	X
Active Directory Settings (overview)				
Realm name	X	✓	✓	X
NetBIOS Domain name	X	✓	✓	X
KDC	X	✓	✓	X
Administration server	X	✓	✓	X
Password server	X	✓	✓	X
S3 Settings				
Access-Key	X	X	X	✓
SMB Settings				
Enabled	✓	✓	✓	✓
Create mask	✓	✓	✓	✓
Directory mask	✓	✓	✓	✓
Guest access	✓	✓	✓	✓
VSS mode	✓	✓	✓	✓
Share extra settings	✓	✓	✓	✓
Global extra settings	✓	✓	✓	X
Trusted Domains	X	✓	✓	X
NFS Settings				
Enabled	✓	✓	✓	✓
Access Type	✓	✓	✓	✓
Squash	✓	✓	✓	✓
Protocols	✓	✓	✓	✓

	Editable by System Administrator		Editable by User with File Services Enabled	
	System Level	User Level	User Level	File Share Level
Anonymous Uid	✓	✓	✓	✓
Anonymous Gid	✓	✓	✓	✓
Security Type	✓	✓	✓	✓
Extra settings	✓	✓	✓	✓
Clients	✓	✓	✓	✓
Local User Settings				
Local user groups and users	X	X	✓	X

6.6.8. FileServiceSettings

6.6.8.1. Active Directory Settings Overview

Configure the settings in the **Active Directory** section if you want the HyperStore File Service to implement Active Directory based authentication for the end users of file shares.

After configuring the Active Directory settings listed below, click **Join** to join the HyperStore File Service to your specified Active Directory service. You will then be prompted to provide Active Directory credentials (username and password). These credentials must have join/admin rights so that the join operation can be completed.

IMPORTANT ! If the HyperStore user has already created any file shares, those file shares must be closed when you edit Active Directory settings. After you edit the settings and then Join the Active Directory service, the file shares can be reopened.

Administration server

Active Directory administration server.

KDC

Active Directory key distribution center (KDC).

NetBIOS Domain name

NetBIOS domain name for the domain controller.

Password server

Active Directory Kerberos password server.

Realm name

Active Directory REALM name. Typically this is the Active Directory domain in upper case (e.g. *EXAMPLE.COM*).

6.6.8.2. DNS Settings Overview

In the **DNS** setting section you can provide DNS service information that the HyperStore File Service will use on its "front end", for resolving:

- Active Directory domain names, when implementing Active Directory based authentication of file service end users. (Active Directory domain names can be configured in the **Active Directory** settings section.)
- SMB "trusted domains". (Trusted domains can be configured in the **SMB** settings section.)
- Hostnames of NFS clients, when implementing NFS client connection filtering. (NFS client hostnames can be configured in the **NFS** settings section.)

Configuring these DNS settings is only necessary if the DNS server that should be used for these tasks is different than the DNS server that the HyperStore File Service uses on its "back end" to resolve the S3 endpoint when tiering data to the object storage tier.

DNS domain(s)

The DNS server domain for the HyperStore File Service to use when resolving hostnames and domains while processing requests on its front end (for example, SMB, NFS, and LDAP requests). Must follow standard domain format rules (e.g. *example.com*). To specify multiple domains use a comma-separated list.

DNS search(es)

The DNS server search domain for the HyperStore File Service to use when resolving hostnames and domains while processing requests on its front end (for example, SMB, NFS, and LDAP requests). Must follow standard domain format rules (e.g. *example.com*). To specify multiple search domains use a comma-separated list.

DNS Server IP(s)

The DNS server IP address(es) for the HyperStore File Service to use when resolving hostnames and domains while processing requests on its front end (for example, SMB, NFS, and LDAP requests). To specify multiple IP addresses use a comma-separated list.

6.6.8.3. General

Waterlevel

When the used space in the HyperStore File Service cache reaches this "Waterlevel" percentage of the cache's total storage capacity, the HyperStore File Service will free up cache space by removing files that have reached their "Reclaim delay" interval (configurable in the **File Service Settings** tab's **Storage** section). Files will be removed from cache on an approximate first-in-first-out (FIFO) basis, although the precise ordering is not guaranteed. Files will be removed until capacity usage in the cache falls back below the Waterlevel.

Files removed from the HyperStore File Service cache in this way will continue to exist in the HyperStore object storage tier and their file metadata will continue to exist in the volume database in the HyperStore File Service (so that file services client applications can retrieve such files through the HyperStore File Service).

Note that this Waterlevel mechanism is implemented based on the aggregate cache usage level. It is not implemented separately for each volume (file share).

The maximum allowed setting value is 90 (%).

Default = 60 (%)

Note For background information on the HyperStore File Service cache, see ["File Services Feature Overview"](#) (page 238).

6.6.8.4. LocalUser

Local Users

If you wish you can create local authentication groups and users for your file shares. Local user authentication is one of the methods that the HyperStore File Service supports for authenticating users who try to access file shares. Local users are authenticated against username/password combinations that are stored locally within the HyperStore File Service. Such users can access shares by either SMB or NFS (depending on which of those two protocols you enable for your file shares).

In the **Local Users** section of settings you can create local groups and users, and set passwords for those users.

First select **Add group** to create a local group for local authentication purposes. Assign a name to the group (from 1 to 248 characters, and the only special characters allowed are "-" or "_"). Also you can explicitly assign the group an available GID -- a numerical Group Identifier from the HyperStore File Service's Linux host system -- or else leave the GID field empty and the system will automatically assign the group an available GID. (It's best to let the system automatically assign the GID unless you're instructed differently by your system administrator.)

After adding one or more local groups you can add local users to the group(s). Click **Add user**, then assign a user name (from 1 to 248 characters, and the only special characters allowed are "-" or "_") and a password to the user. Also you can explicitly assign the user an available numerical UID, or else leave the UID field empty and the system will automatically assign the user an available UID. (It's best to let the system automatically assign the UID unless you're instructed differently by your system administrator.) In the Groups field type the name of the group to which to assign the user. The group name is required. Use space separation between group names if you are associating a user with multiple groups.

USER NAME	USER ID	GROUPS	PASSWORD	
User Name	User ID	Groups (space separated)	Password	 Delete

If you wish you can later delete a group or a user by clicking **Delete** in the group or user's configuration line and then clicking **Save** to save the change

6.6.8.5. NFS

Access Type

The default type of share access to grant to NFS clients. The supported options are:

- NONE -- You can use this as the default access type if you want to restrict RW or RO access so that it's granted only to the clients that you explicitly specify in the **Add New Clients** section, and not to any other clients.
- RW -- Read and write.
- RO -- Read-only.

Default = RW

Anonymous GID and UID

The "Anonymous Uid" and "Anonymous Gid" settings are used together with the "Squash" setting. When the user types that you specify with the "Squash" setting access the share, the HyperStore File Service host machines will remap the users' Linux UserID (UID) and GroupID (GID) to the Anonymous Uid and Anonymous Gid that you configure with the "Anonymous Uid" and "Anonymous Gid" settings.

Anonymous Uid default = -2

Anonymous Gid default = -2

Clients

You can add NFS access Clients if you want to:

- Have NFS access be restricted to certain specified clients. To accomplish this you can configure the default NFS "Access Type" setting (above in the **NFS** settings section) to "NONE" and then add clients for which you configure the "Access Type" as "RW" or "RO".
- Have clients whose NFS configuration settings differ from your default NFS configuration settings in other ways.

To add an NFS client click **Add New Client**. This displays configuration fields in which you can identify a client - using a valid client identifying format such as an IP address or an FQDN -- and then specify NFS configuration settings to apply to that client.

The configuration options are the same set of options that you configured for NFS generally (Access Type, Squash, Protocols, and so on), but here they will apply specifically to the client you are adding. For this client, these settings will override your general (default) NFS settings that you set elsewhere in the **NFS** settings section.

After configuring the settings for the client, click **Save** to save the client configuration.

You can click **Add New Client** again if you wish to add more clients.

If you wish you can later delete a client by clicking **Delete** in the client's configuration panel and then clicking **Save** to save the change.

Note An NFS client can only be deleted at the level at which it was created. A client created at the system level can only be deleted at the system level; a client created at the user level can only be deleted at the user level; and a client created at the file share level can only be deleted at the file share level.

Enabled

This setting controls whether or not the HyperStore File Service supports file sharing through the Network File System (NFS) protocol.

IMPORTANT ! Although the CMC does not prevent having both SMB and NFS enabled for the same file share, the **HyperStore File Service does not support multi-protocol write access to the same file share** -- and potential conflicts or corruptions may occur if the same file share is concurrently written to by SMB clients and NFS clients. Therefore, if both SMB and NFS are enabled on the same file share, a safe access policy must be implemented such as SMB write access, NFS read-only.

Note The HyperStore File Service supports NFS sharing over TCP. It does not support NFS over UDP.

Default = Disabled

Extra settings

Here you can optionally specify additional settings (ones not listed in the GUI) to add to the file share-specific `[EXPORT { }]` section(s) of the NFS-Ganesha configuration file *ganesha.conf*, if you wish to use values different than the NFS-Ganesha defaults. Adding extra settings here is not recommended unless you are knowledgeable about NFS-Ganesha configuration. (Ganesha is an open source NFS software used within the HyperStore File Service.) If you do specify extra settings here, specify one setting per line with each line formatted as *key=value* (and pressing Enter/Return to create additional lines as needed). Your settings will be added to the *ganesha.conf* file exactly as you enter them here, so be careful to enter them correctly (including using correct parameter names).

For reference see for example <http://manpages.ubuntu.com/manpages/bionic/man8/ganesha-export-config.8.html> or https://github.com/nfs-ganesha/nfs-ganesha/blob/next/src/config_samples/config.txt.

Protocols

This setting determines which NFS protocol versions the HyperStore File Service will support. Options are:

- 3 -- Support only NFS version 3
- 4 -- Support only NFS version 4
- 3,4 -- Support NFS version 3 and NFS version 4

Default = 3,4

Security type

This setting controls which authentication method or methods the HyperStore File Service will support for NFS clients. Options are:

- None
- Sys
- Krb5
- Krb5i

Default = None, Sys

Squash

When the user types that you specify with the NFS "Squash" setting access a file share, the HyperStore File Service host machines will remap the users' Linux UserID (UID) and GroupID (GID) to the Anonymous Uid and Anonymous Gid that you configure with the NFS "Anonymous Uid" and "Anonymous Gid" settings. Squashing is a security mechanism, to prevent remotely connected root users from having root privileges on the HyperStore File Service hosts.

Supported options are:

- None -- No user id squashing is performed
- Root -- UID 0 and GID of any value are squashed
- Rootid -- UID 0 and GID 0 are squashed
- All -- All users are squashed

Default = Root

6.6.8.6. S3

Access-Key

The access key ID from the S3 access credentials (access key ID and corresponding secret key) that the HyperStore File Service should use to access the object storage bucket for this file share. If you have multiple active access keys then you can select from among those keys in the Access-Key drop-down list.

6.6.8.7. SMB

Create mask

The default permissions to set on files at file creation time. These permissions are applied only to Linux clients.

For example, a permission value of 0744 would grant full access to the file owner and read-only permission for other users.

Default = 0744

Directory mask

The default permissions to set on directories at directory creation time. These permissions are applied only to Linux clients.

For example, a permission value of 0755 would grant full access to the directory owner, and navigation through sub-folders and read-only permission for other users.

Default = 0755

Enabled

This setting controls whether or not the HyperStore File Service supports file sharing through the Server Message Block (SMB) protocol. This is also often referred to as the Common Internet File System (CIFS) protocol, since CIFS is the most prominent dialect of SMB.

When SMB support is enabled, the HyperStore File Service supports SMB versions 2 and 3. SMB version 1 is not supported.

IMPORTANT ! Although the CMC does not prevent having both SMB and NFS enabled for the same file share, the **HyperStore File Service does not support multi-protocol write access to the same file share** -- and potential conflicts or corruptions may occur if the same file share is concurrently written to by SMB clients and NFS clients. Therefore, if both SMB and NFS are enabled on the same file share, a safe access policy must be implemented such as SMB write access, NFS read-only.

Default = Enabled

Global extra settings

Here you can optionally specify settings to add to the *[global]* section of the Samba configuration file *smb.conf*, if you wish to use values different than the Samba defaults. (Samba is an open source SMB software used within the HyperStore File Service.) Adding global extra settings here is not recommended unless you are knowledgeable about Samba configuration. If you do specify global extra settings here, specify one setting per line with each line formatted as *key=value* (and pressing Enter/Return to create additional lines as needed). Your settings will be added to the *smb.conf* file exactly as you enter them here, so be careful to enter them correctly (including using correct parameter names).

For reference see the official [smb.conf documentation](https://www.samba.org/samba/docs/manual/smb.conf.html) online.

Guest access

This option controls whether the HyperStore File Service will allow SMB access to "guest users" (unauthenticated users, also known as anonymous users). If guest access is allowed, then guests will take on the permissions of the "nobody" user account. Across the file system, the "nobody" user will only have the ACL permissions that are associated with the special principal "everyone@" (except where ACL permissions have been explicitly set for the "nobody" user).

Because of the potential security risks of enabling guest access, it's best for this option to be set to Disabled at the system settings level and the Global File Tenancy settings level, so that all file shares inherit the Disabled setting by default. If this option is enabled at all, it should be enabled only in the Share Properties for a specific file share.

Default = Disabled

VSS Mode

This Volume Shadow Copy Service (VSS) Mode setting controls what will display in the "Previous Versions" tab for a file that an end user is viewing in Windows Explorer. The options are:

- previous versions -- Show all previous versions that exist in the bucket
- snapshots -- Show only versions that were captured in a [file share snapshot](#)

Default = snapshots

Note For file share owners, if you change this setting for an existing share your change will not take effect until you close and reopen the share.

Share extra settings

Here you can optionally specify settings to add to the share-specific *[share name]* section(s) of the Samba configuration file *smb.conf*, if you wish to use values different than the Samba defaults. (Samba is an open source SMB software used within the HyperStore File Service.) Adding share extra settings here is not recommended unless you are knowledgeable about Samba configuration. If you do specify share extra settings here, specify one setting per line with each line formatted as *key=value* (and pressing Enter/Return to create additional lines as needed). Your settings will be added to the *smb.conf* file exactly as you enter them here, so be careful to enter them correctly (including using correct parameter names).

For reference see the official [smb.conf documentation](#) online.

Trusted Domains

At the bottom of the **SMB** settings tab you can add one or more trusted domains to your SMB configuration if you wish. This will apply to all file shares that you create.

For the Trusted Domain setting click **Edit** and then click **Add domain** to display the trusted domain configuration dialog:

In the left-most field enter the trusted domain name (must follow standard domain format rules, such as *example.com*).

Then from the drop-down list choose either [ads](#) or [rid](#) as the ID mapping backend for the trusted domain.

Then enter the ID range (minimum ID value and maximum ID value) for the trusted domain. The valid ID space is from 0 to 999999.

Note If you intend to add multiple trusted domains, the ID ranges from the multiple domains must not overlap. The total valid ID space (within which the per-domain ranges must fall) is from 0 to 999999.

Then click **Save** to complete the addition of the trusted domain.

If you subsequently want to delete a trusted domain that you've added, click **Delete** to the right of the trust domain configuration and then click **Save**.

6.6.8.8. Storage

Archiving Delay

When a client writes a file to the HyperStore File Service, the HyperStore File Service will wait for **at least** this long after the last modification time before starting to archive the file to the object storage tier (before starting to copy the file from the HyperStore File Service cache to the applicable object storage bucket). The actual archiving timing will depend on this 'archiving delay' setting and other factors that can contribute to longer delays, such as the file size and the current archiving queue size.

Set this as a number of seconds -- for example, if you set it to 60 this means 60 seconds. The minimum allowed value is 5 (seconds).

Default = 20 (twenty seconds)

Note For a WORM file share (backed by a bucket with Object Lock enabled) the HyperStore File Service will wait for at least 40 seconds after a file's last modification time before starting to archive the file to the object storage tier, regardless of the Archiving Delay setting.

Enable Read Range

When this setting is enabled the HyperStore File Service supports doing ranged reads of objects from the object storage tier, whereby only a portion of the object's data is downloaded from the object storage bucket into the HyperStore File Service cache.

If HyperStore File Service client applications will sometimes request to retrieve just a portion of a file -- a "ranged read" -- keep this setting enabled.

If HyperStore File Service client applications will always request entire files -- and never request to retrieve just a portion of a file -- then disabling this setting is recommended. Disabling this setting optimizes HyperStore File Service performance when downloading entire objects from object storage buckets.

Default = enabled

Read Part Size

Note This setting is applicable only when a client's requested file data has been removed from the HyperStore File Service cache by the cache space reclamation process and must be retrieved from the object storage tier.

When reading an object from the object storage bucket back into the HyperStore File Service cache, the HyperStore File Service will download the object in parts of this size (in MiBs). For example, with the default Read Part Size of 100MiB, a 450MiB object would be downloaded into the HyperStore File Service cache as four 100MiB parts plus a 50MiB part.

In the case of ranged reads, the HyperStore File Service will download from HyperStore one or more parts of no less than this size, to encompass the client's requested range. For example, with the default Read Part Size of 100MiB, if the client's requested range spans 70MiB of a larger object's data, the HyperStore File Service will download from HyperStore just one 100MiB part that includes the requested 70MiB data range. If the client's requested range spans 240MiB of an object's data, the HyperStore File Service will download from HyperStore three 100MiB parts (300MiB total) that include the requested 240MiB data range.

The minimum allowed value for this setting is 10 (MiBs) and the value must be a multiple of 10.

Default = 100 (MiBs)

Reclaim Delay

When a file is successfully copied from the HyperStore File Service cache to the object storage bucket associated with the file share, this starts the "Reclaim delay" timer for the file. During the "Reclaim delay" period, the file will exist in the HyperStore File Service cache as well as in the object storage bucket. After the "Reclaim delay" period completes, the file is eligible to be automatically removed from the HyperStore File Service cache by the system if the cache storage consumption reaches the configurable ["General" \(page 388\)](#). If a file is removed from the HyperStore File Service cache as part of this cache space reclamation process, the file will continue to exist in the object storage bucket and can be retrieved from there by the HyperStore File Service if a file services client requests the file.

Set this as a number of days (d), hours (h), minutes (m), and/or seconds (s).

Reclaim delay d h m s

The minimum allowed value for this setting is 5m (or 300s).

Default = 5m (five minutes)

6.7. Search

Path: **Search**

Note This functionality displays only if you are logged in to the CMC as a group administrator or a regular user (and only if you have set up the [Metadata Search feature](#) in your system). This functionality does not display in the CMC for system admins.

In the **Search** page you can search for your objects by a variety of object metadata attributes.

The console's **Search** page is available for you to use only if both of these conditions apply:

- Service administrators have enabled the Search Service in your object storage system. If the Search Service is not enabled in your object storage system, the **Search** tab will not display in the console menu bar.
- One or more of your buckets use a storage policy that supports metadata search. You can only use the **Search** page for searching in a bucket that uses a storage policy that supports metadata search. The **Search** page will return an error if none of your buckets use a storage policy that supports metadata search.

If the **Search** page is not available for you to use because one of the conditions above is not met, you still have the ability to [Search By Prefix](#), in the console's **Objects** page.

Note also that:

- The Search Service indexes objects in search-enabled buckets hourly. Consequently, when you upload new objects to a bucket, it may be up to an hour before the objects become searchable.
- Search is not supported for buckets that have bucket names exceeding 50 characters.
- Search is not supported for objects that have object names (including full path) exceeding 200 characters.

To use the **Search** page:

1. Enter the bucket name in the "Bucket" field. As you type a drop-down list of your search-supporting buckets will display and you can select from the list. (Selecting a bucket is mandatory -- you can only search within a bucket, not across multiple buckets.)
2. Click in the "Search Query" text field. This opens the **Search Query** editor, which by default displays a **Visual Editor** tab in which you can enter your desired search criteria.

Note Power users may wish to directly construct a search query in JSON format rather than using the **Visual Editor**. To do so, select the **JSON** tab and construct a query consistent with the [OpenSearch Query DSL](#).

3. In the **Visual Editor** tab you can specify a search query composed of one or more of the following criteria:
 - **Object Name** -- You can enter an exact object name to find just that one object, or a text string to find objects that have names that include the string. Application of this search criteria is not case-sensitive. If you are familiar with regular expression syntax and want to specify a regular expression here, you can click the "Regular expression" checkbox and then enter the expression.

If you want to retrieve a list of all objects that have a certain file extension, leave the first Object Name field segment empty and complete just the second segment (with a file extension such as *pdf* or *docx* for example).

- **Size** -- To find objects at or above a certain size, complete just the "From" field (as a number of bytes) and leave the "To" field empty. To find objects at or below a certain size, do the reverse. To find objects within a size range, complete both fields.
- **Date** -- To find objects created on or after a certain date, complete just the "From" field and leave the "To" field empty (or set it today's date). To find objects created on or before a certain date, do the reverse. To find objects created within a date range, complete both fields.
- **User Metadata** -- If the S3 client application that you used to upload objects to the bucket allows you to create user-defined object metadata as you upload objects, and you created such metadata for all or some of the objects you uploaded, you can use that metadata as criteria in your search query. The client application in uploading the object would have specified your metadata in the form of upload request headers named as *x-amz-meta-**<key>**: <value>* -- for example *x-amz-meta-subject: cats* or *x-amz-meta-importance: high*. In the **Search** page, after clicking **Add User Metadata Filter** to display a User Metadata "Key" field and "Value" field, enter the metadata key and value that you want to search by. Do not include the *x-amz-meta-* prefix. For example:

You must specify the "Key" in lower case (so for example a key of "City" or "Importance" will not return any search results; instead enter the key as "city" or "importance"). By contrast the "Value" field is not case-sensitive (so for example entering "Nyc" in the "Value" field will yield the same search results as entering "NYC" or "nyc" in this field).

You must specify a "Value". The system does not currently support leaving the "Value" field empty (to search for all objects that have a specified "Key", regardless of its value).

If you wish you can click **Add User Metadata Filter** again and specify additional user metadata key-value pairs.

Note The Console does not support the **creation** of user-defined object metadata, so searching by User Metadata is only applicable if you used a different S3 application to upload the objects and that application supports creating user-defined object metadata.

Note that:

- All of the search criteria that you specify are **additive**. If you specify multiple criteria -- for example a size range and a date range and a user metadata key-value pair -- the search will return only objects that meet **all** of the specified criteria.
 - Searching by object "tags" (which some S3 client applications allow you to apply to objects that are already in storage) is not supported.
 - If your query strings for object name or user metadata value include any special characters other than periods or forward slashes the special characters are ignored and surrounding text strings are treated as separate strings when the query is executed. For example a search for user metadata value "satoshi@gmail.com" would match against records that include "satoshi" or "gmail.com" and could return all of "satoshi@yahoo.com", "satoshi@gmail.com", and "hiro@gmail.com". (Therefore, if email addresses in form "<user>@<domain>" are being used as user metadata and all addresses use the same "@<domain>" string, specify just the unique "<user>" string in the search query.)
 - Search query text strings longer than 255 characters are truncated to 255 characters.
4. When you've finished entering your search query criteria click **Save** to save the query (which will then display in the "Search Query" text box in JSON format, even if you used the Visual Editor).
 5. Then click **Search** to execute the query.

This returns a list of objects that match your search criteria, and for each such object the object's size, version ID (if the bucket has versioning enabled), creation date, and user-defined metadata (if any) are shown.

Hyper Search

Bucket

Search Query

```
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
```

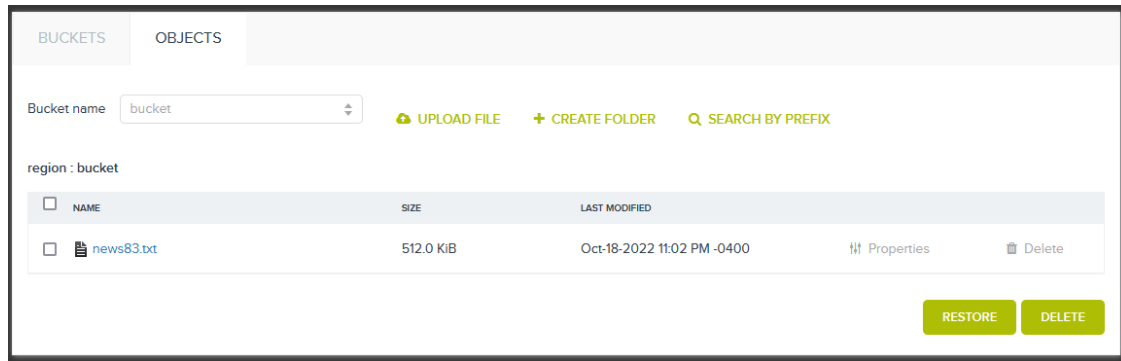
Show entries Search:

Object Name	Size	Version ID	Date	User Metadata
news83.txt	512.0 KiB		2022-10-19T03:02:39.609Z	{"len":"15","city":"SF","cat":"cat3","yr":"2021","media":"document","prog":"Snap-Judgement"}
asset1.txt	512.0 KiB		2022-10-19T03:01:35.829Z	{"len":"44","city":"SF","cat":"cat2","yr":"2021","media":"document","prog":"Wait-Wait-Dont-Tell-me"}

Showing 1 to 2 of 2 entries Previous Next

In the search results list you can:

- Sort the list by any of the columns by clicking the up-down arrows icon to the right of the column heading.
- Further filter the results list by entering a text string in the "Search" box at the top right of the list.
- Click on an object name to open a separate browser tab in which that object appears in the console's **Objects** page, and from there you can download or delete the object, or edit the object's permissions.



To edit your search query, click in the "Search Query" text box again, make your edits, click **Save**, then click **Search**.

To run an entirely new search query, click **Clear**, then click in the "Search Query" text box and enter, save, and execute the search.

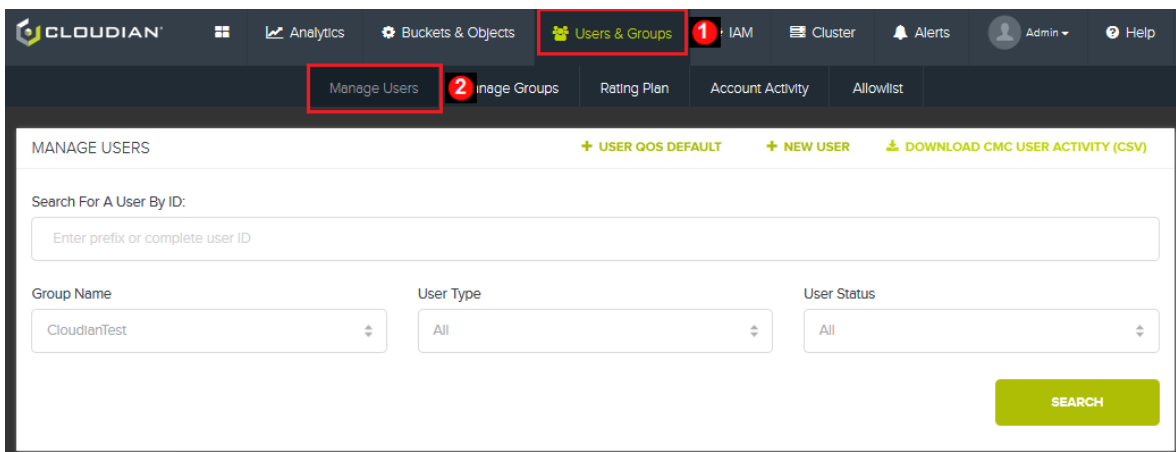
6.8. Users & Groups

The **Users & Groups** tab contains the following functions:

- [Manage Users](#)
- [Manage Groups](#)
- [Rating Plan](#)
- [Account Activity](#)
- [Allowlist](#)

6.8.1. Manage Users

Path: **Users & Groups** → **Manage Users**



Supported tasks:

- ["Add a User"](#) (page 401)
- Work with existing users:
 - ["Retrieve a User or List of Users"](#) (page 404)
 - ["Manage a User's Stored Objects"](#) (page 405)
 - ["Configure a User's File Services Default Settings"](#) (page 406)
 - ["Edit or Suspend a User"](#) (page 407)
 - ["View or Edit a User's Security Credentials"](#) (page 410)
 - ["Unlock a User Who's Locked Out of the CMC"](#) (page 411)
 - ["Disable MFA for a User"](#) (page 412)
 - ["Delete a User"](#) (page 413)
- ["Download User Audit Log"](#) (page 414)
- ["Set Quality of Service \(QoS\) Limits"](#) (page 430)

6.8.1.1. Add a User

IMPORTANT ! For a user who belongs to a group that is enabled for LDAP-based authentication:

* If you want the user to be authenticated against the LDAP system when he or she logs in to the CMC , do not create the user here. Instead, just have the user log into the CMC with their LDAP credentials and upon that first log-in the CMC will automatically create the user in the HyperStore system (and subsequently you will be able to [edit the user's profile](#) if you wish). Note that the user names of LDAP-authenticated users must meet the character restrictions described under "User ID" below. For example, spaces are not allowed in user names.

* If you want the user to be authenticated by a CMC -based password rather than authenticating against the LDAP system, create the user here. The CMC will not use LDAP-based authentication for users created through the **Add New User** interface described below.

To add a user:

1. In the [Manage Users](#) page, click **New User**. This opens the **Add New User** panel.

2. In the **Add New User** panel, complete the following user information:

User ID (mandatory)

- Must be unique within the group.
- Only letters, numbers, dashes, underscores, dots, and ampersands (@) are allowed.

- By default the maximum length is 64 characters. This maximum is controlled by the system configuration setting "[common.user.name.maxLength](#)" (page 755).
- The following user IDs are reserved for system use and are not available to individual users: "anonymous", "public", "null", "none", "admin", "0".

Note The character rules for the user IDs of system administrators are more strict:

- * Maximum length = 26 characters (this is not configurable)
- * Only lower case letters, numbers, and underscores are allowed
- * Cannot end with an underscore
- * Must start with a letter (This restriction applies only to system admin users who are created in the CMC **Add New User** panel. It does not apply to system admin users whose logins to the CMC will be authenticated via an LDAP system. Note however that if an LDAP-based system admin user has an all-numeric user ID, that user will be able to log in to the CMC, but a corresponding [HyperStore Shell](#) user -- for logging in to HyperStore host machines -- will **not** be created. This is due to user ID restrictions in Rocky/RHEL 8 host systems.)

User Type (mandatory)

From the drop-down list select the type of user that you want to create:

- **User** -- A regular user of the storage service. He or she will be able to use the CMC to create storage buckets, upload and download objects, display reports on their service usage, and manage their service access credentials.
- **Group Admin** -- An administrator of a particular group of storage service users. He or she will be able to use the CMC to perform administrative functions for the group, such as adding users or setting user-level QoS profiles. A group admin has a storage service account and can upload their own objects to storage. A group admin can also manage stored objects on behalf of particular users within the group.
- **System Admin** -- A system administrator. He or she will be able to use the CMC to perform a variety of system administration and service administration tasks. System admins cannot use the CMC to create their own S3 storage buckets or upload/download objects. However:
 - System admins can manage buckets and objects on behalf of regular service users, [if your system is configured to allow this](#).
 - System admins other than the default system admin (the pre-configured "admin" user) can use third party S3 applications to create their own S3 buckets and upload/download objects.

Note If you create a System Admin user the system automatically creates a HyperStore Shell account for that user. For more information see "[Enabling the HSH and Managing HSH Users](#)" (page 106).

Group Name (mandatory)

From the drop-down list choose the group in which the new user will be created.

This field is hidden and not relevant if User Type is System Admin. All system admin users belong to the System Admin group.

Password / Confirm Password (mandatory)

The user's password for logging into the CMC. For system admin users this will also be their password

for logging into the [HyperStore Shell](#).

Passwords must meet the following conditions by default:

- Minimum of nine characters, maximum of 64 characters
- Must contain:
 - At least one lower case letter
 - At least one upper case letter
 - At least one number
 - At least one special character such as !, @, #, \$, %, ^, etc.

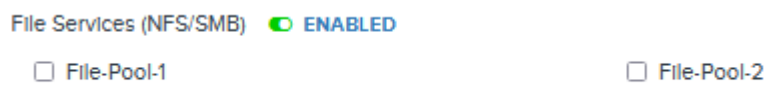
Note You can optionally configure HyperStore to require a higher minimum password length. You can also optionally configure additional password restrictions such as a password expiration period, a restriction against a user's new password being too similar to their previous password, a restriction on password reuse, and a restriction against too-frequent password changes. In the [common configuration setting group](#), these restrictions are configurable by the `common.user.password.*` settings.

File Services

If you've set up the HyperStore File Service as part of your HyperStore system, you can optionally enable File Services functionality for the user. By default File Services functionality is disabled for new users that you create. (If your HyperStore system does not include the HyperStore File Service, you cannot enable File Services functionality for your users -- this option will be grayed out in the **Add New User** interface.)

If you enable File Services functionality for a user, that user will be able to use the CMC to create and configure SMB and/or NFS file shares that will use the user's buckets for back-end storage. **You should only grant this capability to users who have experience with configuring and managing SMB or NFS file shares.**

If you've set up multiple HyperStore File Service pools, when enabling File Services for a user you can select the checkboxes to choose the File Service pools to which the user will be able to publish file shares.



If you give a user access to multiple File Service pools then the user will be able to publish standard file shares to any one of those pools, and will also be able to publish geo-distributed files shares that span multiple File Service pools. Be sure that the user knows the location of each File Service pool, and knows in general which SMB/NFS end users will be using each pool.

For more information on HyperStore's support for File Services functionality, see ["File Services Feature Overview"](#) (page 238) and ["Preparing the File Services Feature"](#) (page 241).

Note File Services cannot be enabled for System Admin users. File Services functionality is allowed only for users who have S3 access, and System Admin accounts do not have S3 access.

More (optional; includes rating plan assignment)

When you click "More" these additional optional fields display:

* Full Name (maximum 64 characters by default. This maximum is controlled by the configurable setting "**common.user.name.maxLength**" (page 755).)

* Address and contact information

Rating Plan

Rating plan to assign the user for billing calculation purposes. A new user's rating plan assignment defaults to whatever rating plan you've assigned to the group to which the user belongs. If you assign a user a different rating plan than the plan assigned to his or her group, the individual plan assignment will be applied to that user rather than the group default plan.

If you don't choose a rating plan for the user, the user is automatically assigned the default rating plan for the user's group.

If your HyperStore system has multiple service regions, rating plan assignment is on a per-region basis. A "Region" drop-down list displays which includes each of your region names. For each region, assign the user a rating plan that will apply to the user's service use within that region.

3. Click **Save**.

6.8.1.2. Retrieve a User or List of Users

In the [Manage Users](#) page you can retrieve a single user or a filtered list of users.

1. Select or enter your user list filtering criteria. You can filter by:

Search for User By ID

Specify a user ID prefix to retrieve a list of users whose IDs start with that prefix.

Specify a complete user ID if you want to retrieve just that one user.

This field is case-sensitive.

Leave this field blank if you want to filter exclusively by the other criteria.

Group Name

This is a drop-down list of group names in the system.

Note If you have more than 100 groups in your system this is implemented as a text input field with auto-complete (rather than a drop-down list).

User Type

- User — Regular service users.
- Group Admin — Group admin users.

- All — Regular service users, group admin users, and system admin users.

User Status

- Active — Users who are currently allowed service access.
- Inactive — Users whose service access credentials have been deactivated but who have not been deleted from the system. They may still have objects in storage.
- All — Active users and inactive users.

Note The CMC does not support retrieving a list of users who have been deleted from the system. If you need to retrieve a list of deleted users, you can do so through the Admin API -- see [GET /user/list](#).

2. Click **Search**. Your filtered search results then display in the lower part of the page.

USER ID	GROUP NAME	USER TYPE	STATUS	ACTIONS
PubsGroupAdmin	Pubs	GroupAdmin	Active	Edit Credentials Set QoS View User Data Delete
PubsUser1	Pubs	User	Active	Edit Credentials Set QoS View User Data Delete
PubsUser2	Pubs	User	Active	Edit Credentials Set QoS View User Data Delete

You can then take any of the following actions regarding the retrieved user(s):

- ["Edit or Suspend a User"](#) (page 407)
- ["View or Edit a User's Security Credentials"](#) (page 410)
- ["Set Quality of Service \(QoS\) Limits"](#) (page 430) for a user
- ["Manage a User's Stored Objects"](#) (page 405) (if your system is configured to allow you to do this)
- ["Configure a User's File Services Default Settings"](#) (page 406) (if your HyperStore system includes the HyperFile Service and you've enabled file services functionality for one or more of your users)
- ["Unlock a User Who's Locked Out of the CMC"](#) (page 411)
- ["Disable MFA for a User"](#) (page 412)
- ["Delete a User"](#) (page 413)

6.8.1.3. Manage a User's Stored Objects

The CMC supports allowing administrators to access and manage data stored by regular users. By default this capability is disabled. To enable this capability set the system configuration setting ["cmc.ui.admin.manageUsers.view.objectDataGrantees"](#) (page 739) to *SystemAdmin, GroupAdmin* (to allow this capability to system administrators and also group administrators [in regard to their own group members]) or set it to *SystemAdmin* (to allow this capability only to system administrators). You can then access users' data via the CMC as follows described below.

1. Use the [Manage Users](#) page to ["Retrieve a User or List of Users"](#) (page 404).
2. In the "Actions" column for the user click **View User Data**. This opens the **Buckets & Objects** page for that user.

You can then work with the user's buckets and objects:

- ["Add a Bucket"](#) (page 307)
- ["Set Bucket Properties"](#) (page 309)
- ["Delete a Bucket"](#) (page 343)
- ["Create or Delete a "Folder""](#) (page 345)
- ["Upload an Object"](#) (page 346)
- ["Set Object Properties"](#) (page 348)
- ["List or Search for Objects"](#) (page 357)
- ["Download an Object"](#) (page 359)
- ["Delete an Object"](#) (page 363)
- ["Retrieve a Tiered Object"](#) (page 360)

Note The **Buckets & Objects** page will continue to be populated with a view of this user's data throughout your CMC login session, unless you use the **Manage Users** page to switch to viewing a different user's data.

6.8.1.4. Configure a User's File Services Default Settings

Note For background information about the HyperStore file services feature see ["File Services Feature Overview"](#) (page 238) and ["Preparing the File Services Feature"](#) (page 241).

If your HyperStore system includes the HyperStore File Service, and if you have enabled File Services functionality for one or more users, then when you [list users](#) in the **Manage Users** page there will be a **File Setting** option for the users for whom you've enabled File Services:

MANAGE USERS ➕ USER QoS DEFAULT ➕ NEW USER 📄 DOWNLOAD CMC USER ACTIVITY (CSV)

Search For A User By ID:

Group Name: User Type: User Status:

SEARCH

USER ID	GROUP NAME	USER TYPE	STATUS	ACTIONS
PubsGroupAdmin	Pubs	GroupAdmin	Active	Edit Credentials Set QoS Delete
PubsUser1	Pubs	User	Active	Edit Credentials Set QoS File Setting Delete

When you click the **File Setting** option, this opens a panel where you can configure File Services default settings for a specific HyperStore user for whom you have enabled File Services functionality. If you have multiple File Service pools in your HyperStore system you can configure these user-level default settings independently for each File Service pool.

USER ID	GROUP NAME	USER TYPE	STATUS	ACTIONS
PubsGroupAdmin	Pubs	GroupAdmin	Active	Edit Credentials Set QoS Delete
PubsUser1	Pubs	User	Active	Edit Credentials Set QoS File Setting Delete

file-pool-1

Storage

DNS

Active Directory

SMB

NFS

file-pool-2

Note that:

- These are mostly the same settings for which you set system-wide defaults in the **Cluster -> Cluster Config -> File Service Settings** tab. The setting values that you set as the defaults for a specific user will override the system-wide defaults, just for that user. These user-specific default settings also include some settings that you can only configure here -- at the user-specific level -- and not at the system-wide default level: the Active Directory settings, and the ability to create SMB Trusted Domains.
- The individual user for whom you've enabled file services has the ability -- when they are logged into the CMC -- to set default configuration settings for their file shares and to customize settings for specific file shares. **Setting values that the user configures will take precedence over the default values that you establish for the user.**

For setting descriptions see "[File Services Configuration Setting Matrix](#)" (page 384). This setting description matrix also indicates the level(s) at which each setting can be edited (system-wide defaults set by system admin, user-specific defaults set by system admin, user-specific global settings set by user, and/or share-specific settings set by user).

IMPORTANT ! Although the CMC does not prevent having both SMB and NFS enabled for the same file share, the **HyperStore File Service does not support multi-protocol write access to the same file share** -- and potential conflicts or corruptions may occur if the same file share is concurrently written to by SMB clients and NFS clients. Therefore, if both SMB and NFS are enabled on the same file share, a safe access policy must be implemented such as SMB write access, NFS read-only.

6.8.1.5. Edit or Suspend a User

1. Use the [Manage Users](#) page to "[Retrieve a User or List of Users](#)" (page 404).
2. In the "Actions" column for the user that you want to edit, click **Edit**. This drops down a panel for editing the user's attributes.

3. Make your desired changes to the user's attributes:

Active User

The "Active User" checkbox is in the upper right corner of the Edit User panel.

- Check the Active User checkbox — This makes the user Active. The user will have access to the storage service and the CMC .
- Uncheck the Active User checkbox — This suspends the user (makes the use status Inactive). The user will be denied access to the storage service and the CMC . However, the user will remain in the system and their objects will remain in storage.

Note If you want to remove a user entirely and have the system delete the user's stored objects, [delete the user](#) rather than just suspending the user.

User Type

- **User** -- Regular user of the storage service. He or she will be able to use the CMC . to create storage buckets, upload and download objects, display reports on their service usage, and manage their service access credentials.
- **Group Admin** -- An administrator of a particular group of storage service users. He or she will be able to use the CMC to perform administrative functions for the group, such as adding users or setting user-level QoS profiles. A group admin has a storage service account and can upload their own objects to storage. A group admin can also manage stored objects on behalf of particular users within the group.
- **System Admin** -- System administrator. He or she will be able to use the CMC to perform a variety of system administration and service administration tasks. System admins do not have a storage service account and cannot upload their own objects to storage. However, system admins can manage stored objects on behalf of particular service users, [if your system is configured to allow this](#).

Note that:

- You cannot change a regular user's or group administrator's User Type to System Admin.
- You cannot change a system administrator's User Type to anything other than System Admin
- If you change a group administrator's User Type to regular user, and that group administrator is in a CMC session at the time that you make the change, that group administrator will not be able

to access the **Manage Users** page. If they try to access the **Manage Users** page the CMC will end the session and when the user logs back in again they will have only regular user privileges, not group administrator privileges.

Canonical ID

The system automatically generates a unique canonical ID for every user. This ID cannot be edited.

Rating Plan

Use the "Rating Plan" drop-down list to assign the user a rating plan for billing calculation purposes. If you assign a user a different rating plan than the plan assigned to his or her group, the individual plan assignment will be applied to that user rather than the group default plan.

Since system administrator user IDs do not have their own S3 service accounts, the "Rating Plan" drop-down list does not display if you are editing a system admin user.

Note for multi-region systems

If your HyperStore system has multiple service regions, rating plan assignment is on a per-region basis. A "Region" drop-down list displays which includes each of your region names. For each region, assign the user a rating plan that will apply to the user's service use within that region, clicking Save each time.

File Services

If you've set up the HyperStore File Service as part of your HyperStore system, you can optionally enable File Services functionality for the user. By default File Services functionality is disabled for new users that you create. (If your HyperStore system does not include the HyperStore File Service, you cannot enable File Services functionality for your users -- this option will be grayed out in the **Edit User** interface.)

If you enable File Services functionality for a user, that user will be able to use the CMC to create and configure SMB and/or NFS file shares that will use the user's buckets for back-end storage. **You should only grant this capability to users who have experience with configuring and managing SMB or NFS file shares.** For more information on HyperStore's support for

If you've set up multiple HyperStore File Service pools, when enabling File Services for a user you can select the checkboxes to choose the File Service pools to which the user will be able to publish file shares. If you give a user access to multiple File Service pools then the user will be able to publish standard file shares to any one of those pools, and will also be able to publish geo-distributed files shares that span multiple File Service pools. Be sure that the user knows the location of each File Service pool, and knows in general which SMB/NFS end users will be using each pool.

For more information on HyperStore's support for File Services functionality, see ["File Services Feature Overview" \(page 238\)](#) and ["Preparing the File Services Feature" \(page 241\)](#).

If you had previously enabled file services for a user, and now you want to disable file services for that user, you can do so by changing the user's File Services setting to disabled. Note however that if the user **still has file shares**, the user will need to delete those file shares before you can disable file services for the user. The CMC will not allow you to disable file services for a user who still has file shares. The same holds true if you want to remove a user's current access to a particular File Service pool -- you cannot remove this access if the user currently has a file share published to that File Service pool. The user will need to delete the file share before you can end the user's access to that File Service pool.

In a scenario where an uncooperative user will not delete their file shares, contact Clouddian Support for information about how you can delete the shares on your own. (As system administrator you cannot

delete a user's file shares through the CMC, but Cloudian Support can guide you through "back door" methods for doing so.)

Note File Services cannot be enabled for System Admin users. File Services functionality is allowed only for users who have S3 access, and System Admin accounts do not have S3 access.

Full name and contact information (click '**More**' to display these fields)

User's full name (maximum length is 64 characters by default; this maximum is configurable by the setting "**common.user.name.maxLength**" (page 755).) and contact information.

Note If the user is configured to be authenticated against an external LDAP system rather than by a CMC -based password, the edit user interface will display some text indicating that the user is LDAP-enabled. This is not editable. Whether or not a user is LDAP-enabled is determined at the time that the user is created and this cannot be changed subsequently.

4. Click **Save**.

For information about changing a user's password see "**View or Edit a User's Security Credentials**" (page 410).

6.8.1.6. View or Edit a User's Security Credentials

1. Use the [Manage Users](#) page to "**Retrieve a User or List of Users**" (page 404).
2. In the "Actions" column for the user whose security credentials you want to view or edit, click **Credentials**. This displays a panel for changing the user's CMC sign-in password and viewing and managing the user's S3 access credentials.

User Credentials

SIGN-IN CREDENTIALS

USER ID: PubsUser1 **GROUP ID:** Pubs

NEW PASSWORD: **CONFIRM PASSWORD:**

CHANGE PASSWORD

S3 ACCESS CREDENTIALS

CREATED	ACCESS KEY ID	ACTIONS
Apr 08 2018 07:49:36 GMT-0400	6ed8c97da762ec742d88 * * Active Access Key	View Secret Key Inactivate Delete

CREATE NEW KEY

Close

Note If the user is an LDAP-authenticated user, the "Sign-In Credentials" section of the **Security Credentials** page does not display. Instead you should manage the user's password in your LDAP system.

3. Take the desired action:

- **To set a new CMC sign-in password** for the user, enter the password in the "New Password" field and in the "Confirm Password" field, then click **Change Password**. For system admin users this will also be their password for logging into the [HyperStore Shell](#).

Passwords must meet the following conditions by default:

- Minimum of nine characters, maximum of 64 characters
- Must contain:
 - At least one lower case letter
 - At least one upper case letter
 - At least one number
 - At least one special character such as !, @, #, \$, %, ^, etc.

Note You can optionally configure HyperStore to require a higher minimum password length. You can also optionally configure additional password restrictions such as a password expiration period, a restriction against a user's new password being too similar to their previous password, a restriction on password reuse, and a restriction against too-frequent password changes. In the [common configuration setting group](#), these restrictions are configurable by the `common.user.password.*` settings.

- **To view the S3 secret access key** that corresponds to an access key ID, to the right of the access key ID click **View Secret Key**. A secret key display box appears which enables you to view the user's secret key and to copy it using `<Ctrl>-c` if you want to. Note that the **OK** and **Cancel** buttons have no effect other than to close the secret key display box.
- **To activate or deactivate an S3 access key**, to the right of the displayed access key ID click **Activate** or **Inactivate**.
- **To create a new S3 access key** click **Create New Key**. A new access key ID then displays in the access key list.
- **To delete an S3 access key**, to the far right of the displayed access key ID click **Delete**. You will be asked to confirm that you want to delete the key.

4. When you're done click **Close** to close the **User Credentials** panel.

6.8.1.7. Unlock a User Who's Locked Out of the CMC

A user may become temporarily locked out of the CMC if you have the password lock-out feature enabled in your system, and within a defined time interval the user makes too many login attempts using an incorrect password. The password-lock feature is configurable by the settings ["common.user.password.lock.enabled"](#) (page 756), ["common.user.password.lock.maxFailedAttempts"](#) (page 757), and ["common.user.password.lock.durationSec"](#) (page 756). By default the password lock-out feature is enabled if your original HyperStore version was 7.3 or later, or disabled if your original HyperStore version was earlier than that.

For a locked-out user, once the user stops attempting to log in with an incorrect password the lock-out releases automatically after a configurable time interval (30 minutes by default). Alternatively, you can release the lock immediately as described below:

1. Use the [Manage Users](#) page to **"Retrieve a User or List of Users"** (page 404).

When you've retrieved a user or list of users, a user who is currently locked out of the CMC will have a red padlock in their Status column.

USER ID	GROUP NAME	USER TYPE	STATUS	ACTIONS
user001	gr01	User	 Active	Unlock Edit Security Credentials Set QoS Delete
user002	gr01	User	Active	Edit Security Credentials Set QoS Delete

2. In the Actions column for the locked user, click **Unlock**.
3. After unlocking the user, confirm that a success message displays at the top of the CMC and that the red padlock no longer displays in the Status column for the user.

You have successfully unlocked user: user001

MANAGE USERS ➕ USER QOS DEFAULT ➕ NEW USER ⬇️ DOWNLOAD CMC USER ACTIVITY (CSV)

Search For A User By ID:

Enter prefix or complete user ID

Group Name: User Type: User Status:

SEARCH

USER ID	GROUP NAME	USER TYPE	STATUS	ACTIONS
user001	gr01	User	Active	Edit Security Credentials Set QoS Delete
user002	gr01	User	Active	Edit Security Credentials Set QoS Delete

The user can now log in to the CMC by supplying their correct user ID and password.

Note

* A group administrator logged in to the CMC can see if a user in her group is locked out (she can see the red padlock in the user's Status column). However, only a system administrator can intervene to unlock a user.

* If a system administrator becomes locked out of the CMC -- due to too many login attempts within an incorrect password -- the lock on that administrator can be released by a different system administrator (in the way described above) or by using the Admin API (see [POST /user/unlock](#) [Unlock a locked-out user](#)). Alternatively, as with any type of locked out user the lock-out will release automatically after the configurable time interval.

* Users' attempts to log in while they are locked out are recorded in the [CMC user audit log](#), as are administrator interventions to unlock a locked-out user.

6.8.1.8. Disable MFA for a User

HyperStore users can enable multi-factor authentication (MFA) on their own accounts, so that they are required to enter an MFA code in order to log into the CMC (in addition to entering their user name and password). All

user types -- regular users, group administrators, and system administrators -- can enable MFA on their own accounts, in the CMC's ["Security Credentials" \(page 572\)](#) page. Users can also disable MFA on their own accounts, in that same CMC page.

Optionally, the system can be [configured to require that all users set up and use MFA for CMC logins](#).

As a system administrator you can also disable MFA on other users' accounts. You may need to do this if, for example, a user has a problem with their virtual MFA device (MFA application on their computer or mobile device) that prevents them from being able to generate a valid MFA code and log into the CMC.

Note that:

- Only system administrators can disable MFA on another user's account. Group administrators do not have this capability.
- As a system administrator you can disable MFA on another user's account, but you cannot enable MFA on another user's account. For MFA to be enabled on a user's account, the user must do it themselves, in the CMC's ["Security Credentials" \(page 572\)](#) page.
- You cannot disable MFA for a user whose current status is Inactive (suspended). You must first [change the user's status to Active](#) and then you can disable MFA for the user.
- If the system is configured to require MFA for CMC logins, and you disable MFA on a user's account, the user will be required to set up MFA again the next time they try to log into the CMC.

To disable MFA on a user's account:

1. Use the [Manage Users](#) page to ["Retrieve a User or List of Users" \(page 404\)](#).

When you've retrieved a user or list of users, for users for whom MFA is currently enabled there will be a **Disable MFA** option in the "Actions" column. In this screen image, the users "PubsUser1" and "PubsUser2" have MFA enabled and the user "PubsGroupAdmin" does not:

USER ID	GROUP NAME	USER TYPE	STATUS	ACTIONS
PubsGroupAdmin	Pubs	GroupAdmin	Active	Edit Security Credentials Set QoS Delete
PubsUser1	Pubs	User	Active	Disable MFA Edit Security Credentials Set QoS Delete
PubsUser2	Pubs	User	Active	Disable MFA Edit Security Credentials Set QoS Delete

2. In the "Actions" column for the user for whom you want to disable MFA, click **Disable MFA**. When the CMC prompts you to confirm that you want to disable MFA for the user, click **OK**.

Once MFA has been disabled for the user, in the user list the "Actions" column will no longer display a **Disable MFA** option for the user.

The user can now log in to the CMC with just their user name and password, without having to supply an MFA code.

Note The user, if they wish, can subsequently enable MFA on their account again. The user could, for example, reinstall the problematic MFA application on their client device, or install a different MFA application on their device, and then in the CMC's ["Security Credentials" \(page 572\)](#) page the user can go through the process of enabling MFA on their account again.

6.8.1.9. Delete a User

You can delete a HyperStore service user from the system.

Note that by default you cannot delete:

- A user who still owns buckets. To disable this restriction and allow deletion of users who own buckets, use the configuration setting "[common.user.allowDeleteWithBuckets](#)" (page 754).
- A user who still owns IAM resources (IAM groups, users, roles, or policies). To disable this restriction and allow deletion of users who own IAM resources, use the configuration setting "[common.user.allowDeleteWithIAMResources](#)" (page 754).

Also, you cannot delete the default system administrator account (the user with user ID "admin"). This is not allowed.

To delete a user:

1. Use the [Manage Users](#) page to "[Retrieve a User or List of Users](#)" (page 404).
2. In the "Actions" column for the user that you want to delete, click **Delete**.
3. When prompted by the system, confirm that you want to delete the user.

Note If you delete a system admin user, this also deletes the user's HyperStore Shell account. For more information see "[Enabling the HSH and Managing HSH Users](#)" (page 106).

6.8.1.10. Download User Audit Log

[CMC Interface]

In the [Manage Users](#) page you can click **Download CMC User Activity** to download audit log data that records the following CMC activity:

- User logins (including failed login attempts) and logouts
- Creation, editing, and deletion of user accounts by system admins and group admins
- Changing of user passwords
- Events related to the [user lock-out feature](#)

The **Download CMC User Activity** function collects [ui-action.log](#) data from each of your HyperStore nodes and aggregates the data into one CSV file. By default configuration *ui-action.log* data is retained for 90 days, so the **Download CMC User Activity** function aggregates the last 90 days of CMC user action records from across your whole HyperStore system. The generation of the aggregated CSV file may take some time to complete.

The records in the CSV file are in this format:

Timestamp,SourceIP,AdminAccountID,Event,TargetCID,TargetUserId,TargetGroupId,TargetIAMEntityType,TargetIAMEntityName,Result

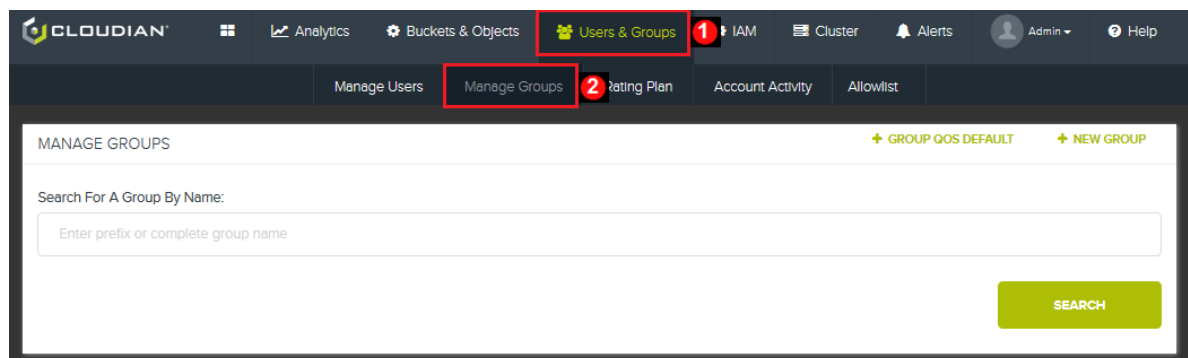
- The *Timestamp* is in format YYYY-MM-DD HH:MM:SS_mmm
- The *SourceIP* is the IP address from which the request originated
- The *AdminAccountID* is the user ID of the administrator, in cases when an administrator is taking action regarding a user (such as creating or deleting or unlocking a user). System admin IDs are in the form "0|<ID>" where 0 is the system admin group ID, and the "|" is escaped as "%7C" -- so for example "0|admin" displays as "0%7Cadmin". For actions taken by a group admin in regard to users within their group, the *AdminID* is the canonical user ID of the group admin.

- The *Event* is one of:
 - ChangePassword
 - CreateLocalUserForLdap
 - CreateUser
 - DeleteUser
 - DisableAccount
 - EditUser
 - EnableAccount
 - Login
 - Logout
 - MultiFactorAuthentication
 - UnlockLogin
- The *Target* fields identify the user. *CID* stands for canonical user ID.
- The *Result* is a status code string that describes the result of the action.

Note The data in the CSV file lists all the records from one of your nodes, then all the records from another node, and so on for all nodes. There is nothing in the file that distinguishes which CMC node records are from, or when the set of records from one node ends and the set of records from a different node begins (other than the jump in timestamp). You may wish to use a spreadsheet application to sort the entire file by the timestamp field or other fields.

6.8.2. Manage Groups

Path: **Users & Groups** → **Manage Groups**



Supported tasks:

- ["Add a Group" \(page 416\)](#)
- Work with existing groups:
 - ["Retrieve a Group or a List of Groups" \(page 420\)](#)
 - ["Edit a Group" \(page 421\)](#)
 - ["Set Quality of Service \(QoS\) Limits" \(page 430\)](#)
 - ["Delete a Group" \(page 422\)](#)

6.8.2.1. Add a Group

1. In the CMC's [Manage Groups](#) page, click **New Group**. This opens the **Add New Group** panel.

2. In the **Add New Group** panel, complete the group information:

Group Name (mandatory)

- Must be unique within your entire HyperStore service.
- Only letters, numbers, dashes, and underscores are allowed.
- Maximum allowed length is 64 characters.

Note If you intend to enable LDAP authentication for this group (as described further below), the "Group Name" should if possible be the exact name of the group as it exists in the LDAP system. If the "Group Name" field's character restrictions prevent you from using the group's exact name from LDAP, enter something similar as the "Group Name" and then use the "LDAP Org Unit" field to specify the group's exact name from LDAP.

Group Description (optional)

- Maximum allowed length is 64 characters.

Rating Plan (optional)

- Rating plan to assign to the group for billing calculation purposes. This rating plan will apply to each user in the group, with the exception of any users to whom you individually assign a different rating plan.
- If you don't choose a rating plan for the group, the **"Default Rating Plan"** (page 426) (Default-RP) is automatically assigned to the group.

If your service deployment has multiple service regions, rating plan assignment is on a per-region basis. A "Region" drop-down list displays which includes each of your region names. For each region, assign the group a rating plan that will apply to the group's service use within that region.

Enable S3 endpoints display filter (optional)

- Select this option if you want to filter the S3 endpoints (S3 service URLs) that this group's users

will see when they log into the CMC and they go to their [Security Credentials](#) page. If you select this checkbox you will then be presented a list to select which of the system's configured S3 HTTP endpoints, S3 HTTPS endpoints, and S3 website endpoints will display for this group's users in the **Security Credentials** page.

- If you do not select this option, then by default **all** of the system's S3 HTTP endpoints, S3 HTTPS endpoints, and S3 website endpoints will display for this group's users when they view their **Security Credentials** page.

Note If you enable the S3 endpoints display filter and choose the endpoints to display for this group's users, and then at a later point in time you delete one of those S3 endpoints from the HyperStore system configuration and replace it with a different endpoint (by using the [installer's function for changing service endpoints](#)), then you must subsequently [edit the group configuration](#) to update the group's S3 endpoint display filtering. Otherwise neither the original S3 endpoint nor the replacement S3 endpoint will display for the group.

3. Optionally, enable and configure LDAP authentication for the group, so that members of the group can log into the CMC with their LDAP credentials. For more information see ["Enabling and Configuring LDAP Authentication for Group Members"](#) (page 417) below.
4. Click **Save**.

Enabling and Configuring LDAP Authentication for Group Members

Optionally you can enable LDAP authentication for the group by selecting the "Enable LDAP Authentication" checkbox. When LDAP authentication is enabled for a group, new users who belong to the group can log into the CMC using their LDAP credentials and the CMC will **automatically provision those users into HyperStore** (including establishing S3 access credentials for the users, for those who are not system admin users). Subsequently whenever such users log in, the CMC will recognize them as registered HyperStore users but will continue to authenticate them against the LDAP system rather than by reference to CMC-based passwords.

Note For more background information on the LDAP integration feature see ["LDAP Integration"](#) (page 278).

When you select the "Enable LDAP Authentication" checkbox, additional fields display in which you can configure how the CMC will authenticate this group's members against your Active Directory or other LDAP system.

ADD NEW GROUP

Active Group

Group Name: *

accounting

Group Description:

Rating Plan:

Default-RP

☐ Enable S3 endpoints display filter

☒ Enable LDAP Authentication

LDAP Org. Unit:

LDAP Server URL: *

LDAP User DN Template: *

LDAP Search:

LDAP Search User Base:

LDAP Match Attribute:

TEST

CANCEL

SAVE

LDAP Org Unit (optional)

The group's name from the LDAP system. This would typically be the group's "ou" (Organization Unit) value in the LDAP system, but could also be for example the "l" (Location) value or "memberOf" value -- depending on which LDAP attribute is to be used to identify users' group membership when the CMC authenticates them against the LDAP system.

If you use the variable `{groupid}` in any of the other LDAP authentication configuration attributes, when implementing LDAP authentication HyperStore will automatically replace the variable with the LDAP Org Unit value.

LDAP Server URL (mandatory)

Use this attribute to specify the URL that the CMC should use to access the LDAP Server when authenticating users in this group. For example:

`ldap://my.ldap.server:389`

Note that if you use *ldaps* (LDAP secured by TLS), the LDAP server must use a CA-verified certificate not a self-signed certificate. HyperStore does not support connecting to an LDAP server that's using a self-signed SSL certificate.

Note To use LDAP authentication for [HyperStore Shell](#) users, your LDAP server must use either TLS or START_TLS.

LDAP User DN Template (mandatory)

Use this attribute to specify how users within this group will be authenticated against the LDAP system when they log into the CMC. It is a template that defines how user names supplied during CMC login will be mapped to user-identifying information in the LDAP system. Two typical ways of configuring this

template are:

- **Distinguished Name.** With this approach the template specification would include the LDAP attribute "uid" set to equal the CMC token "{userId}" (as shown in the example below), the LDAP attribute "ou" set to equal the group's organizational unit value from the LDAP system, and the domain components from LDAP. For example:

```
uid={userId},ou=engineering,dc=my-company,dc=com
```

With the DN template above, LDAP-enabled users from this group will log in with their LDAP *uid* value as their CMC user ID. During login the CMC will also verify that the *ou* value in the user's LDAP record matches against the *ou* value from the template.

Note If you are configuring LDAP authentication for the System Admin group, use the Distinguished Name approach for the user DN template. Also, after enabling and configuring LDAP authentication for the System Admin group in the CMC, see ["Additional Configuration Step Required for LDAP Authentication of System Admin Group Users"](#) (page 420).

- **userPrincipalName.** With this approach the template would simply map the LDAP attribute "userPrincipalName" to the CMC token "{userId}", like this:

```
userPrincipalName={userId}
```

With the approach above LDAP-enabled users from this group will log in with their LDAP *userPrincipalName* value (such as <user>@<domain>) as their CMC user ID. Optionally, to implement additional LDAP-based authorization filters such as the users' group or location, you can use the "LDAP Search", "LDAP Search User Base", and "LDAP Match Attribute" settings as described below.

LDAP Search attributes (optional)

If you want to establish a LDAP-based user authorization filter to complement the user authentication template that you set with the "LDAP User DN Template" field, then use the "LDAP Search", "LDAP Search User Base", and "LDAP Match Attribute" fields to configure the filter. If you do so, then LDAP-enabled users from this group when logging in to the CMC will need to meet the requirements of the authentication template and also the requirements of the filter.

LDAP Search

Use the "LDAP Search" field to specify the user identifier type that you used in the "LDAP User DN Template" field, in format "(<LDAP_user_identifier_attribute>={userId})". This is used to retrieve a user's LDAP record in order to apply the filtering. Here are two examples:

- (uid={userId})
- (userPrincipalName={userId})

LDAP Search User Base

Use the "LDAP Search User Base" field to specify the LDAP search base from which the CMC should start when retrieving the user's LDAP record in order to apply filtering. For example, *dc=my-company,dc=com*. Or for another example, *uid={userId},ou=engineering,dc=my-company,dc=com*.

LDAP Match Attribute

Use the "LDAP Match Attribute" field to specify an LDAP attribute value against which LDAP-enabled users in this group must match in order to be authorized to log into the CMC. Use this

format: `<attribute>=<value>`. Here are two "LDAP Match Attribute" examples:

- `l=California`
- `memberOf=Sales`

Remember to click **Save** after configuring LDAP authentication for the group.

Important Note About Provisioning of Users in a Group That Is Enabled for LDAP Authentication

Within a HyperStore group that has LDAP authentication enabled you can have both LDAP-authenticated users and (if you wish) users who are authenticated by a CMC-based password rather than LDAP:

- **For users who you want to be authenticated by LDAP, do not manually create those users through the CMC's Add New User interface** (or the Admin API `PUT /user` method). Instead, let the CMC create those users automatically when they log in for the first time and are successfully authenticated against the LDAP system.
- Only use the CMC's **Add New User** interface (or Admin API `PUT /user` method) if you want to also have some users who are **not** LDAP-authenticated. Users who you create through the **Add New User** interface or Admin API will be authenticated by their CMC-based passwords -- not by the LDAP system.

Note If you want a **group administrator** to be authenticated by LDAP, have him or her log into the CMC using their LDAP credentials. Once this occurs and the CMC automatically provisions them into the system as a regular user, you can subsequently edit their user profile (using the CMC's [Edit User](#) function) to promote them to the group admin role.

Additional Configuration Step Required for LDAP Authentication of System Admin Group Users

For LDAP authentication to work for system admin users when they log into the [HyperStore Shell](#), along with enabling LDAP for the System Admin group as described above you must also perform this additional configuration step:

1. Log in to the Config Controller node (as root or as a locally authenticated HyperStore Shell user).
2. Set the Distinguished Name for binding to your LDAP service, and the password:

```
hsctl config set hsh.ldap.bindDN=<bind Distinguished Name>
hsctl config set hsh.ldap.bindPassword=<bind password>
hsctl config apply hsh
```

6.8.2.2. Retrieve a Group or a List of Groups

In the CMC's [Manage Groups](#) page you can retrieve a single group or a filtered list of groups.

1. In the "Search for a Group By Name" field:
 - To retrieve just one specific group, enter the full group name. This field is case-sensitive.

Note To retrieve the System Admin group, enter "0" as the group name.

- To retrieve a group list filtered by group name prefix, enter the prefix.
- To retrieve a list of all groups, leave this field empty.

- Click **Search**. Your filtered search results then display in the lower part of the page.

GROUP NAME	GROUP DESCRIPTION	STATUS	ACTIONS
TestGroup	Test Group Account	Active	Group QoS User QoS Group Default Edit Delete

6.8.2.3. Edit a Group

You can change attributes of an existing group, such as the group's rating plan assignment, service status, or LDAP integration.

- Use the CMC's [Manage Groups](#) page to **"Retrieve a Group or a List of Groups"** (page 420).

Note If you want to retrieve the **System Admin group** in order to edit that group's attributes, the group name is **0**.

- In the "Actions" column for the group click **Edit**. This displays a panel for editing group attributes.

GROUP NAME	GROUP DESCRIPTION	STATUS	ACTIONS
TechPubs		Active	Group QoS User QoS Group Default Edit Delete

☒ Active Group

Group Description:

Rating Plan:

☐ Enable S3 endpoints display filter

☐ Enable LDAP Authentication

CANCEL
SAVE

Note If you are editing the System Admin group, only the LDAP Authentication settings will display. The other attributes are not applicable to the System Admin group.

- Make your desired changes to the group attributes:

Active Group

- Active Group checkbox checked — Group is active. Group's users have access to the storage service and the CMC.
- Active Group checkbox unchecked — Group is suspended. All the users in the group — including the group administrator(s) — will be denied access to the storage service and the CMC. However, the group's users will remain in the system and their objects will remain in storage.

Group Description

- Maximum allowed length is 64 characters.

Rating Plan

- Use the "Rating Plan" drop-down list to assign the group a rating plan for billing calculation

purposes. This rating plan will apply to each user in the group, with the exception of any users to whom you individually assign a different rating plan.

Note for multi-region systems

If your service deployment has multiple service regions, rating plan assignment is on a per-region basis. A "Region" drop-down list displays which includes each of your region names. For each region, assign the group a rating plan that will apply to the group's service use within that region.

Enable LDAP Authentication

- For information about this option, in the "Add a Group" section see ["Enabling and Configuring LDAP Authentication for Group Members" \(page 417\)](#) (the LDAP configuration interface is the same whether you're adding a group or editing an existing group).

Note If you enable LDAP Authentication for an existing group to which users have already been added, those existing users will continue to be authenticated by reference to their CMC-based passwords -- not by LDAP authentication. LDAP authentication will be supported only for new users. For more details see ["Enabling and Configuring LDAP Authentication for Group Members" \(page 417\)](#).

Enable S3 endpoints display filter (optional)

- Select this option if you want to filter the S3 endpoints (S3 service URLs) that this group's users will see when they log into the CMC and they go to their [Security Credentials](#) page. If you select this checkbox you will then be presented a list to select which of the system's configured S3 HTTP endpoints, S3 HTTPS endpoints, and S3 website endpoints will display for this group's users in the **Security Credentials** page.
- If you do not select this option, then by default **all** of the system's S3 HTTP endpoints, S3 HTTPS endpoints, and S3 website endpoints will display for this group's users when they view their **Security Credentials** page.

Note If you enable the S3 endpoints display filter and choose the endpoints to display for this group's users, and then at a later point in time you delete one of those S3 endpoints from the HyperStore system configuration and replace it with a different endpoint (by using the [installer's function for changing service endpoints](#)), then you must subsequently edit the group configuration to update the group's S3 endpoint display filtering. Otherwise neither the original S3 endpoint nor the replacement S3 endpoint will display for the group.

4. Click **Save**.

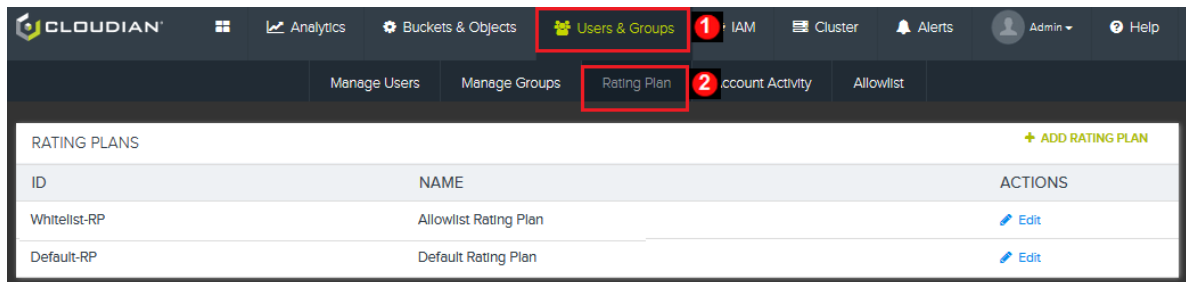
6.8.2.4. Delete a Group

Note You cannot delete a group that currently has users in it. You must delete the users first, one by one (the CMC does not currently support bulk deletion of users). After deleting all the users you can delete the group.

1. Use the CMC's [Manage Groups](#) page to "[Retrieve a Group or a List of Groups](#)" (page 420).
2. In the "Actions" column for the group that you want to delete, click **Delete**.
3. When prompted by the system, confirm that you want to delete the group.

6.8.3. Rating Plan

Path: **Users & Groups** → **Rating Plan**



Supported tasks:

- "[Add a Rating Plan](#)" (page 423)
- "[Edit a Rating Plan](#)" (page 425)
- "[Delete a Rating Plan](#)" (page 427)

Note The **Rating Plan** page is for creating and maintaining rating plans. It is not for assigning a rating plan to users. To assign a rating plan to a group of users, use the CMC's [Manage Groups](#) interface. To assign a rating plan to a specific user, use the [Manage Users](#) interface.

For an overview of HyperStore user billing functionality, see "[Billing Feature Overview](#)" (page 258).

6.8.3.1. Add a Rating Plan

Rating plans specify pricing for various types and levels of user activity, to facilitate billing. Through the CMC's [Rating Plan](#) page you can create and configure new rating plans.

Note HyperStore includes a pre-configured [Default Rating Plan](#) named "Default-RP". If you wish you can [edit the contents](#) of that plan. There is also a pre-configured plan named "Whitelist-RP" which supports the HyperStore [allowlist feature](#).

To create a new rating plan:

1. In the **Rating Plans** page click **Add Rating Plan**. This opens the **Add Rating Plan** panel.

2. Assign the plan a unique ID, and optionally a Name.
3. From the Currency drop-down list, choose the currency units on which the plan's pricing structure will be based (for example, USD for U.S. dollars).
4. Specify the billing rates that will constitute the rating plan. You can specify billing rates per:
 - Average GiBs of data in storage during the calendar month for which the bill is calculated ("Storage-Size Rates")
 - GiBs of data uploaded to the system. ("Data-Transfer-IN Rates")
 - GiBs of data downloaded from the system. ("Data-Transfer-OUT Rates")
 - 10,000 HTTP GET or HEAD requests. ("HTTP[S] GET/HEAD Rates")
 - 10,000 HTTP PUT or POST requests. ("HTTP[S] PUT/POST Rates")
 - 10,000 HTTP DELETE requests. ("HTTP[S] DELETE Rates")

IMPORTANT ! If you want to bill for data upload or download volume, or for HTTP request volume, you must enable the [Track/Report Usage for Request Rates and Data Transfer Rates](#) setting on the CMC's **Configuration Settings** page. By default this setting is disabled and the system does not maintain per-user HTTP request counts and data transfer byte counts.

Note For an example of these rates applied to a user's bill, see ["Example of a Rating Plan Applied to Calculate a User's Monthly Bill"](#) (page 260)

For each rated activity you can establish either a single-tier or multi-tier pricing structure, based on activity level.

With **single-tier pricing**, the per-unit rate charged for the activity is not impacted by the activity level: whether the activity level is low or high, the per-unit charge remains the same. The initial display for each rated activity accommodates single-tier pricing.

For example, in the screen below, for Storage-Size Rates, a single-tier pricing structure can be established by simply specifying a Rate Per GiB Per Month. In this example, the rate is \$2 per GiB-month.

That same per GiB-month rate applies whether a user stores, for instance, 10 GiB-month (\$20 charge) or 100 GiB-month (\$200 charge).

STORAGE SIZE (GiB)	RATE PER GiB-MONTH
	2

ADD

To create a **multi-tier pricing** structure for a rated activity, first click on the activity type (for example, "Data-Transfer-IN Rates"). Then click **Add** once for each additional pricing tier that you want to specify, beyond a single tier. For example, in the screen below, **Add** has been clicked twice to enable a three tier pricing structure for Data-Transfer-IN Rates. In this example, the first five GiB of data uploaded during the billing period are priced at \$2 per GiB; the next five GiB are priced at \$2.50 per GiB; and any uploads beyond that level (above the 10 GiBs encompassed by the first two tiers) are priced at \$3 per GiB.

	DATA TRANSFER (GiB)	RATE PER GiB	
Up to	5	2	Delete
Next	5	2.5	Delete
Above	10	3	

ADD

Note that the numbers that you enter in the "Rate Per" column signify units of the currency that you chose in the upper part of the **Add Rating Plan** panel. You can use decimals if you want; for example, if your currency is dollars, you can enter ".1" in the "Rate Per" column to indicate a charge of 10 cents (.1 dollars) per unit of activity.

Note For HTTP request rates, the pricing is based on blocks of 10,000 requests. So for example, if you want the first 50,000 requests to be charged at a certain price per block of 10,000, then in the "Number of 10,000 Requests" field for that tier, **enter 5 — not 50,000**.

- When you're done creating the new rating plan, click **Save** at the bottom of the **Add Rating Plan** panel. The plan will then appear in the **Rating Plans** list.

6.8.3.2. Edit a Rating Plan

- In the CMC's [Rating Plan](#) page click **Edit** to the right of the plan name. This opens a panel in which you can edit the plan attributes.

2. Edit the plan attributes. For attribute descriptions see .
3. Click **Save**.

Note When billing is calculated at the end of a month, the most current version of the rating plan is applied to the whole month's activity. For example, suppose you have a "Gold" rating plan. Late in January, you edit the pricing structure of the Gold plan. At the end of January when bills are generated for customers on the Gold plan, all their January activities will be billed in a way that reflects the modified pricing structure that you established in late January.

Note HyperStore includes a pre-configured [Default Rating Plan](#) named "Default-RP". If you wish you can edit the contents of that plan. There is also a pre-configured plan named "Whitelist-RP" which supports the HyperStore [allowlist feature](#).

6.8.3.3. Default Rating Plan

The HyperStore system comes with a default rating plan (with unique ID "Default-RP"). In the CMC's [Rating Plan](#) page you can [edit this plan](#) but you cannot delete it.

Groups and users that you do not explicitly assign a rating plan will automatically be assigned the default rating plan.

The default rating plan's currency is US dollars and the pricing structure is as follows:

- Storage billing
 - First tier, up to 1 GiB: \$0.14 per GiB-month
 - Second tier, from 1+ GiB to 6 GiB: \$0.12 per GiB-month
 - Third tier, above 6 GiB: \$0.10 per GiB-month
- Data Transfer IN
 - First tier, up to 1 GiB: \$0.20 per GiB
 - Second tier, above 1 GiB: \$0.10 per GiB

- Data Transfer OUT
 - First tier, up to 1 GiB: \$0.20 per GiB
 - Second tier, above 1 GiB: \$0.10 per GiB
- HTTP GETs/HEADs
 - First tier, up to 10 blocks of 10,000 (that is, up to 100,000 GETs/HEADs): \$0.02 per 10,000 GETs/HEADs
 - Second tier, above 10 blocks of 10,000 (above 100,000 GETs/HEADs): \$0.01 per 10,000 GETs/HEADs
- HTTP PUTs/POSTs
 - No tiering: \$0.02 per 10,000 PUTs/POSTs
- HTTP DELETES
 - No charge for DELETES

6.8.3.4. Delete a Rating Plan

In the CMC's [Rating Plan](#) page click **Delete** to the right of the name of the plan that you want to delete. After you confirm that you want to take this action, the rating plan will be deleted from the system.

Note If you delete a plan that is currently assigned to some users, those users will be automatically switched to their group default rating plan, or to the system [Default Rating Plan](#) ("Default-RP") if no group default plan has been set.

You cannot delete the system default rating plan, or the "Whitelist-RP" plan (which supports the HyperStore [allowlist feature](#)).

6.8.4. Account Activity

Path: **Users & Groups** → **Account Activity**

The screenshot shows the Cloudian CMC interface. The top navigation bar includes 'Users & Groups' (1) and 'IAM'. Below it, the 'Account Activity' tab (2) is selected. The main form has filters for Group Name (CloudianTest), User ID, Time Period (Dec 2019), Region (region1), and Traffic Type (Normal). A 'DISPLAY REPORT' button is at the bottom right.

Supported task:

- Generate a billable activity report for a specified user

IMPORTANT ! Billing calculation is derived from hourly rollup usage data. The retention period for hourly rollup usage data is configured by the setting "[s3.usage.reports.hourlyRollupTTL](#)" (page 839). The default retention period is 65 days. Once this rollup data is deleted it can no longer be used to generate users' bills. So be sure to have `s3.usage.reports.hourlyRollupTTL` set to a value large enough to accommodate your billing routine.

To generate a billable activity report for a user:

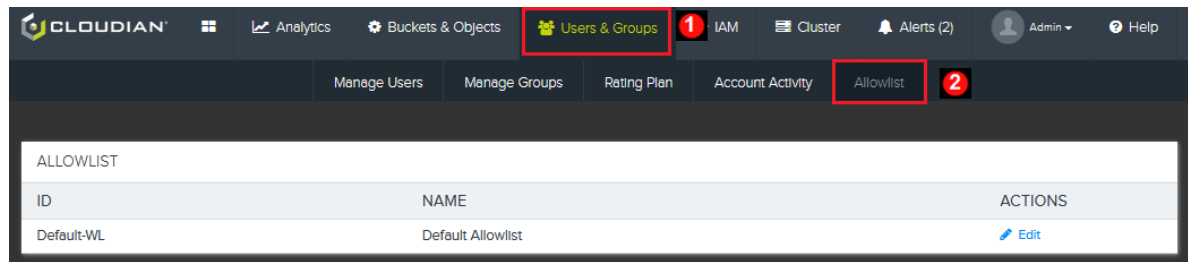
1. Choose a "Group Name" from the drop-down list and enter a "User ID" for the user.
2. Choose a "Time Period" from the drop-down list. The billing periods are calendar months. You can only create a bill for a past, completed month; you cannot create a bill for the current, in-progress month. The default is the most recent completed month.
3. Choose a "Region" from the drop-down list. Bills are calculated per service region. The CMC does not currently support multi-region aggregate bill generation.
4. Choose a "Traffic Type" from the drop-down list. The options are "Normal" or "Allowlist". "Allowlist" refers to request traffic that originates from a configurable list of source IP addresses that are granted special pricing -- typically zero charge for upload/download traffic. This is only an option if the "Allowlist" feature is enabled in the system. "Normal" refers to all other traffic. For more information see "[Creating an Allowlist for Free Traffic \(Optional\)](#)" (page 261).
5. Click **Display Report** to display activity and charges for the selected billing period.

The image below shows a sample report. The upper part displays summary report parameters including the name of the rating plan that has been applied to the user's activity. The lower middle shows the charging rules from the rating plan (in this case, a three-tier charging structure for price per GiB-month, based on the average level of storage volume used), and the lower right shows the user's usage level for the month (in this case 108 GiB-months) and the resulting charges.

ACCOUNT ACTIVITY				Print
Account Info				
<u>Billing Period:</u>	10/01/2015-10/31/2015	<u>Name:</u>	Test User	
<u>Date Printed:</u>	Nov 25 2015	<u>Account Id:</u>	test	
<u>Region:</u>	region1	<u>Group Description:</u>	Test Group	
<u>Rating Plan:</u>	Default-RP	<u>Traffic Type:</u>	Normal	
Storage Bytes				
	Up to 1 GiB-Month : 0.14 (USD)	108 GiB-Month	10.94	
	Next 5 GiB-Month : 0.12 (USD)			
	Above : 0.10 (USD)			
	Total (USD)		10.94	

6.8.5. Allowlist

Path: **Users & Groups** → **Allowlist**



Supported tasks:

- Add or remove IP addresses from the allowlist

Note This page displays in the CMC if the system configuration setting "[admin.-billing.allowlist.enabled](#)" (page 702) is set to *true* (as it is by default).

The HyperStore billing "allowlist" feature enables service providers to specify a list of IP addresses or subnets that are allowed to have free-of-charge S3 traffic with the HyperStore system. For S3 requests originating from IP addresses on the allowlist, a special rating plan is used that applies zero charge to all the traffic-related pricing metrics:

- Price per GiB data transferred in
- Price per GiB data transferred out
- Price per 10,000 HTTP PUT requests
- Price per 10,000 HTTP GET requests
- Price per 10,000 HTTP DELETE requests

The billing allowlist feature affects only the pricing of **traffic**. It does **not** affect the pricing of data **storage**. For data storage billing, users' regular assigned rating plan pricing is applied.

The special rating plan assigned to allowlisted addresses — the rating plan that prices traffic at "0" — has rating plan ID "Whitelist-RP". If you wish, you can edit this rating plan in the [Rating Plan](#) page of the CMC (for example, if you want traffic originating from your allowlisted IP addresses to be specially priced but not free).

Note In the current release you can only have one allowlist, with only one rating plan applied to it.

Note If you are using load balancers in front of the HyperStore S3 Service, the allowlist feature will only work if you use PROXY Protocol between the load balancers and the S3 Service. This protocol allows the load balancers to pass the IP addresses of originating clients to the S3 Service along with the S3 requests. For more information about enabling PROXY Protocol support on the S3 Service side, see the system configuration setting "[s3.proxyProtocol.enabled](#)" (page 826). For guidance on configuring the load balancers consult with Cloudian Sales Engineering or Support.

Note that using the "X-Forwarded-For" HTTP header is **not** sufficient to support the allowlist feature. You must use PROXY Protocol if you have load balancers in front of the S3 Service and want to use the allowlist feature .

Note for multi-region systems

In a multi-region HyperStore deployment, the allowlist is applied the same across all regions. There is no

support for region-specific allowlists.

To add or remove IP addresses from the allowlist:

1. In the **Allowlist** page, to the right of the allowlist name, click **Edit**. This opens the **Edit Allowlist** panel.

ID	NAME	ACTIONS
Default-WL	Default Allowlist	Edit

EDIT ALLOWLIST

ID

Default-WL

Name

Default Allowlist

Rating Plan

Whitelist-RP

IP/Subnet

IP/Subnet

SAVE

2. If you want you can change the name of the allowlist ("Default Allowlist"), but you cannot change its ID ("Default-WL").
3. In the "IP/Subnet" box, enter IP addresses or subnets one at a time, pressing your keyboard's Enter key after each entry. You can enter as many as you want. (If you are editing a list that you created through this dialog previously, you also have the option of removing IP addresses or subnets from the list.)

Note The system will validate IP addresses and subnets for correct syntax. The entire list will be rejected if any address or subnet fails the syntax validation check.

4. After making your changes, click **Save**.

For billing purposes, changes that you make to the composition of the allowlist (by adding or deleting IP addresses or subnets) will take effect starting with the next hourly roll-up of HyperStore usage data.

6.8.6. Set Quality of Service (QoS) Limits

Use the CMC's **Manage Users** page and **Manage Groups** page to set Quality of Service (QoS) limits for users and groups.

Note By default the HyperStore system's enforcement of QoS restrictions is **disabled**. For information about enabling the QoS feature -- and information about the option of having the system generate alerts when users hit QoS thresholds -- see ["Preparing the QoS Feature"](#) (page 248).

To set QoS limits:

1. Navigate to the QoS configuration panel for the type of QoS controls that you want to set. The table below shows the path to each QoS panel. It also shows how each panel is pre-populated with default

settings.

QoS Task	Path to Configuration Panel	Configuration Panel Name	Panel's Default Values
Set a system default user QoS profile	In the Manage Users page, click User QoS Default .	User QoS Limits: Defaults	User QoS Default. Defaults to "unlimited" for all QoS settings.
Set a default user QoS profile for members of a group	In the Manage Groups page, retrieve the group. Then for the group click User QoS Group Default .	User QoS Limits: Group Defaults	Defaults to system default user QoS profile.
Set a user-specific QoS profile	In the Manage Users page, retrieve the user. Then click Set QoS for the user.	User QoS Limits: Overrides	Defaults to default user QoS profile for the user's group.
Set a system default group QoS profile	In the Manage Groups page, click Group QoS Default .	Group QoS Limits: Defaults	Defaults to "unlimited" for all QoS settings.
Set a group-specific group QoS profile	In the Manage Groups page, retrieve the group. Then click Group QoS for the group.	Group QoS Limits: Overrides	Defaults to system default group QoS profile.

All QoS configuration panels have the same appearance and the same options, as shown in the sample panel below:

USER QOS LIMITS: OVERRIDES

QOS ITEM	WARNING LIMIT	HIGH LIMIT
Storage Quota (KiB)	Unlimited ☒	Unlimited ☒
Storage Quota Count	Unlimited ☒	Unlimited ☒
Request Rate (Requests/minute)	Unlimited ☒	Unlimited ☒
Data Bytes IN (KiB/minute)	Unlimited ☒	Unlimited ☒
Data Bytes OUT (KiB/minute)	Unlimited ☒	Unlimited ☒

DELETE OVERRIDES

CANCEL

SAVE

- Make your desired edits to the QoS limits. If you deselect an "Unlimited" checkbox, a field appears in which you can enter a numerical limit. The supported QoS limit types are described below.

Storage Quota (KiB) — Warning Limit

Storage quota warning threshold ("soft limit"), in number of KiBs

Implementation detail:

- *For user QoS* — If a user's total stored data reaches this limit, a WARN level message is written to the S3 Service's application log.
- *For group QoS* — If a group's total stored data reaches this limit, a WARN level message is written to the S3 Service's application log.

Storage Quota (KiB) — High Limit

Storage quota maximum ("hard limit"), in number of KiBs

Implementation detail:

- *For user QoS* — If a user's total stored data reaches this limit, the user will be blocked from uploading additional data until she deletes some of her currently stored data. Also, a WARN level message is written to the S3 Service's application log.
- *For group QoS* — If a group's total stored data reaches this limit, all of the group's users will be blocked from uploading additional data until some of their currently stored data is deleted. Also, a WARN level message is written to the S3 Service's application log.

Storage Quota Count — Warning Limit

Storage quota warning threshold ("soft limit"), in total number of objects. Note that folders count as objects, as well as files.

Implementation detail:

- *For user QoS* — If a user's total stored data reaches this limit, a WARN level message is written to the S3 Service's application log.
- *For group QoS* — If a group's total stored data reaches this limit, a WARN level message is written to the S3 Service's application log.

Storage Quota Count — High Limit

Storage quota maximum ("hard limit"), in total number of objects. Note that folders count as objects, as well as files.

Implementation detail:

- *For user QoS* — If a user's total stored data reaches this limit, the user will be blocked from uploading additional data until she deletes some of her currently stored data. Also, a WARN level message is written to the S3 Service's application log.
- *For group QoS* — If a group's total stored data reaches this limit, all of the group's users will be blocked from uploading additional data until some of their currently stored data is deleted. Also, a WARN level message is written to the S3 Service's application log.

Request Rate — Warning Limit

Request rate warning threshold ("soft limit"), in total number of HTTP requests per minute.

Implementation detail:

- *For user QoS* — On receipt of a first HTTP request from a user, a 60 second timer is started for that user. If during the 60 seconds the total number of requests reaches the Request Rate Warning Limit, a WARN level message is written to the S3 Service's application log, and requests

from the user continue to be fulfilled. At the end of the 60 seconds, the request counter for the user is reset. Subsequently, the next request that comes in from the user triggers the start of a new 60 second interval, and the process repeats.

- *For group QoS* — The implementation is the same as for user QoS, except that it applies to requests from all users in the group, in the aggregate. For example, a request from any user in the group triggers the start of the 60 second timer, and subsequent requests from any user in the group are counted toward the per-minute total.

HTTP DELETE requests are not counted toward Request Rate controls.

Request Rate — High Limit

Request rate maximum ("hard limit"), in total number of HTTP requests per minute.

Implementation detail:

- *For user QoS* — On receipt of a first request from a user, a 60 second timer is started for that user (the same timer described in Request Rate Warning Limit). If during the 60 seconds the number of requests reaches Request Rate High Limit, the system temporarily blocks all requests from the user (and also writes a WARN level message to the S3 Service's application log). At the end of the 60 seconds the block on requests is released and the request counter is reset. Subsequently, the next request that comes in from the user triggers the start of a new 60 second interval, and the process repeats.
- *For group QoS* — The implementation is the same as for user QoS, except that it applies to requests from all users in the group, in the aggregate. For example, a request from any user in the group triggers the start of the 60 second timer, and subsequent requests from any user in the group are counted toward the per-minute total. If a block is triggered by the high limit being reached, the block applies to all users in the group.

Data Bytes IN (KiB/minute) — Warning Limit

Inbound data rate warning threshold ("soft limit"), in KiBs per minute.

This works the same as described for the Request Rate Warning Limit, except what's counted during each timed 60 second interval is inbound kibibytes of data.

Data Bytes IN (KiB/minute) — High Limit

Inbound data rate maximum ("hard limit"), in KiBs per minute.

This works the same as described for the Request Rate High Limit, except what's counted during each timed 60 second interval is inbound kibibytes of data. Note that if a block is triggered by the Data Bytes IN High Limit being reached, the block applies to all HTTP request types (not just PUTs.)

Data Bytes OUT (KiB/minute) — Warning Limit

Outbound data rate warning threshold ("soft limit"), in KiBs per minute.

This works the same as described for the Request Rate Warning Limit, except what's counted during each timed 60 second interval is outbound kibibytes of data.

Data Bytes OUT (KiB/minute) — High Limit

Outbound data rate maximum ("hard limit"), in KiBs per minute.

This works the same as described for the Request Rate High Limit, except what's counted during each timed 60 second interval is outbound kibibytes of data. Note that if a block is triggered by the Data Bytes OUT High Limit being reached, the block applies to all HTTP request types (not just GETs.)

Note The system does not notify users who have exceeded a QoS Warning Limit.

3. Click **Save**.

Deleting QoS Overrides to Restore Defaults

Each QoS configuration panel includes a **Delete Overrides** button. The table below shows what this button does in each QoS panel.

QoS Configuration Panel	What the Delete Overrides Button Does
User QoS Limits: Defaults	Deletes all group-specific default user QoS profiles. Each group's default user QoS profile will revert to the system default user QoS profile. (This action will not delete QoS overrides set at the individual user level.)
User QoS Limits: Group Defaults	Deletes all user-specific QoS profiles for users within a group. All users in the group will revert to the default user QoS profile for the group.
User QoS Limits: Overrides	Deletes a user-specific QoS profile. The user will revert to the default user QoS profile for the user's group
Group QoS Limits: Defaults	Deletes all group-specific group QoS profiles. All groups will revert to the system default group QoS profile.
Group QoS Limits: Overrides	Deletes a group-specific QoS profile. The group will revert to the system default group QoS profile.

QoS Controls in a Multi-Region System

In a multi-region system, QoS controls are configured and enforced separately for each region.

If you have a multi-region system, a drop-down list of all your regions will display at the top of each QoS configuration panel. Use the drop-down list to choose the region for which you are setting QoS controls. Be sure to establish QoS limits for each region, clicking **Save** before moving on to the next region.

The QoS limits that you establish for a service region will be applied only to group and user activity in that particular region. For example, if you have a two-region system with regions named "West" and "East", you might set your QoS limits in the "West" region so that each user is allowed to store 20GiB of data. The 20GiB storage cap applies only to user activity in your "West" region. You would also have a separate cap configured for the "East" region. For example, users might also be allowed 20GiB of storage in the "East" region (for a total of 40GiB per user across the entire multi-region system); or a different value might be configured for the "East" region such as 10GiB or 30GiB.

6.9. IAM

The **IAM** tab contains the following functions:

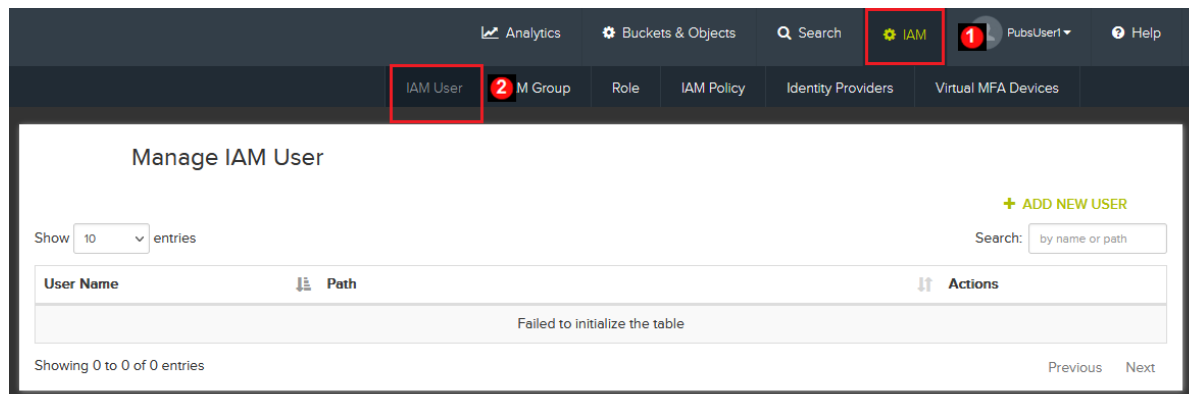
- [Manage IAM User](#)
- [Manage IAM Group](#)
- [Manage IAM Policy](#)
- [Manage IAM Role](#)
- [Manage Identity Providers](#)
- ["Manage MFA Devices"](#) (page 464)

Note For an overview of HyperStore's IAM feature, including limitations on the scope of HyperStore's IAM support, see [HyperStore Support for the AWS IAM API](#).

Note S3 credentials are needed to access the HyperStore IAM Service (and the CMC's IAM functions), and the CMC's system administrative user named "admin" does not have these credentials by default. For information on creating S3 credentials for the "admin" user so that this user can access the IAM Service, see [Default 'admin' User Does Not Have S3 Access Credentials](#).

6.9.1. Manage IAM User

Path: **IAM** → **IAM User**



Supported tasks:

- ["Add an IAM User"](#) (page 436)
- Work With an Existing IAM User
 - ["Select an IAM User to Work With"](#) (page 437)
 - ["Manage an IAM User's S3 Access Keys"](#) (page 437)
 - ["Manage an IAM User's Group Membership"](#) (page 438)
 - ["Manage an IAM User's Permissions"](#) (page 440)
- ["Delete an IAM User"](#) (page 442)

You can create one or more IAM (Identity and Access Management) users under your user account. This is a way of enabling specified users to use the HyperStore S3 Service without giving them the S3 access credentials associated with your root account and without giving them the full range of permissions associated with your root account. Note that all service activity by IAM users under your account **will be counted toward your account's service usage**.

Note

* S3 credentials are needed to access the HyperStore IAM Service (and the CMC's IAM functions), and the CMC's default system administrative user -- the user named "admin" -- does not have these credentials by default. For information on creating S3 credentials for the "admin" user so that this user can access the IAM Service, see [Default 'admin' User Does Not Have S3 Access Credentials](#).

* IAM users created by the default "admin" user cannot create buckets or perform other S3 operations. Such IAM users can only perform [role-based administrative operations](#), if granted permissions by the "admin" user.

* IAM users created by a system administrator other than the default "admin" user can create buckets and perform other S3 operations, as well as performing role-based administrative operations, if granted permissions by the system administrator.

6.9.1.1. Add an IAM User

Note When you create a new IAM user:

* The IAM user by default has no S3 service access or permissions. After creating an IAM user, generate an S3 access key for the user and grant the user whatever S3 permissions you wish to grant them.

* IAM users are not allowed to log into the CMC . IAM users will need to use an S3 client application other than the CMC to access the HyperStore S3 Service .

To create a new IAM user under your HyperStore user account:

1. In the **Manage IAM User** page, click **Add New User**. This opens the panel for adding a new IAM user.

2. In the new user panel, complete the user information:

User Name (mandatory)

- Only letters, numbers, dashes, and underscores are allowed.
- Each IAM user under a single parent HyperStore user account must have a unique IAM user name.

Path (optional)

- The path is an optional way of identifying the user's location within an organizational structure. For example you might specify a path such as `/MyCompany/London/`.
- Specifying a path **does not join the user into an IAM group** -- you must do that as a separate action regardless of whether you specify a path for the user or not -- and has no impact on the user's privileges. It's simply a way of identifying the user's location within an organization.
- Leave this field blank if you don't want to use a path for the user.

3. Click **Save**.

6.9.1.2. Select an IAM User to Work With

In the **Manage IAM User** page the IAM users that you've created will be listed in alphabetical order, with up to 10 users displayed at a time. If you have more than 10 users you can click "Next" in the lower right to see additional users. Alternatively, if you have a large number of users, you may want to use the "Search" field to retrieve a single IAM user (enter the exact user name in the Search field) or a filtered list of users (enter a text string such as a prefix or suffix that you've used in user names or paths).

Once the desired user is displayed in the list, **click the user name**. This opens the **Manage IAM User detail page** for the user.

The screenshot shows the 'Manage IAM User' page for user 'IamUser1'. At the top, there's a back arrow and the title 'Manage IAM User'. To the right is an 'EDIT USER' link. Below the title, the 'User Name' is 'IamUser1' and the 'Path' is '/'. There are three tabs: 'IAM ACCESS KEY' (selected), 'IAM POLICIES', and 'IAM GROUPS'. Below the tabs is a '+ CREATE NEW KEY' button. On the left, there's a 'Show 10 entries' dropdown. On the right, there's a search field labeled 'Search: by access key id'. Below these is a table with columns: 'Access Key ID', 'Status', 'Secret Access Key', and 'Actions'. The table is empty, with a message 'No data available in table' in the center. At the bottom left, it says 'Showing 0 to 0 of 0 entries'. At the bottom right, there are 'Previous' and 'Next' links.

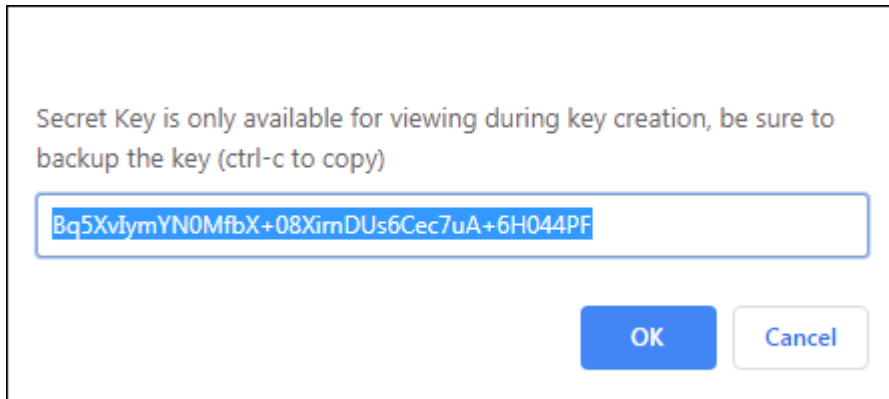
From this user detail page you can ["Manage an IAM User's S3 Access Keys"](#) (page 437) or ["Manage an IAM User's Group Membership"](#) (page 438) or ["Manage an IAM User's Permissions"](#) (page 440).

Note You can also edit the user's user name or path, by clicking **Edit User**.

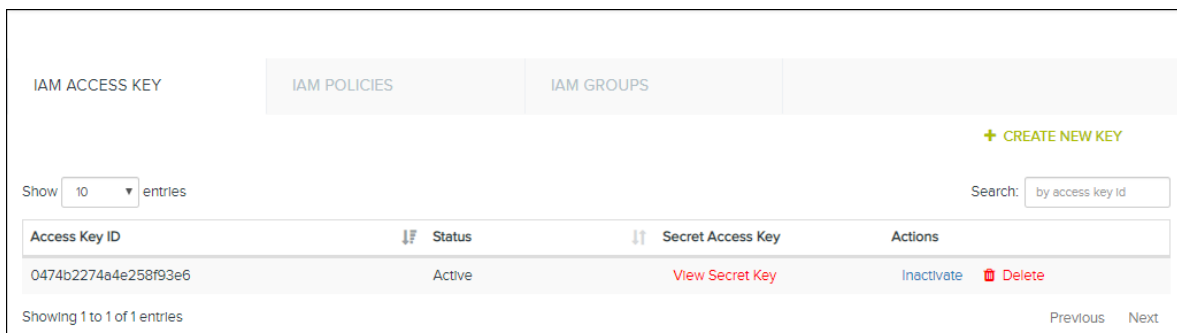
6.9.1.3. Manage an IAM User's S3 Access Keys

When you create a new IAM user, the user does not yet have any S3 access keys. The user needs an S3 access key (access key ID and corresponding secret key) to be able to use the HyperStore S3 Service .

In the lower half of the user's [Manage IAM User detail page](#), the **IAM Access Key** tab displays by default. To create a new S3 access key for the user, click **Create New Key**. You will then be shown the newly created secret key.



Copy (ctrl-c) the secret key and put it in a secure location. Then click **Done**. The access key ID will then display at the bottom of the Manage IAM User detail page.



IMPORTANT ! Make sure to securely store the secret key if you have not already. If you refresh the page or if you leave the page and then return to it, you will no longer be able to view the secret key. You will only be able to view the access key ID.

By system default each IAM user is allowed a maximum of two S3 access keys (each of which is comprised of an access key ID and a corresponding secret key).

Along with creating keys, the **IAM Access Key** tab supports these actions

- To make an access key inactive, in the "Actions" column for the key click **Inactivate**. Note that inactive keys count toward the maximum of two keys that each IAM user is allowed.

Note If an IAM user has no active access key he or she will not be able to access the Hyper-Store S3 Service . Deactivating all of an IAM user's keys can be a means for you to temporarily suspend the IAM user if you wish.

- To activate an inactive access key, in the "Actions" column for the key click **Activate**.
- To delete an access key , in the "Actions" column for the key click **Delete**.

6.9.1.4. Manage an IAM User's Group Membership

An IAM user can belong to one or more IAM groups that you have created. When an IAM user belongs to a group the user inherits the permissions associated with that group. (For more information on permissions see

["Manage an IAM User's Permissions" \(page 440\)](#). You can manage a user's membership in a group either from the user's [Manage IAM User detail page](#) or from the group's Manage IAM Group detail page.

To manage a user's group membership from the user's Manage IAM User detail page, select the **IAM Groups** tab.

The screenshot shows the 'Manage IAM User' interface. At the top, there's a back arrow and the title 'Manage IAM User'. On the right, there's an 'EDIT USER' link. Below the title, the 'User Name' is 'IamUser1' and the 'Path' is '/'. There are three tabs: 'IAM ACCESS KEY', 'IAM POLICIES', and 'IAM GROUPS' (which is selected and circled in red). Below the tabs, there's a '+ ADD TO GROUP' button. A search bar is labeled 'Search: by name or path'. Below that is a table with columns 'Group Name', 'Path', and 'Actions'. The table is currently empty, showing 'No data available in table'. At the bottom, it says 'Showing 0 to 0 of 0 entries' and has 'Previous' and 'Next' links.

To add the user to an IAM group that you have created under your HyperStore account, click **Add to Group**. Then start typing in the "Name of the IAM Group" field and select the group from the drop-down list that displays (the list is based on a fuzzy match of the character[s] that you've typed -- unless you enclose your entry in quotes in which case exact matching is used). After selecting a group from the list click **Add**

The screenshot shows a dialog box titled 'Select The Group To Add The User To: *'. It has a search input field with a dropdown arrow. Below the input field, a list of groups is shown: 'IamGroupA' and 'IamGroupB'. At the bottom right of the dialog, there are two buttons: 'CANCEL' and 'ADD'.

The group will then appear in the list of groups that the user belongs to.

The screenshot shows the 'Manage IAM User' interface with the 'IAM GROUPS' tab selected. The '+ ADD TO GROUP' button is still present. The search bar is labeled 'Search: by name or path'. The table now contains one entry: 'IamGroupA' with a path of '/'. In the 'Actions' column for this entry, there is a red trash icon and the text 'Remove User from Group'. At the bottom, it says 'Showing 1 to 1 of 1 entries' and has 'Previous' and 'Next' links.

To remove the user from an IAM group: In the list of groups that the user belongs to, under the "Actions" column for the group click **Remove User from Group**.

Note By default configuration HyperStore allows an IAM user to belong to a maximum of 10 IAM groups at a time. See the configuration setting "[iam.user.maxGroups](#)" (page 796).

6.9.1.5. Manage an IAM User's Permissions

IAM users by default have no permissions -- they by default are not allowed to perform any of the actions associated with the HyperStore S3 Service (such as creating buckets or uploading and reading objects) or any of the actions associated with the HyperStore IAM Service (such as creating other IAM users or groups). An IAM user gains permissions only by way of IAM "policies" that are either connected to a group that the user belongs to or connected directly to the user.

The most common way to manage IAM user permissions is to connect policies to IAM groups, and IAM users then inherit the permissions associated with whichever IAM groups they belong to. This is the most efficient way to manage IAM user permissions, particularly if you have many IAM users. For information on connecting policies to IAM groups, see "[Manage IAM Group](#)" (page 442).

The system also supports connecting one or more policies directly to an individual IAM user, as described below. If multiple policies are connected with a user -- either directly or by way of the user's group membership(s) -- then the user gains the combined permissions associated with the multiple policies. If there are conflicts within the policies such that one policy allows a certain action and the other policy denies permission to that same action, the "deny" takes precedence and the user is not allowed to perform that action.

When adding a policy directly to a user, either you can attach an existing **managed policy** (an reusable policy that you've already created in the [Manage IAM Policy](#) page) to the user or you can create an **inline policy** for the user (a policy specifically for this user). Typically managed policies are a more efficient way to grant permissions to your IAM groups and users. However, an inline policy may be appropriate if you want to create a policy just for a specific user and be certain that no other group or user will ever use that policy.

To add a policy to an IAM user:

1. In the user's [Manage IAM User detail page](#), with the **IAM Policies** tab selected, click **Add IAM Policy**. This opens the Add IAM Policy panel.

2. In the Add IAM Policy panel you can either:
 - Attach an existing managed policy to the user, by selecting "Managed Policy", then start typing in the "Managed Policy Name" field and select the policy from the drop-down list that displays (the list is based on a fuzzy match of the character[s] that you've typed -- unless you enclose your

entry in quotes in which case exact matching is used). Then click **Add**.

- Create a new inline policy for the user, by selecting "Inline Policy", giving a name to the policy, and then clicking in the "Policy Document" field to open the **Create Policy** editor. For information about working with the **Create Policy** editor, see Step 4 from ["Add an IAM Managed Policy" \(page 455\)](#). Although that step is from the managed policy creation instructions, the functionality of the **Create Policy** editor is the same if you're creating an inline policy.

After you've attached a managed or inline policy to the user, the policy's name will appear in the list of policies that are currently attached to the user, and the user will gain the permissions defined by the policy.

Manage IAM User

[EDIT USER](#)

User Name iamUser1

Path /

IAM ACCESS KEY **IAM POLICIES** **IAM GROUPS**

[+ ADD IAM POLICY](#)

Show 10 entries Search: by name

Policy Type	Policy Name	Policy Document	Actions
Managed Policy	S3_Read_Only	View Document	Detach from User

Showing 1 to 1 of 1 entries Previous Next

Editing an Inline Policy for a User

In a user's [Manage IAM User detail page](#), with the **IAM Policies** tab selected, a list of the policies currently connected to the user displays at the bottom of the page. To edit an inline policy, to the right of the policy name click **View Document**. The existing policy document will then be shown in an **IAM Policy Document Detail** page.

IAM Policy Document Detail

Policy Name
HR-Bucket-Access

Policy Document

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

[Cancel](#) [Save](#)

To edit the policy, click in the "Policy Document" field, make your edits to the policy document, then click **Save**. The user will have her permissions updated in accordance with the policy change.

Removing a Policy from a User

In a user's [Manage IAM User detail page](#), with the **IAM Policies** tab selected, a list of the policies currently connected to the user displays at the bottom of the page. To remove a policy from the user, in the policy's **Actions** column click **Detach from user** (for a managed policy) or **Delete from user** (for an inline policy).

When you remove a policy from an IAM user, the user loses whatever permissions had been defined in the removed policy.

6.9.1.6. Delete an IAM User

This procedure is for deleting an IAM user. Note first that:

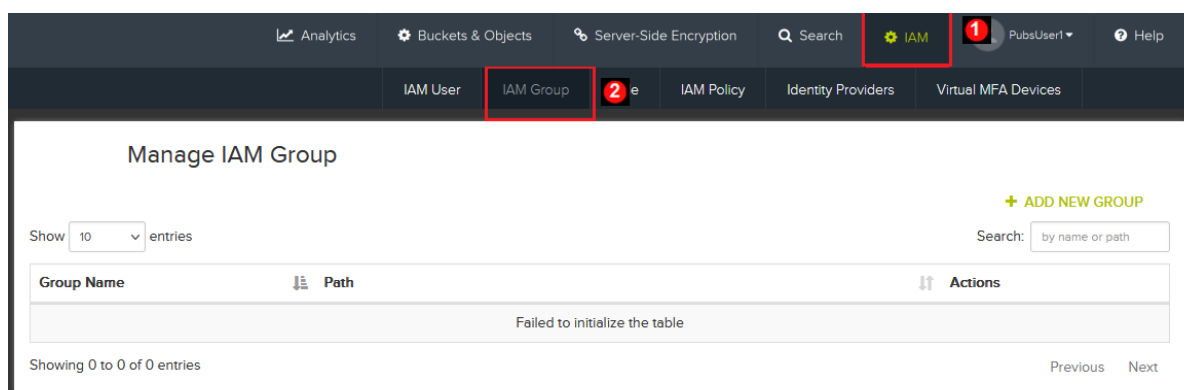
- **Before deleting an IAM user** you must:
 - [Delete the user's S3 credentials](#)
 - [Remove the user from any IAM groups](#) that she is in
 - [Delete any inline policies and detach any managed policies](#) from the user
- Deleting an IAM user will not delete the user's stored S3 buckets and objects from the system. The S3 data is considered to belong to the parent HyperStore user account.

To delete an IAM user:

1. In the **Manage IAM User** page, if the user that you want to delete is not among those listed when you first access the page, retrieve the user by typing the user name in the "Search" field.
2. In the "Actions" column for the user that you want to delete, click **Delete**.
3. When prompted by the system, confirm that you want to delete the user.

6.9.2. Manage IAM Group

Path: **IAM** → **IAM Group**



Supported tasks:

- ["Add an IAM Group" \(page 443\)](#)
- Work With an Existing IAM Group
 - ["Select an IAM Group to Work With" \(page 444\)](#)
 - ["Manage an IAM Group's User Membership" \(page 445\)](#)
 - ["Manage an IAM Group's Permissions" \(page 446\)](#)
- ["Delete an IAM Group" \(page 448\)](#)

You can create one or more IAM groups under your HyperStore user account, as a convenient way to grant permissions to your IAM users. Any of your IAM users that you add to an IAM group's membership will inherit the S3 and IAM permissions that you grant to the IAM group.

Note S3 credentials are needed to access the HyperStore IAM Service (and the CMC's IAM functions), and the CMC's system administrative user named "admin" does not have these credentials by default. For information on creating S3 credentials for the "admin" user so that this user can access the IAM Service, see [Default 'admin' User Does Not Have S3 Access Credentials](#).

6.9.2.1. Add an IAM Group

Note When you create a new IAM group, the IAM group by default has no S3 Service permissions. After creating a group you can grant permissions to the group, and join users into the group.

To create a new IAM group under your HyperStore user account:

1. In the **Manage IAM Group** page, click **Add New Group**. This opens the panel for adding a new IAM group.

The screenshot shows a web interface titled "Manage IAM Group". In the top right corner, there is a green button with a plus icon and the text "ADD NEW GROUP". Below this, there are two input fields: "Group Name" (marked with a red asterisk) and "Path". The "Group Name" field has a placeholder text "Group Name" and the "Path" field has a placeholder text "Path". At the bottom right of the panel, there are two buttons: a grey "CANCEL" button and a green "SAVE" button.

2. In the new group panel, complete the group information:

Group Name (mandatory)

- Only letters, numbers, dashes, and underscores are allowed.
- Each IAM group under a single parent HyperStore user account must have a unique IAM group name.

Path (optional)

- The path is an optional way of identifying the group's location within an organizational structure. For example if the group name is "Quality_Assurance" you might specify a path such as "/MyCompany/Engineering/" (if the QA group is part of the Engineering division of your

company).

- The path has no impact on group privileges or on which users can be put in the group. It's simply a way to help you identify the group within a broader organizational context, if you choose to do so.
- Leave this field blank if you don't want to use a path for the group.

3. Click **Save**.

Note By default configuration HyperStore allows a maximum of 300 IAM groups to exist at a time, per account root. See the configuration setting "**iam.maxGroups**" (page 792).

6.9.2.2. Select an IAM Group to Work With

In the **Manage IAM Group** page the IAM groups that you've created will be listed in alphabetical order, with up to 10 groups displayed at a time. If you have more than 10 groups you can click "Next" in the lower right to see additional groups. Alternatively, if you have a large number of groups, you may want to use the "Search" field to retrieve a single IAM group (enter the exact group name in the Search field) or a filtered list of groups (enter a text string such as a prefix or suffix that you've used in group names or paths).

Once the desired group is displayed in the list, **click the group name**. This opens the **Manage IAM Group detail page** for the group.

The screenshot shows the 'Manage IAM Group' interface. At the top, there's a back arrow and the title 'Manage IAM Group'. On the right, there's an 'EDIT GROUP' link. Below the title, there are two input fields: 'Group Name' with the value 'IamGroupA' and 'Path' with the value '/'. Underneath these fields are two tabs: 'IAM POLICIES' (active) and 'IAM USERS'. To the right of the tabs is an 'ADD IAM POLICY' link. Below the tabs, there's a 'Show' dropdown set to '10' and a 'Search' field with the placeholder 'by name'. At the bottom, there's a table with four columns: 'Policy Type', 'Policy Name', 'Policy Document', and 'Actions'. The table is currently empty, with a message 'No data available in table' in the center. At the very bottom, there's a status bar showing 'Showing 0 to 0 of 0 entries' and 'Previous' and 'Next' navigation links.

From this page you can:

- ["Manage an IAM Group's User Membership" \(page 445\)](#)
- ["Manage an IAM Group's Permissions" \(page 446\)](#).

Note You can also edit the group's group name or path, by clicking **Edit Group**.

6.9.2.3. Manage an IAM Group's User Membership

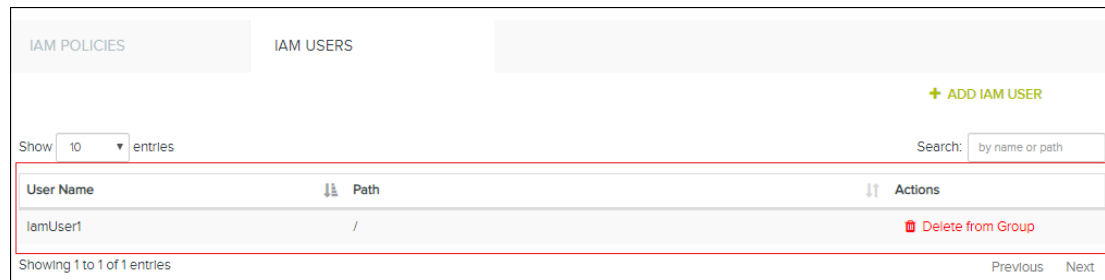
An IAM user can be a member of one or more IAM groups. When an IAM user belongs to a group the user inherits the permissions associated with that group. (For more information on permissions see ["Manage an IAM Group's Permissions" \(page 446\)](#)). You can manage a user's membership in a group either from the group's [Manage IAM Group detail page](#) or from the user's [Manage IAM User detail page](#).

To manage a group's user membership from the group's Manage IAM Group detail page:

1. Select the **IAM Users** tab.

2. To add a user to the IAM group, click **Add IAM User**. Then start typing in the "Name of the IAM User" field and select the user from the drop-down list that displays (the list is based on a fuzzy match of the character[s] that you've typed -- unless you enclose your entry in quotes in which case exact matching is used).

3. After selecting a user from the list click **Add**. The user will then appear in the list of users that belong to the group.



To remove a user from the IAM group: In the list of users that belong to the group, under the "Actions" column for the user click **Delete from Group**.

Note By default configuration HyperStore allows an IAM user to belong to a maximum of 10 IAM groups at a time. See the configuration setting "[iam.user.maxGroups](#)" (page 796).

6.9.2.4. Manage an IAM Group's Permissions

The purpose of having an IAM group is to grant permissions to the group and have those permissions be inherited by all of the users in the group. This is an efficient way of granting permissions to IAM users, rather than granting permissions to one user at a time.

For each IAM group you can connect one or more IAM "policies", with each policy defining a particular set of permissions. If multiple policies are connected to a group, then the members of the group get the combined permissions associated with the multiple policies. If there are conflicts within the connected policies such that one policy allows a certain action and the other policy denies permission to that same action, the "deny" takes precedence and users are not allowed to perform that action.

Note IAM users by default have no permissions. They gain only those permissions that are explicitly granted by IAM policies that are connected to the user or to the group(s) to which the user belongs.

When adding a policy to a group, either you can attach an existing **managed policy** (an reusable policy that you've already created in the [Manage IAM Policy](#) page) to the group or you can create an **inline policy** for the group (a policy specifically for this group). Typically managed policies are a more efficient way to grant permissions to your IAM groups and users. However, an inline policy may be appropriate if you want to create a policy just for a specific group and be certain that no other group will ever use that policy.

To add a policy to an IAM group:

1. In the group's [Manage IAM Group detail page](#), with the **IAM Policies** tab selected, click **Add IAM Policy**. This opens the Add IAM Policy panel.

2. In the Add IAM Policy panel you can either:

- Attach an existing managed policy to the group, by selecting "Managed Policy", then start typing in the "Managed Policy Name" field and select the policy from the drop-down list that displays (the list is based on a fuzzy match of the character[s] that you've typed -- unless you enclose your entries in quotes in which case exact matching is used). Then click **Add**.
- Create a new inline policy for the group, by selecting "Inline Policy", giving a name to the policy, and then clicking in the "Policy Document" field to open the **Create Policy** editor. For information about working with the **Create Policy** editor, see Step 4 from "[Add an IAM Managed Policy](#)" (page 455). Although that step is from the managed policy creation instructions, the functionality of the **Create Policy** editor is the same if you're creating an inline policy.

After you've attached a managed or inline policy to the group, the policy's name will appear in the list of policies that are currently attached to the group, and users in the group will gain the permissions defined by the policy.

Policy Type	Policy Name	Policy Document	Actions
Managed Policy	S3_Read_Only	View Document	Detach from Group

Editing an Inline Policy for a Group

In a group's [Manage IAM Group detail page](#), with the **IAM Policies** tab selected, a list of the policies currently connected to the group displays at the bottom of the page. To edit an inline policy, to the right of the policy name click **View Document**. The existing policy document will then be shown in an **IAM Policy Document Detail** page.

IAM Policy Document Detail

Policy Name
HR-Bucket-Access

Policy Document

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

Cancel Save

To edit the policy, click in the "Policy Document" field, make your edits to the policy document, then click **Save**. Users in the group will have their permissions updated in accordance with the policy change.

Removing a Policy from a Group

In a group's [Manage IAM Group detail page](#), with the **IAM Policies** tab selected, a list of the policies currently connected to the group displays at the bottom of the page. To remove a policy from the group, in the policy's **Actions** column click **Detach from group** (for a managed policy) or **Delete from group** (for an inline policy).

When you remove a policy from an IAM group, the group's members lose whatever permissions had been defined in the removed policy.

6.9.2.5. Delete an IAM Group

Before deleting an IAM group you must:

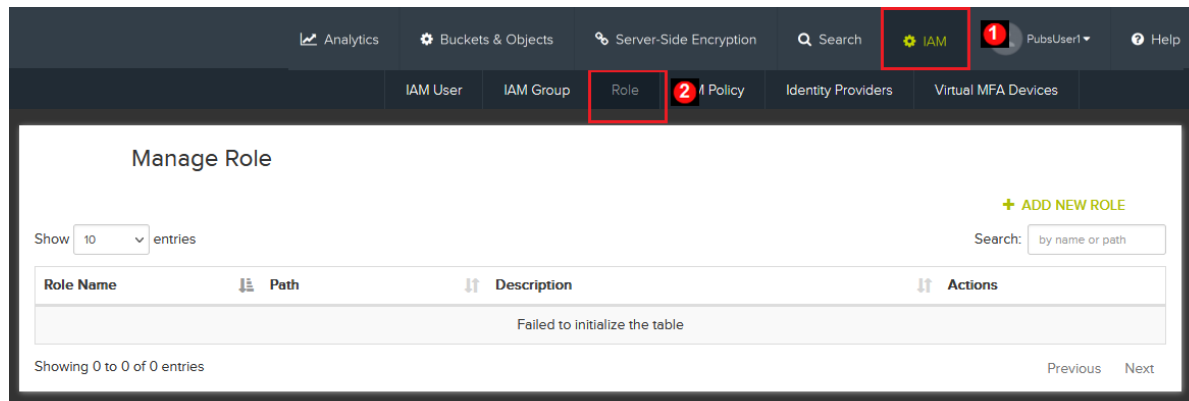
- [Remove any IAM users](#) who are currently in the group. You cannot delete an IAM group that currently has IAM users in it. You do not need to delete the IAM users from the system -- you only need to remove them from the group.
- [Delete any inline policies and detach any managed policies](#) from the group.

To delete an IAM group:

1. In the **Manage IAM Group** page, if the group that you want to delete is not among those listed when you first access the page, retrieve the group by typing the group name in the "Search" field.
2. In the "Actions" column for the group that you want to delete, click **Delete**.
3. When prompted by the system, confirm that you want to delete the group.

6.9.3. Manage IAM Role

Path: **IAM** → **Role**



Supported tasks:

- ["Add an IAM Role" \(page 449\)](#)
- Work With an Existing IAM Role
 - ["Select an IAM Role to Work With" \(page 451\)](#)
 - ["Manage an IAM Role's Permissions" \(page 452\)](#)
 - ["View or Edit an IAM Role's Trust Policy" \(page 454\)](#)
 - ["Change the Maximum Session Duration of an IAM Role" \(page 454\)](#)
- ["Delete an IAM Role" \(page 454\)](#)

You can create one or more IAM roles under your HyperStore user account. When creating a role you can use a SAML provider as a principal in the role's trust policy. The trust policy enables federated users who successfully sign in using the SAML identity provider to assume the IAM role.

Creating an IAM role (as described below) is one part of the procedure for setting up access to the HyperStore S3 Service by persons authenticated by an external SAML identity provider. For the full procedure see [SAML Support](#). Note that as part of that procedure you should create a SAML identity provider IAM profile (as described in ["Manage Identity Providers" \(page 462\)](#)) before creating an IAM role that specifies that identity provider in the role's trust policy.

Note S3 credentials are needed to access the HyperStore IAM Service (and the CMC's IAM functions), and the CMC's system administrative user named "admin" does not have these credentials by default. For information on creating S3 credentials for the "admin" user so that this user can access the IAM Service, see [Default 'admin' User Does Not Have S3 Access Credentials](#).

6.9.3.1. Add an IAM Role

To create a new IAM role under your HyperStore user account:

1. In the **Manage IAM Role** page, click **Add New Role**. This opens the panel for adding a new IAM role.

Manage Role + ADD NEW ROLE

Role Name * Path

Description

Assume Role Policy Document *

- In the new role panel, complete the role information:

Role Name (mandatory)

- Only letters, numbers, dashes, and underscores are allowed.
- Each IAM role under a single parent HyperStore user account must have a unique IAM role name.

Path (optional)

- The path is an optional way of identifying the role's location within an organizational structure. For example if the role name is "Tester" you might specify a path such as "/MyCompany/Engineering/" (if the Tester role is going to be assumed by member's of the Engineering division of your company).
- The path has no impact on the role's privileges or on which users can assume the role. It's simply a way to help you identify the role within a broader organizational context, if you choose to do so.
- Leave this field blank if you don't want to use a path for the role.

Description (optional)

Description of the role.

Assume Role Policy Document (also known as Trust Policy) (mandatory)

The trust relationship policy document that grants an entity permission to assume the role. This must be a properly formatted JSON document. No visual editor is available to create this document here, so you must either type in the policy content -- if you are knowledgeable about JSON formatting and trust policy composition -- or else paste in a properly formatted trust policy document.

For an example trust policy document see ["Example Assume Role Policy \(Trust Policy\)" \(page 450\)](#) below.

- Click **Save**.

Example Assume Role Policy (Trust Policy)

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"/>
```

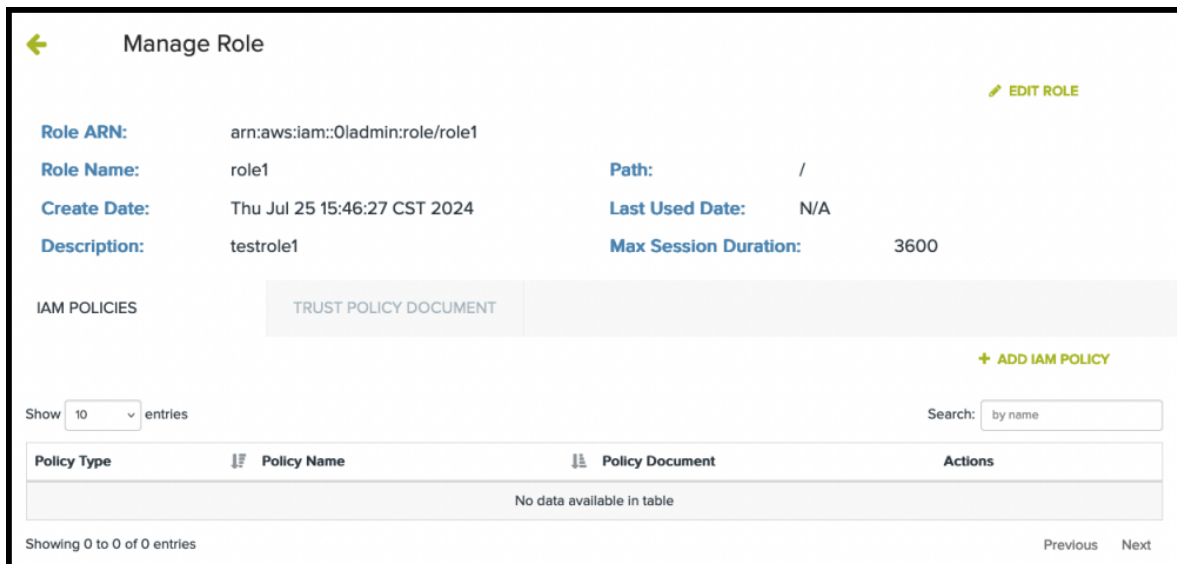
For *Condition* in a trust policy, HyperStore supports only the *StringEquals* condition operator and only the following condition keys:

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

6.9.3.2. Select an IAM Role to Work With

In the **Manage IAM Role** page the IAM roles that you've created will be listed in alphabetical order, with up to 10 roles displayed at a time. If you have more than 10 roles you can click "Next" in the lower right to see additional roles. Alternatively, if you have a large number of roles, you may want to use the "Search" field to retrieve a single IAM role (enter the exact role name in the Search field) or a filtered list of roles (enter a text string such as a prefix or suffix that you've used in role names or paths).

Once the desired role is displayed in the list, **click the role name**. This opens the **Manage IAM Role detail page** for the role.



From this page you can:

- ["Manage an IAM Role's Permissions"](#) (page 452)
- ["View or Edit an IAM Role's Trust Policy"](#) (page 454)
- ["Change the Maximum Session Duration of an IAM Role"](#) (page 454)

6.9.3.3. Manage an IAM Role's Permissions

The purpose of having an IAM role is to grant permissions to the role and have those permissions be inherited by persons or applications that assume the role.

For each IAM role you can connect one or more IAM "policies", with each policy defining a particular set of permissions. If multiple policies are connected to a role, then persons or applications that assume the role get the combined permissions associated with the multiple policies. If there are conflicts within the connected policies such that one policy allows a certain action and the other policy denies permission to that same action, the "deny" takes precedence and entities assuming the role are not allowed to perform that action.

Note IAM roles by default have no permissions. They gain only those permissions that are explicitly granted by IAM policies that are connected to the role.

When adding a policy to a role, either you can attach an existing **managed policy** (an reusable policy that you've already created in the [Manage IAM Policy](#) page) to the role or you can create an **inline policy** for the role (a policy specifically for this role). Typically managed policies are a more efficient way to grant permissions to your IAM groups and users and roles. However, an inline policy may be appropriate if you want to create a policy just for a specific role and be certain that no other entity will ever use that policy.

To add a policy to an IAM role:

1. In the role's [Manage IAM Role detail page](#), with the **IAM Policies** tab selected, click **Add IAM Policy**. This opens the Add IAM Policy panel.

2. In the Add IAM Policy panel you can either:

- Attach an existing managed policy to the role, by selecting "Managed Policy", then start typing in the "Managed Policy Name" field and select the policy from the drop-down list that displays (the list is based on a fuzzy match of the character[s] that you've typed -- unless you enclose your entry in quotes in which case exact matching is used). Then click **Add**.
- Create a new inline policy for the role, by selecting "Inline Policy", giving a name to the policy, and then clicking in the "Policy Document" field to open the **Create Policy** editor. For information about working with the **Create Policy** editor, see Step 4 from ["Add an IAM Managed Policy" \(page 455\)](#). Although that step is from the managed policy creation instructions, the functionality of the **Create Policy** editor is the same if you're creating an inline policy.

After you've connected a managed or inline policy to the role, the policy's name will appear in the list of policies that are currently connected to the role, and persons or applications that assume the role will gain the permissions defined by the policy.

Policy Type	Policy Name	Policy Document	Actions
Managed Policy	S3_read_write	View Document	Detach from Role

Editing an Inline Policy for a Role

In the role's [Manage IAM Role detail page](#), with the **IAM Policies** tab selected, a list of the policies currently connected to the role displays at the bottom of the page. To edit an inline policy, to the right of the policy name click **View Document**. The existing policy document will then be shown in an **IAM Policy Document Detail** page.

To edit the policy, click in the "Policy Document" field, make your edits to the policy document, then click **Save**.

Removing a Policy from a Role

In the role's [Manage IAM Role detail page](#), with the **IAM Policies** tab selected, a list of the policies currently connected to the role displays at the bottom of the page. To remove a policy from the role, in the policy's **Actions** column click **Detach from role** (for a managed policy) or **Delete from role** (for an inline policy).

6.9.3.4. View or Edit an IAM Role's Trust Policy

To view an IAM role's current trust policy, in the role's [Manage IAM Role detail page](#) click the **Trust Policy Document** tab. Then if you wish to edit the trust policy click **Edit Trust Policy**, make your edits, and click **Save**.

6.9.3.5. Change the Maximum Session Duration of an IAM Role

By default, an authenticated session of a person or application that assumes an IAM role -- a session during which the entity assuming the role is granted temporary security credentials and can perform the operations allowed by the policy or policies connected to the role -- has a maximum duration of 1 hour. If you wish you can increase this maximum session duration. To do so in the role's [Manage IAM Role detail page](#) click **Edit Role** and then set a new value for the "Max Session Duration", as a number of seconds. The minimum is 3600 seconds (one hour) and the maximum is 43200 seconds (twelve hours). After making your change click **Save**.

6.9.3.6. Delete an IAM Role

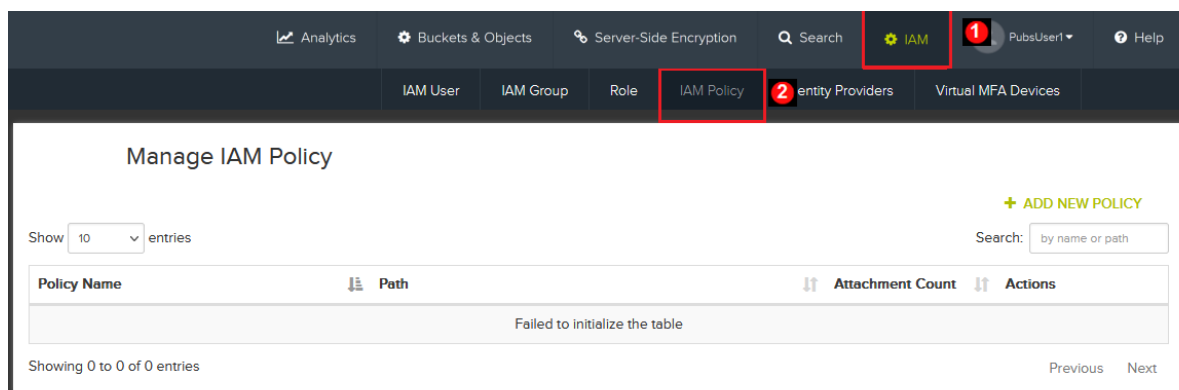
Before deleting an IAM role you must [delete any inline policies and detach any managed policies](#) from the role.

To delete an IAM role:

1. In the **Manage IAM Role** page, in the "Actions" column for the role that you want to delete, click **Delete**.
2. When prompted by the system, confirm that you want to delete the role.

6.9.4. Manage IAM Policy

Path: **IAM** → **IAM Policy**



Supported tasks:

- ["Add an IAM Managed Policy" \(page 455\)](#)
- Work With an Existing IAM Managed Policy
 - ["Select an IAM Managed Policy to Work With" \(page 457\)](#)
 - ["Attach or Detach an IAM Managed Policy to Groups, Users, and Roles" \(page 458\)](#)
 - ["Change an IAM Managed Policy By Adding a New Version" \(page 460\)](#)
- ["Delete an IAM Managed Policy" \(page 461\)](#)

The **Manage IAM Policy** page is for creating and working with IAM **managed policies** -- not inline policies:

- An IAM **managed policy** is an independent, reusable policy that you can attach to multiple IAM groups, users, and/or roles. In most circumstances managed policies are the best way to grant permissions to your IAM groups, users, and roles.
- An IAM **inline policy** is a policy that you create for one specific IAM group, user, or role. The policy exists solely as a part of that group's profile or that user's profile or that role's profile, and cannot be applied to any other groups or users or roles. The one circumstance where you might want to create an inline policy is if you want to be certain that the policy is used only for one particular group, user, or role, and not for any other groups, users, or roles. To create an inline policy you use the [Manage IAM User](#) page or the [Manage IAM Group](#) page or the [Manage Role](#) page -- not the **Manage IAM Policy** page.

Note S3 credentials are needed to access the HyperStore IAM Service (and the CMC's IAM functions), and the CMC's system administrative user named "admin" does not have these credentials by default. For information on creating S3 credentials for the "admin" user so that this user can access the IAM Service, see [Default 'admin' User Does Not Have S3 Access Credentials](#).

6.9.4.1. Add an IAM Managed Policy

Note When you create a new IAM managed policy, the policy by default is not attached to any of your IAM groups or users or roles. After creating a policy you can attach it to your IAM groups or users or roles.

To create a new IAM managed policy under your HyperStore user account:

1. In the **Manage IAM Policy** page, click **Add New Policy**. This opens the panel for adding a new IAM managed policy.

Manage IAM Policy + ADD NEW POLICY

Policy Name *

Path

Policy Description

Policy Document

2. In the new policy panel, enter a Policy Name for the policy. Only letters, numbers, dashes, and under-scores are allowed in the policy name, and the name must be unique within your HyperStore user account. Optionally you can also enter a Path (to identify the policy's location within an organizational structure -- for example `"/MyCompany/Engineering/"`) and/or a Policy Description.
3. Next, click in the "Policy Document" field. This opens the **Create Policy** editor.

Create Policy ×

VISUAL EDITOR JSON

SERVICE S3 Switch to IAM

EFFECT ✔ Allow

ACTIONS ☐ All S3 actions(s3:*)

- ▶ List
- ▶ Read
- ▶ Write
- ▶ Permissions management

RESOURCES ☒ All resources (selected actions support all resources)

+ ADD ADDITIONAL PERMISSION

4. Use the **Create Policy** editor to specify the permissions that will comprise this policy.
 - The editor provides both a Visual Editor tab (appropriate for most users creating a policy) and a JSON editor tab (for power users who prefer to write a policy directly with JavaScript Object Notation).

Note If you want to create an IAM policy that includes permissions to perform HyperStore administrative actions, you must use the JSON editor rather than the Visual Editor. For more information on HyperStore admin permissions see [Role-Based Access to Admin API Operations](#).

- In the Visual Editor:
 - By using the "Switch to [S3/IAM]" toggle in the upper right of the Visual Editor you can choose to define S3 action permissions (such as creating buckets and uploading objects) or IAM action permissions (such as creating IAM groups and users). More commonly you would be defining S3 permissions, but the interface does support defining IAM permissions if you desire.

Because the HyperStore S3 and IAM actions listed in the editor are compatible with the Amazon Web Services (AWS) S3 and IAM APIs, you can refer to AWS online documentation if you need a definition of a particular action:

- [Actions, Resources, and Condition Keys for Amazon S3](#)
- [Actions, Resources, and Condition Keys for Identity And Access Management](#)

- By clicking "Add Additional Permission" at the bottom of the Visual Editor you can configure multiple permission "statements" within a single policy. Each statement:
 - Can be directed toward either S3 actions or IAM actions. (You cannot combine S3 permissions and IAM permissions within the same statement. You can, however, combine S3 permission statements and IAM permission statements within the same policy.)
 - Can either Allow or Deny permission to perform the actions that you specify. (Use the Effect: Allow/Deny toggle).
 - Can define a scope of resources (buckets and objects) to which the policy applies. (Use the Resources toggle that lets you apply the statement to all applicable resources -- the default -- or only to a specified resource or resources).

5. When you're done configuring the policy's permission definitions, in the **Create Policy** editor click **Save**. This closes the **Create Policy** editor and returns you to the **Manage IAM Policy** page.

6. Back in the **Manage IAM Policy** page, click **Add**.

The IAM managed policy's name will then appear in the list of your existing policies.

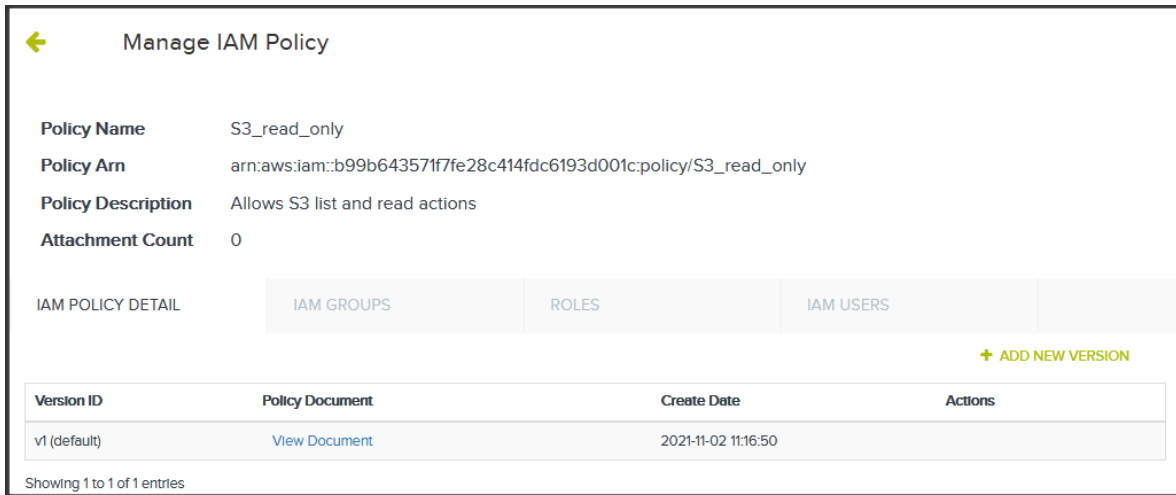
Manage IAM Policy			
Show 10 entries		+ ADD NEW POLICY Search: by name or path	
Policy Name	Path	Attachment Count	Actions
S3_read_only	/	0	Delete
Showing 1 to 1 of 1 entries		Previous Next	

6.9.4.2. Select an IAM Managed Policy to Work With

In the **Manage IAM Policy** page the IAM managed policies that you've created will be listed in alphabetical order, with up to 10 policies displayed at a time. If you have more than 10 policies you can click "Next" in the

lower right to see additional policies. Alternatively, if you have a large number of policies, you may want to use the "Search" field to retrieve a single IAM managed policy (enter the exact policy name in the Search field) or a filtered list of managed policies (enter a text string such as a prefix or suffix that you've used in policy names or paths).

Once the desired managed policy is displayed in the list, **click the policy name**. This opens the **Manage IAM Policy detail page** for the policy.



From this page you can:

- ["Attach or Detach an IAM Managed Policy to Groups, Users, and Roles"](#) (page 458)
- ["Change an IAM Managed Policy By Adding a New Version"](#) (page 460)

6.9.4.3. Attach or Detach an IAM Managed Policy to Groups, Users, and Roles

You can attach an IAM managed policy to one or more of the following:

- An IAM group, in which case all the IAM users who belong to that group will inherit the permissions defined by that IAM managed policy.
- An IAM user, in which case that individual user will inherit the permissions defined by that IAM managed policy.
- An IAM role, in which case users who assume that role will inherit the permissions defined by that IAM managed policy.

If you wish you can attach a managed policy to multiple groups, users, and/or roles. To do this, you will need to execute the attachment process one group or user or role at a time (for example, attach the policy to a group and then attach it to another group and so on).

Once you've [selected an IAM managed policy and opened its policy detail page](#), in the policy detail page you can attach the policy to one or more IAM groups, users, and/or roles. (Alternatively you can also use a group's [Manage IAM Group detail](#) page, a user's [Manage IAM User detail](#) page, or a role's [Manage IAM Role detail](#) page to attach managed policies to a group, user, or role).

To attach an IAM managed policy to a group from the policy's Manage IAM Policy detail page, select the **IAM Groups** tab.

Manage IAM Policy

Policy Name: S3_read_only
 Policy Arn: arn:aws:iam::b99b643571f7fe28c414fdc6193d001c:policy/S3_read_only
 Policy Description: Allows S3 list and read actions
 Attachment Count: 0

Tabs: IAM POLICY DETAIL | **IAM GROUPS** | ROLES | IAM USERS

+ ATTACH TO GROUP

Show 10 entries Search: by name

Group Name	Actions
No data available in table	

Showing 0 to 0 of 0 entries Previous Next

Next, click **Attach to Group**. Then click in the "Select the Group to attach the Policy to" field, and select the group from the drop-down list that displays. The list shows all the IAM groups that have been created under your HyperStore user account that do not currently have this IAM managed policy attached.

Select The Group To Attach The Policy To:

I

- IamGroupA
- IamGroupB

CANCEL ATTACH

If you have many IAM groups under your account and you want to filter the drop-down list, type a text string in the "Select the Group to attach the Policy to" field. This will shorten the drop-down list by limiting it to groups whose group names match against the text string.

After selecting a group from the list click **Attach**. The group will then appear in the list of groups to which this IAM managed policy is attached.

Tabs: IAM POLICY DETAIL | **IAM GROUPS** | ROLES | IAM USERS

+ ATTACH TO GROUP

Show 10 entries Search: by name

Group Name	Actions
IamGroupA	Detach from Group

Showing 1 to 1 of 1 entries Previous Next

To detach the policy from an IAM group: In the list of groups to which the policy is attached, under the "Actions" column for the group click **Detach from Group**.

To attach the policy to a specific IAM user or an IAM role: Follow the instructions above, except in the policy's Manage IAM Policy detail page, select the **IAM Users** tab or the **IAM Roles** tab rather than the **IAM Groups** tab.

Checking an IAM Managed Policy's Current Attachments

At any time you can see which groups, users, and roles an IAM managed policy is currently attached to by accessing the policy's [Manage IAM Policy detail page](#) and then selecting the **IAM Groups** tab or **IAM Users** tab or **IAM Roles** tab.

6.9.4.4. Change an IAM Managed Policy By Adding a New Version

You can modify an existing IAM managed policy, but not by overwriting the policy. Instead, you can create a new version of the policy. You can then retain multiple versions of the policy -- the new version and the older version(s) -- or if you wish you can delete older versions of the policy. If you have multiple versions of a policy, you can choose a version to be the **default version** of the policy. **The default version of a policy is the operative version** -- the default version is the version that the system will apply to groups, users, and roles to which the policy is attached. For example if you create a new version of an existing policy and you set the new version to be the default version, then the system will automatically start applying the new version to all groups, users, and/or roles to which the policy is attached.

To create a new version of an IAM managed policy, first [select the policy and open its policy detail page](#). Then in the policy detail page click **Add New Version**. This opens a panel in which you can create the new version of the policy.

The screenshot shows the 'Manage IAM Policy' interface. At the top, there's a back arrow and the title 'Manage IAM Policy'. Below this, a summary section displays:

- Policy Name:** S3_read_only
- Policy Arn:** arn:aws:iam::b99b643571f7fe28c414fdc6193d001c:policy/S3_read_only
- Policy Description:** Allows S3 list and read actions
- Attachment Count:** 0

 Below the summary is a tabbed interface with 'IAM POLICY DETAIL' selected, and other tabs for 'IAM GROUPS', 'ROLES', and 'IAM USERS'. A red box highlights a green '+ ADD NEW VERSION' button.

The 'Add New Version' panel is open, showing:

- Version ID:** A dropdown menu with 'Select to preload' and a 'Set As Default' checkbox.
- Policy Document:** A large text area with the placeholder text 'Detail of the IAM Policy Document'.
- Buttons:** 'CANCEL' and 'SAVE' buttons at the bottom right.

At the bottom of the interface is a table listing existing policy versions:

Version ID	Policy Document	Create Date	Actions
v1 (default)	View Document	2021-11-02 11:16:50	

Below the table, it says 'Showing 1 to 1 of 1 entries'.

First decide if you want this new version of the policy to become the default version (the operative version that is applied to any IAM groups, users, or roles to which the policy is attached). If you want it to become the default version select **Set As Default** in the upper right of the new policy version panel.

Then, to use an existing version of the policy as the starting point from which to create the new version, click the **Version ID** drop-down and choose a version from the list. Then click in the **Policy Document** field to start your editing. (Alternatively, to create the new policy version from scratch rather than starting from an existing version, ignore the **Version ID** drop-down and just click in the **Policy Document** field.)

Whether starting from an existing version or starting from scratch, once you click in the **Policy Document** field, the document editor opens and you can use either the Visual Editor tab or the JSON tab (for power users) to create the new version of the policy. When you're done constructing the new policy document click **Save** in the lower right of document editor (which closes the editor) and then click **Save** again in the new version creation panel.

You will now see multiple policy versions listed in the policy details page. In the screen shot below, the new version (v2) is the default version.

Manage IAM Policy

Policy Name S3_read_only
Policy Arn arn:aws:iam::b99b643571f7fe28c414fdc6193d001c:policy/S3_read_only
Policy Description Allows S3 list and read actions
Attachment Count 0

IAM POLICY DETAIL | IAM GROUPS | ROLES | IAM USERS

[+ ADD NEW VERSION](#)

Version ID	Policy Document	Create Date	Actions
v2 (default)	View Document	2021-11-02 14:19:05	
v1	View Document	2021-11-02 11:16:50	Delete Set As Default

Showing 1 to 2 of 2 entries

If you wish, you can use the **Actions** column to delete non-default versions of the policy, or to promote a non-default version of the policy to become the default version.

Note In the policy details page you cannot delete the default version of a policy -- you can only delete versions that are not the default version. After you've deleted all non-default versions of a policy you can delete the default version using the method described in ["Delete an IAM Managed Policy"](#) (page 461).

6.9.4.5. Delete an IAM Managed Policy

Before deleting an IAM managed policy you must delete any non-default versions of the policy, and detach the policy from any IAM groups, users, or roles to which it is currently attached. You can perform both of those tasks in the policy's detail page, as described in ["Change an IAM Managed Policy By Adding a New Version"](#) (page 460) and ["Attach or Detach an IAM Managed Policy to Groups, Users, and Roles"](#) (page 458).

After deleting non-default versions of the policy and detaching the policy from any IAM entities to which it was attached, you can do the following to delete the policy:

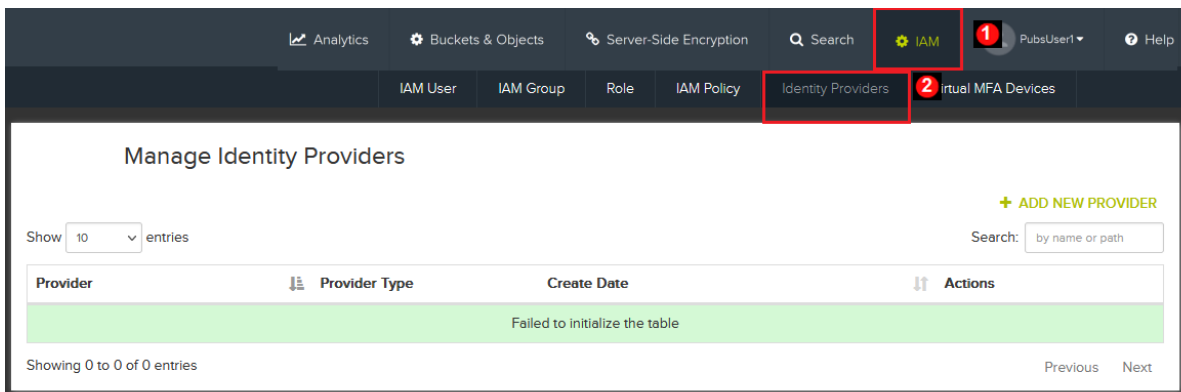
1. In the **Manage IAM Policy** page your existing policies are listed. In the "Actions" column for the policy that you want to delete, click **Delete**.

Note If you have more than 10 policies and the policy that you want to delete is not among those in the alphabetized list when you first access the page, click "Next" to see more policies. Alternatively you can retrieve the policy by entering the policy name in the "Search" field.

2. When prompted, confirm that you want to delete the policy.

6.9.5. Manage Identity Providers

Path: **IAM** → **Identity Providers**



Supported tasks:

- ["Add an Identity Provider"](#) (page 463)
- ["View or Replace an Identity Provider's Metadata Document"](#) (page 463)
- ["Delete an Identity Provider"](#) (page 464)

You can create IAM profiles for one or more SAML identity providers under your HyperStore user account. You can then use a SAML provider as a principal in an [IAM role](#)'s trust policy. The trust policy enables federated users who successfully sign in using the SAML identity provider to assume the IAM role.

Creating an IAM profile for an SAML identity provider (as described below) is one part of the procedure for setting up access to the HyperStore S3 Service by persons authenticated by an external SAML identity provider. For the full procedure see [SAML Support](#).

Note SAML identity providers -- providers that support SAML 2.0 -- are the only supported type of providers. OpenID identity providers are not currently supported.

Note S3 credentials are needed to access the HyperStore IAM Service (and the CMC's IAM functions), and the CMC's system administrative user named "admin" does not have these credentials by default. For information on creating S3 credentials for the "admin" user so that this user can access the IAM Service, see [Default 'admin' User Does Not Have S3 Access Credentials](#).

6.9.5.1. Add an Identity Provider

To create a new identity provider under your HyperStore user account:

1. In the **Manage Identity Providers** page, click **Add New Provider**. This opens the panel for adding a new identity provider.

2. In the new identity provider panel, complete the provider information:

Provider Name (mandatory)

- Only letters, numbers, dashes, and underscores are allowed.
- Each identity provider under a single parent HyperStore user account must have a unique identity provider name.

Provider Type (mandatory)

The only provider type currently supported is SAML. A SAML identity provider is an identity provider that supports SAML 2.0

SAML Metadata Document (mandatory)

Here you must upload the SAML provider's metadata document from your local machine. This is an XML document generated by the identity provider application (and which you've saved to the machine from which you're connecting to the CMC). The metadata document includes the issuer's name, expiration information, and security keys that can be used to validate the SAML authentication response (assertions) that are received from the identity provider, amongst other information about the provider.

3. Click **Save**.

6.9.5.2. View or Replace an Identity Provider's Metadata Document

In the **Manage Identity Provider** page the identity provider profiles that you've created will be listed in alphabetical order. In the list, click the provider name. This opens the **Manage Identity Provider detail page** for the provider.

Manage Identity Providers

Provider	SamlProvider1	Provider Type	SAML
Provider Arn	arn:aws:iam::b99b643571f7fe28c414fdc6193d001c:saml-provider/SamlProvider1		
Create Date	Fri Nov 05 07:47:14 PDT 2021		

IAM PROVIDER METADATA

[+ VIEW XML](#)

Metadata Summary

Issuer URL: testEntityId

Valid Until: 6/27/2020

SSO Service Location: https://localhost:8443/saml/sso

DOWNLOAD **REPLACE**

In this page you can click:

- **View XML** to view the provider's metadata document.
- **Download** to download the provider's metadata document, from the CMC to your local machine.
- **Replace** to replace the provider's current metadata document in the CMC with a different metadata document that you have on your local machine.

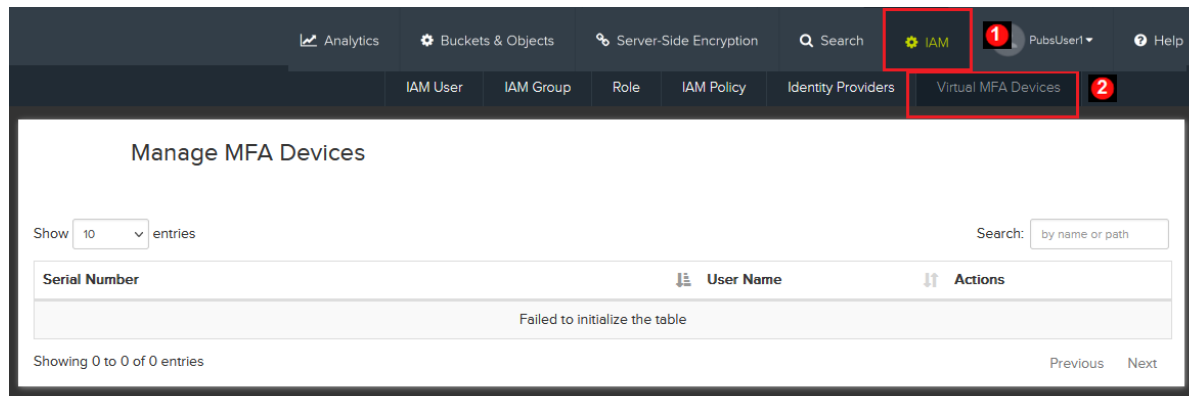
6.9.5.3. Delete an Identity Provider

To delete an identity provider profile:

1. In the **Manage Identity Providers** page, in the "Actions" column for the provider that you want to delete, click **Delete**.
2. When prompted by the system, confirm that you want to delete the provider.

6.9.6. Manage MFA Devices

Path: **IAM** → **Virtual MFA Devices**



Supported tasks:

- List MFA devices that exist under your root account
- Deactivate an MFA device for one of your IAM users for whom the device is currently active
- Delete an MFA device that is not currently active for any user

This CMC page **does not support**:

- Creating MFA devices
- Activating an MFA device for an IAM user or for your root account

IAM users cannot log into the CMC, and the CMC cannot be used to create or activate Multi-Factor Authentication (MFA) devices for your IAM users. However, if there is a third party application that is calling the Hyper-Store IAM Service to create MFA devices under your root account and to activate MFA devices for your IAM users, you can use this CMC page to view a list of those devices. You can also use this page to deactivate a device that's currently active for one of your IAM users, and delete devices that are not currently active for any user.

If you have [set up MFA for your root account](#) -- for your own logins to the CMC -- that MFA device will also be displayed in the list (and marked as the device used by the root account).

Manage MFA Devices

Show 10 entries

Search: by name or path

Serial Number	User Name	Actions
arn:aws:iam::c9a8e2be90e7a6fbfe8ac16dd6f10b30:mfa/123456789012345678901234567890123456...		Delete
arn:aws:iam::c9a8e2be90e7a6fbfe8ac16dd6f10b30:mfa/Ab_=.123		Delete
arn:aws:iam::c9a8e2be90e7a6fbfe8ac16dd6f10b30:mfa/Ab_=.1234	iamuser2	Deactivate
arn:aws:iam::c9a8e2be90e7a6fbfe8ac16dd6f10b30:mfa/Abc_1-2-.=0	iamuser3	Deactivate
arn:aws:iam::c9a8e2be90e7a6fbfe8ac16dd6f10b30:mfa/basic-VMD1	root account	
arn:aws:iam::c9a8e2be90e7a6fbfe8ac16dd6f10b30:mfa/iam-VMD1	iamuser1	Deactivate
arn:aws:iam::c9a8e2be90e7a6fbfe8ac16dd6f10b30:mfa/iam-VMD2		Delete

Showing 1 to 7 of 7 entries

PreviousNext

Note You cannot use this CMC page to deactivate an MFA device that's active for the root account. If you want to deactivate an MFA device that's active for the root account, go to the [Security Credentials](#) page and click where it says Multi-Factor Authentication **Enabled**. Then confirm you want to disable

MFA on the root account. After confirming, you can return to the **Manage MFA Devices** page and delete the deactivated device if you wish.

6.10. Cluster

The **Cluster** tab contains the following functions:

- [Data Centers](#)
- Nodes
 - [Node Status](#)
 - [Node Activity](#)
 - [Node Advanced](#)
- Cluster Config
 - [Cluster Information](#)
 - [Configuration Settings](#)
 - [File Service Settings](#)
- [Storage Policies](#)
- [Repair Status](#)
- [Operation Status](#)

6.10.1. Data Centers

Path: **Cluster** → **Data Centers**

Node Status: ⬮ Unreachable ⬮ Has Disk Error ⬮ Stop Write ⬮ Disk Above 80% Full ⬮ Has Alerts ⬮ Under Maintenance ⬮ Add Node in progress ⬮ All Clear

HyperStore Core Services Status

HOST	ADMIN	CMC	IAM	CASSANDRA	HYPERSTORE	REDIS MON	REDIS CRED	REDIS QOS	S3
deploy-node1	✓	✓	✓	✓	✓	✓ -	✓	✓	✓
deploy-node2	✓	✓	✓	✓	✓	✓	✓	✓ -	✓
deploy-node3	✓	✓	✓	✓	✓		✓ -		✓

* Master/Primary

Auxiliary Services Status

FILE SERVICE POOL	HOST	KFS	FILE
file-pool-1	deploy-file1	✓	✓
	deploy-file2	✓	✓
	deploy-file3	✓	✓
file-pool-2	deploy-file4	✓	✓

Supported tasks:

- View Status of All Nodes in a Data Center (below)
- Adding Nodes — For the full procedure including important steps to take before you add nodes, see ["Adding Nodes"](#) (page 579).
- Adding a Data Center — For the full procedure including important steps to take before you add a data center, see ["Adding a Data Center"](#) (page 590).
- Adding a Region — For the full procedure including important steps to take before you add a region, see ["Adding a Region"](#) (page 597).

6.10.1.1. View Status of All Nodes in a Data Center

The upper part of the **Data Centers** page displays a panel for each data center in your HyperStore system. For each data center, each HyperStore node in the data center is represented by a color-coded cube:



Red with question mark indicates that the **node is unreachable** due to a network problem.



Red with "X" indicates that the **node has experienced disk errors**. Click the node icon to jump to the **Node Status** page for the node, then check that page's [Disk Detail Info](#) panel for more information.



Red with disk stack indicates the node has **stopped accepting writes** -- and the system has stopped directing S3 write requests to that node -- because all of the node's disks are nearly full. For more information on this "stop-write" condition and how to remedy it see ["Automatic Stop of Writes to a Node at 90% Usage"](#) (page 125)

Orange with disk stack indicates the **node's data disks are in aggregate more than 80% full**. Click the node icon to jump to the **Node Status** page for the node, then check that page's [Disk Detail Info](#) panel for more information.



Note For this status "more than 80% full" means that more than 80% of the node's total capacity is either used or "reserved". For more information on "reserved" capacity see ["Capacity Managed"](#) (page 285).



Orange with exclamation mark indicates that the **node has Medium, High, or Critical level alerts that have not yet been acknowledged** by a system administrator. This status will not display if a node has only Low level alerts. Click the node icon to jump to the **Node Status** page for the node, then at the bottom of that page see the [Alert List](#) panel for more information.



Blue with gear indicates that the **node is under maintenance**. This means either that you or another administrator put the node into "maintenance mode" (see ["Start Maintenance Mode"](#) (page 489)).



Green with check mark indicates that the **node has no unacknowledged alerts** nor any of the conditions listed above.

If multiple of the above conditions apply to a node, that node's icon will reflect just the highest priority condition, with the conditions prioritized in this order: "Under Maintenance" > "Unreachable" > "Has Disk Error" > "Disks Above 80% Full" > "Has Alerts".

To view a node's hostname and summary node statistics, hold your cursor over a cube.

The screenshot shows the HyperStore Core Services Status panel. A tooltip for **deploy-node3** is displayed, showing the following metrics:

Disk Used	11%, 4.0 GiB/35.7 GiB	CPU Util	3.0%
Disk Reads/sec	0 B	Disk Writes/sec	44.4 KiB
Network Traffic (Get)	18.0 KiB	Network Traffic (Put)	20.1 KiB
Request Throughput (Get)	39 B	Request Throughput (Put)	6 B
95th percentile Request Latency (ms) (Get)	31	95th percentile Request Latency (ms) (Put)	36

The main table below shows the status of various services across three nodes:

HOST	HYPERSTORE	REDIS MON	REDIS CRED	REDIS QOS	S3
deploy-node1	✓	✓ -	✓	✓	✓
deploy-node2	✓	✓	✓	✓ -	✓
deploy-node3	✓		✓ -		✓

* Master/Primary

The hover text shows the same summary node status information as is available on the upper part of the CMC's **Node Status** page. For description of the status items, see ["View a Node's Summary Status"](#) (page 472) from the **Node Status** page documentation.

To check the status of individual services on every node in a data center, in the lower part of the **Data Centers** page view the **Services Status** panel.

The screenshot shows two panels: **HyperStore Core Services Status** and **Auxiliary Services Status**.

HyperStore Core Services Status

HOST	ADMIN	CMC	IAM	CASSANDRA	HYPERSTORE	REDIS MON	REDIS CRED	REDIS QOS	S3
deploy-node1	✓	✓	✓	✓	✓	✓ -	✓	✓	✓
deploy-node2	✓	✓	✓	✓	✓	✓	✓	✓ -	✓
deploy-node3	✓	✓	✓	✓	✓		✓ -		✓

* Master/Primary

Auxiliary Services Status

FILE SERVICE POOL	HOST	K3S	FILE
file-pool-1	deploy-file1	✓	✓
	deploy-file2	✓	✓
	deploy-file3	✓	✓
file-pool-2	deploy-file4	✓	✓

For each service on each node, one of the following service status icons is displayed:



The service is **up and running**.



The service is **down**.



The service status is unknown because the node is **unreachable** due to a network problem.

Note The service statuses are automatically checked and updated each minute.

6.10.1.2. HyperStore Core Services Status

Status is shown for the following HyperStore core services:

- [Admin Service](#)
- [CMC Service](#) (Cloudian Management Console)
- [IAM Service](#)
- [Cassandra \(Metadata DB\)](#)
- [HyperStore Service](#)
- [Redis Monitor](#)
- [Redis Credentials DB](#)
- [Redis QoS DB](#)
- [S3 Service](#)

For Redis Monitor, Redis Credentials DB, and Redis QoS DB, the service status icon includes a red asterisk (*) if it is the master or primary instance of the service. For the Redis Credentials DB or QoS DB, if the master instance of the service goes down, one of the slave instances becomes the new master (and that new master instance will be marked with an asterisk in the display). By contrast, if the Redis Monitor primary instance goes down, the backup instance takes over the Redis monitoring duties but it does not become the primary instance (and therefore the backup instance will not be marked with an asterisk -- instead the asterisk remains attached to the primary instance even though it's down).

6.10.1.3. Auxiliary Services Status

The "Auxiliary Services Status" section shows:

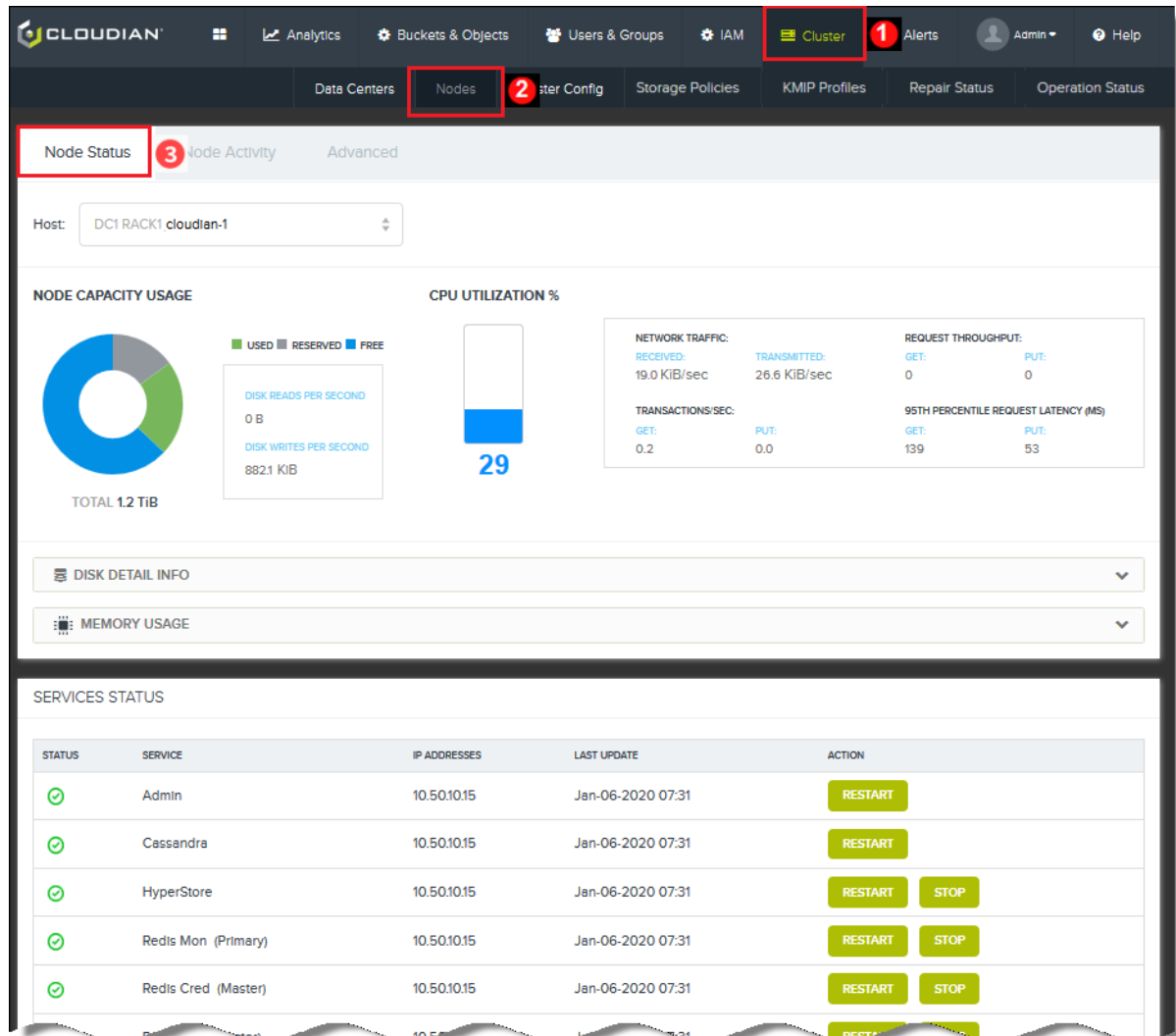
- The status of the [HyperStore File Service](#), if you have installed this optional service.
- The status of the [HyperStore Search Service](#), if you have installed this optional service.

Note The File Service and the Search Service are implemented as containerized applications. For each of these services, K3s -- a lightweight Kubernetes distribution -- provides a container orchestration layer, for load balancing and high availability. If the "Auxiliary Services Status" section indicates

that the File Service or the Search Service or their corresponding K3s services are down/unavailable in your system, contact Cloudbian Support for instructions to check on and restart the downed service.

6.10.2. Node Status

Path: **Cluster** → **Nodes** → **Node Status**



Supported tasks:

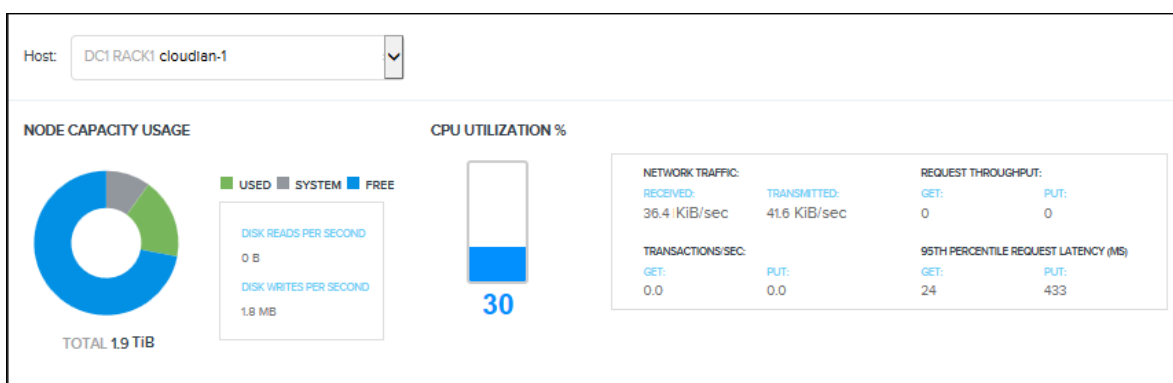
- ["View a Node's Summary Status"](#) (page 472)
- ["View a Node's Disk Detail"](#) (page 474)
- ["Locate a Disk on a HyperStore Appliance"](#) (page 477)
- ["View a Node's Memory Usage"](#) (page 477)
- ["View a Node's Services Status"](#) (page 478)
- ["Start, Stop, or Restart Services On a Node"](#) (page 479)
- ["View and Acknowledge Node Alerts"](#) (page 480)

Note The **Node Status** page does not display status information for [HyperStore File Service nodes](#) or for [HyperStore Search Service nodes](#). If your system includes these optional services you can view the up/down status of those services at the bottom of the **"Data Centers"** (page 466) page.

Note If your HyperStore system has multiple service regions, a drop-down list displays at the top of the page so you can select a region first before selecting a node for which to view status.

6.10.2.1. View a Node's Summary Status

The upper part of the CMC's [Node Status](#) page provides a dashboard view of the health and performance of an individual node within your HyperStore system. A drop-down list lets you choose a node for which to display information.



For the selected node, the **Node Status** page displays current information for the status and performance items described below.

Note The **Node Status** page — and the status and performance information displayed on the page — automatically refreshes once each minute. In the event that the node cannot be reached by the HyperStore Monitoring Data Collector due to a network problem, the top of the **Node Status** page displays a message saying "Node is not reachable. Information on this page may not be accurate."

Node Capacity Usage

The **Node Capacity Usage** graphic shows the percentages of total disk space that are currently Used, System, or Free, on the node as a whole. To see the percentage numbers hold your cursor over each portion of the tri-colored circle.

- The **Used** segment of the circle indicates what portion of the node's total disk space is currently consumed by stored object data. This segment displays in **green** if less than 70% of total capacity is used; or in **orange** if from 70% to 89% is used; or in **red** if 90% or more is used.
- The **System** segment indicates the portion of the node's total disk space that is allocated to system needs and cannot be used for object data storage. The System portion consists of the Linux "reserved blocks percentage" plus the HyperStore stop-write buffer.
 - By default in Rocky Linux and RHEL the "reserved blocks percentage" for a file system (the

portion of the disk space that's reserved for privileged processes) is 5% of disk capacity. In a HyperStore Appliance it's customized to 0%. See your OS documentation if you want to change the current reserved blocks percentage for your HyperStore host machines.

- By default the HyperStore stop-write buffer is 10% of disk capacity. For information on this feature see ["Automatic Stop of Writes to a Disk at 90% Usage"](#) (page 125).

The System segment always displays in **gray**.

- The **Free** segment indicates the portion of the node's total disk capacity that is neither used nor allocated to system needs, and is therefore available for storing new object data. The Free segment always displays in **blue**.

IMPORTANT ! See ["Capacity Monitoring and Expansion"](#) (page 656) for guidance about capacity management and the importance of early planning for cluster expansions.

Disk Reads per second

Across **all** of the node's disks, the average disk read throughput per second during the past minute. The metric will auto-scale from B (bytes) to KiB or MiB or GiB as appropriate.

This system stat is recalculated each minute, based on the most recent minute of data.

Disk Writes per second

Across **all** of the node's disks, the average disk write throughput per second during the past minute. The metric will auto-scale from B (bytes) to KiB or MiB or GiB as appropriate.

This system stat is recalculated each minute, based on the most recent minute of data.

CPU Utilization %

Current CPU utilization on the node. In the icon that graphically shows disk usage percentage, the disk used portion of the icon displays in green if disk usage is less than 70%; in yellow if usage is between 70% and 89%; or in red if usage is 90% or higher.

Network Traffic

The aggregate network interface throughput for the node during the past minute, for all types of network traffic including but not limited to S3 request traffic. For example, data transmission associated with cluster maintenance operations would count toward these statistics. The metric will auto-scale from B (bytes) to KiB or MiB or GiB as appropriate.

For nodes with multiple network interfaces, these system stats are aggregations across the multiple interfaces.

These system stats are recalculated each minute, based on the most recent minute of data.

Request Throughput

For just this node, the data throughput for S3 GET transactions and S3 PUT transactions, expressed as bytes per second.

These S3 stats are recalculated each five minutes, based on the most recent five minutes of S3 transaction activity.

Note HEAD transactions are counted toward the GET stat, and POST transactions are counted toward the PUT stat.

Transactions/sec

For just this node, the number of S3 GET transactions and S3 PUT transactions processed per second.

These S3 stats are recalculated each five minutes, based on the most recent five minutes of S3 transaction activity.

Note HEAD transactions are counted toward the GET stat, and POST transactions are counted toward the PUT stat.

95th Percentile Request Latency (ms)

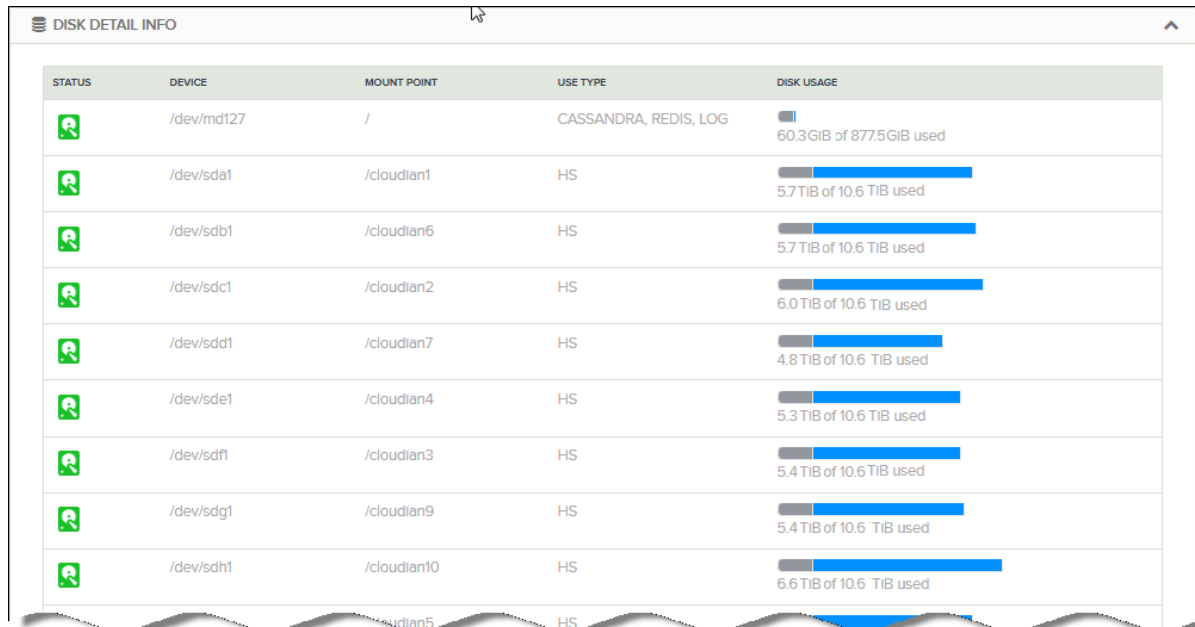
For just this node, the 95th percentile latencies for S3 PUT and S3 GET transactions in milliseconds.

These S3 stats are recalculated each five minutes, based on the most recent approximately 1000 GET transactions and 1000 PUT transactions. Each 95th percentile latency value indicates that of the last 1000 transactions of that type, 95% completed in that many milliseconds or less.

Note HEAD transactions are counted toward the GET stat, and POST transactions are counted toward the PUT stat.

6.10.2.2. View a Node's Disk Detail

To check the status of a node's disks, on the CMC's [Node Status](#) page open the **Disk Detail Info** panel.



STATUS	DEVICE	MOUNT POINT	USE TYPE	DISK USAGE
	/dev/md127	/	CASSANDRA, REDIS, LOG	60.3GiB of 877.5GiB used
	/dev/sda1	/cloudian1	HS	5.7 TiB of 10.6 TiB used
	/dev/sdb1	/cloudian6	HS	5.7 TiB of 10.6 TiB used
	/dev/sdc1	/cloudian2	HS	6.0 TiB of 10.6 TiB used
	/dev/sdd1	/cloudian7	HS	4.8 TiB of 10.6 TiB used
	/dev/sde1	/cloudian4	HS	5.3 TiB of 10.6 TiB used
	/dev/sdf1	/cloudian3	HS	5.4 TiB of 10.6 TiB used
	/dev/sdg1	/cloudian9	HS	5.4 TiB of 10.6 TiB used
	/dev/sdh1	/cloudian10	HS	6.6 TiB of 10.6 TiB used
	/dev/sdi1	/cloudian5	HS	5.7 TiB of 10.6 TiB used

The panel displays the information described below.

Status

This field displays an icon that indicates the disk's summary status.

Icon	Meaning
	OK — The green disk icon indicates that there are no errors for the disk.
	Error — The orange disk icon indicates that the disk is in an error state. - For a HyperStore data disk drive ("Use Type" = "HS"), an Error status indicates that read/write errors related to this disk are appearing in the HyperStore Service application log (<i>cloudian-hyperstore.log</i>), but not in excess of the error rate threshold that would trigger the automatic disabling of the disk. - If multipath I/O (MPIO) is being used for a HyperStore data disk, an Error status will also display if a path to an MPIO disk is down. (The disk will not be disabled unless all paths to the disk are down -- see the information in the next row of this table). - For an SSD storing the OS and metadata ("Use Type" = "Cassandra"), an Error status is triggered if <i>cassandra.log</i> messages appear that include the string "No space left on device". For HyperStore Appliances only, an Error status is also triggered if the drive fails or is no longer part of the software RAID mirroring. If a disk is in an Error state, a Clear Error History button appears toward the bottom of the Detail Disk Info panel. You can click this button to clear the disk's Error status and return the disk status to "OK". If the error occurs again, the disk status display will revert back to showing an Error status for the disk. Note A disk will also show as being in an Error status if you unmount the disk without the disk having first been marked as disabled by the HyperStore system.
	The red disk icon means one of two things depending on whether the disk is a HyperStore data disk or an OS disk: Disabled (data disk) — For a data disk the red icon indicates that the HyperStore system has disabled the disk (so HyperStore doesn't try to write to or read from it any more), unmounted the disk, and transferred all of the disk's tokens to other disks on the host. A data disk can be put into Disabled status in either of two ways: - You or another administrator disabled the disk using the HyperStore <i>disableDisk</i> function. For background information see "Disabling a HyperStore Data Disk" (page 639). - The HyperStore system detected indications of disk failure and automatically disabled the disk, in accordance with the HyperStore Disk Failure Action configuration setting. Note that in the case of a multipath (MPIO) disk the automatic disabling of the disk will occur only if all paths to the disk have failed. Note For recovering from a disabled disk, see "Enabling a HyperStore Data Disk" (page 640) or -- if the disk is bad -- "Replacing a HyperStore Data Disk" (page 642). Note that for a multipath (MPIO) disk you can re-enable the disk if at least one path to the disk is back up.

Icon	Meaning
	• Disk Failure Under RAID-1 (OS disk) — For an OS disk on a HyperStore Appliance node, the red icon indicates that one of the two mirrored disks storing the OS and system metadata has failed and been marked as Offline. This status detection and reporting is supported only for HyperStore Appliance machines, not for software-only deployments of HyperStore. Note If this occurs, the disk will also be marked by a red exclamation mark in the appliance disk map. As part of the Smart Support feature, if a data disk fails (becomes disabled), information about the failed disk is automatically sent to Cloudian Support within minutes. This triggers the automatic opening of a Support case for the failed disk. For HyperStore Appliances, automatic case creation is also performed for failed OS disks.

Device

Disk drive device.

Mount Point

File system mount point for each disk.

Use Type

For each disk, this will be one or more of the following use types:

- "HS" — The disk is storing the HyperStore File System (HSFS). The HSFS is where S3 object data is stored (in the form of object replicas and/or erasure coded fragments).
- "Cassandra" — The disk is storing Cassandra data and/or the Cassandra commit log.
- "Redis" — The disk is storing Redis data
- "Log" — The disk is storing application logs.
- "UNAVAIL" -- The disk is in an error status due to having been unmounted, or is in a disabled or failed status. See the description of Status above.

Disk Usage

This colored bar graphic indicates what portion of the disk's capacity is currently used -- consumed by object storage -- (in **blue**) and what portion is allocated to system needs (in **gray**). For more information about these capacity consumption categories see "[Node Capacity Usage](#)" (page 472)

The blue portion of the graphic turns to:

- **Orange** if the system-allocated space and used space together consume 70 percent or more of the disk's total capacity (but less than 90 percent of total capacity)
- **Red** if the system-allocated space and used space together consume 90 percent or more of the disk's total capacity.

Note Below the bar graphic, in the summary text display that says for example "X GiB of Y TiB used", the system-allocated capacity is counted as "used" along with the capacity consumed by object storage.

6.10.2.3. Locate a Disk on a HyperStore Appliance

If the node is a HyperStore appliance, an **Appliance** panel displays in the CMC's [Node Status](#) page. In the event of a disk problem (as indicated in the [Disk Detail](#) section) you can open the **Appliance** panel for information and tools to help you physically locate the disk within your hardware environment.

The **Appliance** panel displays a drive table that shows the drive usage type ("OS" for drives that store the OS and metadata, or "Data" for drives that storage object data), device name, serial number, and slot number of each drive on the appliance. For each drive in the table, in the right-most column there is a button that you can use to "blink" the ID light of the drive on the appliance. First look through the drive table to find the device name that you're looking for, then click the button for that drive. The exact behavior of the drive light on the appliance depends on the appliance model -- blinking red is the most common -- but regardless, the behavior of the drive light will clearly distinguish that drive from the others on the appliance for a period of three minutes. (The blinking duration is configurable by the setting "[common.appliance.blinkInterval.disk](#)" (page 743); the default is three minutes).

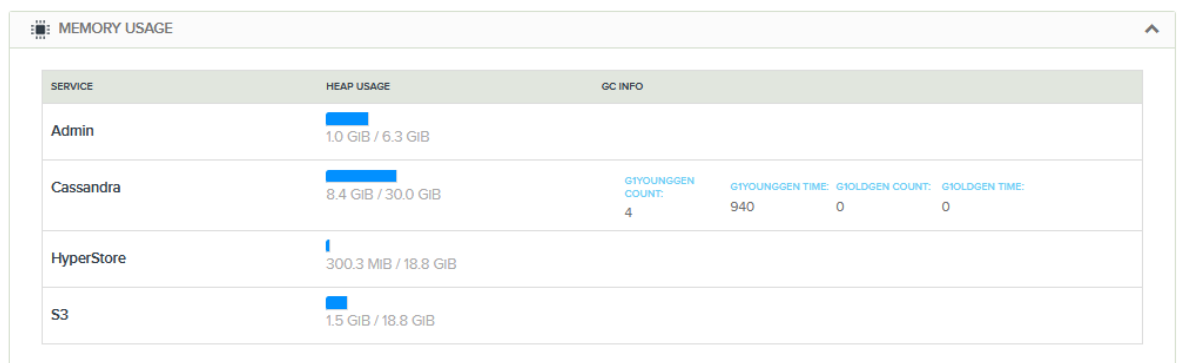
Note If a red exclamation mark appears beside a device name in the drive table, this means that the device is offline (not running). A script monitors the /sys directory on the appliance to detect devices that have gone offline.

Locating the Appliance in a Rack or Cage

If along with help in finding the drive you also want help finding the appliance itself among multiple machines in a rack or cage, you can click **Blink Chassis** in the upper part of the **Appliance** section to "blink" the appliance's chassis light for three minutes. (The blinking duration is configurable by the setting "[common.appliance.blinkInterval.chassis](#)" (page 743); the default is three minutes).

6.10.2.4. View a Node's Memory Usage

To check a node's memory usage, in the CMC's [Node Status](#) page open the **Memory Usage** panel.



The panel displays the following information:

Service

Service name. This will be one of the HyperStore system's four major Java-based services: Admin, Cassandra, HyperStore, and S3.

Heap Usage

This column shows each service's current JVM heap memory usage and also the maximum JVM heap size allocated to each service.

GC Info

Statistics for garbage collection (GC), the recurrent automatic process within a JVM whereby memory is freed up from Java objects that are no longer in-use. These statistics are provided only for the Cassandra service. The supported statistics are:

- G1YoungGenCount — The number of G1GC (Garbage First Garbage Collector) young generation garbage collections executed since the last start-up of the Cassandra service on this node.
- G1YoungGenTime — The aggregate time (in milliseconds) spent on executing G1GC young generation garbage collections since the last start-up of the Cassandra service on this node.
- G1OldGenCount — The number of G1GC old generation garbage collections executed since the last start-up of the Cassandra service on this node.
- G1OldGenTime — The aggregate time (in milliseconds) spent on executing G1GC old generation garbage collections since the last start-up of the Cassandra service on this node.

6.10.2.5. View a Node's Services Status

To check on the status of individual services on a node, in the CMC's [Node Status](#) page open the **Services Status** panel.

STATUS	SERVICE	IP ADDRESSES	LAST UPDATE	ACTION
	Admin	10.10.10.10	Mar-09-2018 13:16	<button>RESTART</button>
	Cassandra	10.10.10.10	Mar-09-2018 13:16	<button>RESTART</button>
	HyperStore	10.10.10.10	Mar-09-2018 13:16	<button>START</button>
	Redis Mon (Primary)	10.10.10.10	Mar-09-2018 13:16	<button>RESTART</button> <button>STOP</button>
	Redis Cred (Master)	10.10.10.10	Mar-09-2018 13:16	<button>RESTART</button> <button>STOP</button>
	Redis Qos (Master)	10.10.10.10	Mar-09-2018 13:16	<button>RESTART</button> <button>STOP</button>
	S3	10.10.10.10	Mar-09-2018 13:16	<button>RESTART</button>

RESTART ALL

For each service, one of the following service status icons is displayed:



The service is **up and running**.



The service is **down**.



The service status is unknown because the node is **unreachable** due to a network problem.

Status is shown for the following services:

- [Admin Service](#)
- [Cassandra \(Metadata DB\)](#)
- [HyperStore Service](#)
- [Redis Credentials DB](#) (if hosted on the node)
- [Redis QoS DB](#) (if hosted on the node)
- [Redis Monitor](#) (if hosted on the node)
- [S3 Service](#)

Note The node's status is automatically checked and updated each minute, as indicated by the "Last Update" column.

6.10.2.6. Start, Stop, or Restart Services On a Node

To stop, start, or restart individual services on a node, in the CMC's [Node Status](#) page open the **Services Status** panel.

STATUS	SERVICE	IP ADDRESSES	LAST UPDATE	ACTION
✓	Admin	10.0.0.10	Mar-09-2018 13:16	RESTART
✓	Cassandra	10.0.0.10	Mar-09-2018 13:16	RESTART
⊘	HyperStore	10.0.0.10	Mar-09-2018 13:16	START
✓	Redis Mon (Primary)	10.0.0.10	Mar-09-2018 13:16	RESTART STOP
✓	Redis Cred (Master)	10.0.0.10	Mar-09-2018 13:16	RESTART STOP
✓	Redis Qos (Master)	10.0.0.10	Mar-09-2018 13:16	RESTART STOP
✓	S3	10.0.0.10	Mar-09-2018 13:16	RESTART

RESTART ALL

For services that are currently up and running, **Restart** and **Stop** buttons display in the "Action" column. If you click either button, you will be asked to confirm that you want to proceed and then upon your confirmation the restart or stop operation will execute. If the operation succeeds you will see a success message — for example, "Stopping Redis QoS...[OK]".

For services that are not currently running, a **Start** button displays in the "Action" column. If you click **Start**, the service starts up and you should see a success message — for example, "Starting Redis QoS...[OK]".

At the bottom right of the **Services Status** section a **Restart All** button displays. If you click **Restart All** and confirm, then:

- Services that are running will be restarted.
- Services that are down will be started.

Note In a **single-region** HyperStore system, the **Node Status** page does not support stopping the Admin Service or S3 Service. In a **multi-region** HyperStore system, the **Node Status** page does not support stopping the Admin Service or S3 Service in the default region.

On a **single-node** system, in addition to the above restrictions the **Node Status** page also does not support stopping the Cassandra Service. In a **multi-node** system the **Node Status** page does support stopping Cassandra, but doing so may cause certain CMC functions to no longer work.

If you want to stop a service that you cannot stop on the **Node Status** page you can [do so with the HyperStore installer tool or with the HyperStore initialization scripts](#).

6.10.2.7. View and Acknowledge Node Alerts

At the bottom of the CMC's **Node Status** page is an **Alert List** section that displays a list of node alerts (for the one node that you've selected at the top of the page). In this section you can review and acknowledge node alerts.

ALERT LIST					+ SHOW ACKNOWLEDGED
☐	SEVERITY	ALERT TYPE	ALERT TEXT	LAST UPDATE	COUNT
☐	High	HyperStore	[Service Down or Unreachable]	Mar-09-2018 13:21	17

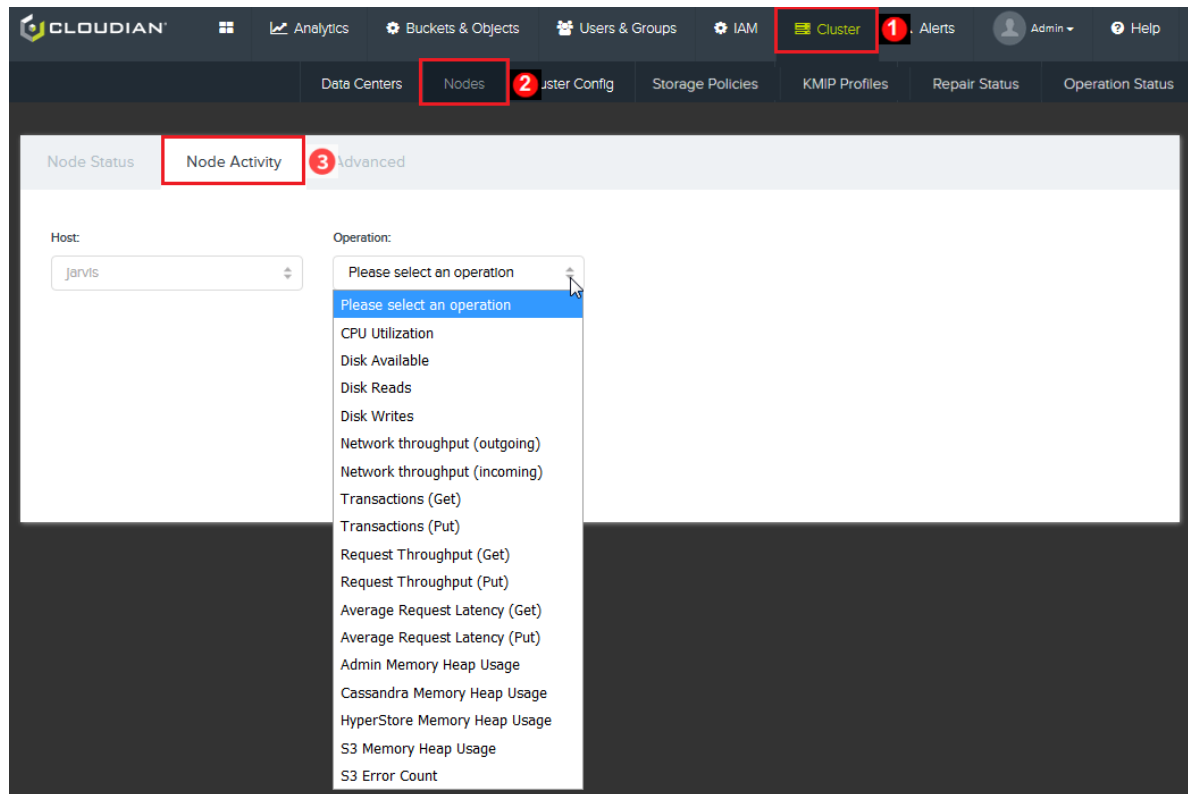
[ACKNOWLEDGE](#)

This section has the same functionality as the CMC's **Alerts** page, except that the information is limited to the one node that you've selected. For more guidance on understanding and working with this display, see **"Alerts"** (page 555).

Note Acknowledged alerts are **automatically deleted from the system** after a time period configured by the **"monitoring.event.acknowledgedTTI"** (page 807) setting. By default this period is 86400 seconds (one day). After they are deleted acknowledged alerts will not display in the Alert List even if you click **Show Acknowledged**. If you wish you can reduce the configurable time-to-live for acknowledged alerts to as little as 1 second (so that they are deleted from your system right after acknowledgment). Note that regardless of your configured time-to-live for acknowledged alerts, a record of your system's alert history will persist in the [Smart Support](#) logs that by default are uploaded to Clodian Support each day.

6.10.3. Node Activity

Path: **Cluster** → **Nodes** → **Node Activity**



Supported task:

- Check a node's recent activity

In the CMC's **Node Activity** page you can view a variety of health and performance statistics for individual HyperStore nodes. For each node, statistics from the past 30 days are available, and within that time period you can flexibly drill down into whatever specific interval you're interested in.

Select a node in the "Host" field and then in the "Operation" field you can choose any one of the statistics listed below.

CPU Utilization

CPU utilization percentage for the node. This is measured once per every five minutes.

Disk Available

Disk space available on the node. Note that:

- Only disks that store HyperStore data directories or the Metadata DB (Cassandra) data directory count toward this stat. This stat does not count disks that are being used only for OS or application files, for example.
- This stat calculates disk usage in a way that counts a disk's "reserved-blocks-percentage" (the portion of the disk space that's reserved for privileged processes) as used space. Consequently this stat may indicate a higher degree of disk usage than would the Linux *df* command, which does not count reserved space towards a disk's used space. By default in Linux systems the configurable "reserved-blocks-percentage" for a file system is 5% of disk capacity.

Disk Reads/Writes

Across all the node's disks that are being used for S3 object storage, the average disk read and write

throughput per second. These stats are measured each minute, based on the most recent minute of activity.

Network Throughput Outgoing/Incoming

The node's outgoing and incoming network interface throughput per second. This encompasses all types of network traffic including but not limited to S3 request traffic. For example, data transmission associated with cluster maintenance operations would count toward these statistics. These stats are measured each minute, based on the most recent minute of activity.

For throughput specifically of S3 request traffic, see the Request Throughput stats.

Transactions (GET/PUT)

Number of S3 transactions processed per second by the node. This is broken out to GET transactions and PUT transactions.

HEAD transactions are counted toward the GET stat, and POST transactions are counted toward the PUT stat.

These stats are measured each five minutes, based on the most recent five minutes of activity.

Request Throughput (GET/PUT)

For S3 transactions, the data volume throughput per second for the node. This is broken out to GET transactions and PUT transactions.

HEAD transactions are counted toward the GET stat, and POST transactions are counted toward the PUT stat.

These stats are measured each five minutes, based on the most recent five minutes of activity.

Average Request Latency (GET/PUT)

For S3 transactions, the 95th percentile request latency for the node, in milliseconds. This is broken out to GET transactions and PUT transactions.

New statistic values are calculated each five minutes, based on the most recent approximately 1000 GET transactions and 1000 PUT transactions. The latency values indicate that of the last 1000 transactions of that type (GET or PUT), 95% completed in that many milliseconds or less.

HEAD transactions are counted toward the GET stat, and POST transactions are counted toward the PUT stat.

<Service> Memory Heap Usage

The current JVM heap memory usage (in number of bytes) by each of the HyperStore system's major Java-based services on the node: Admin, Cassandra, HyperStore, and S3.

These stats are measured each five minutes.

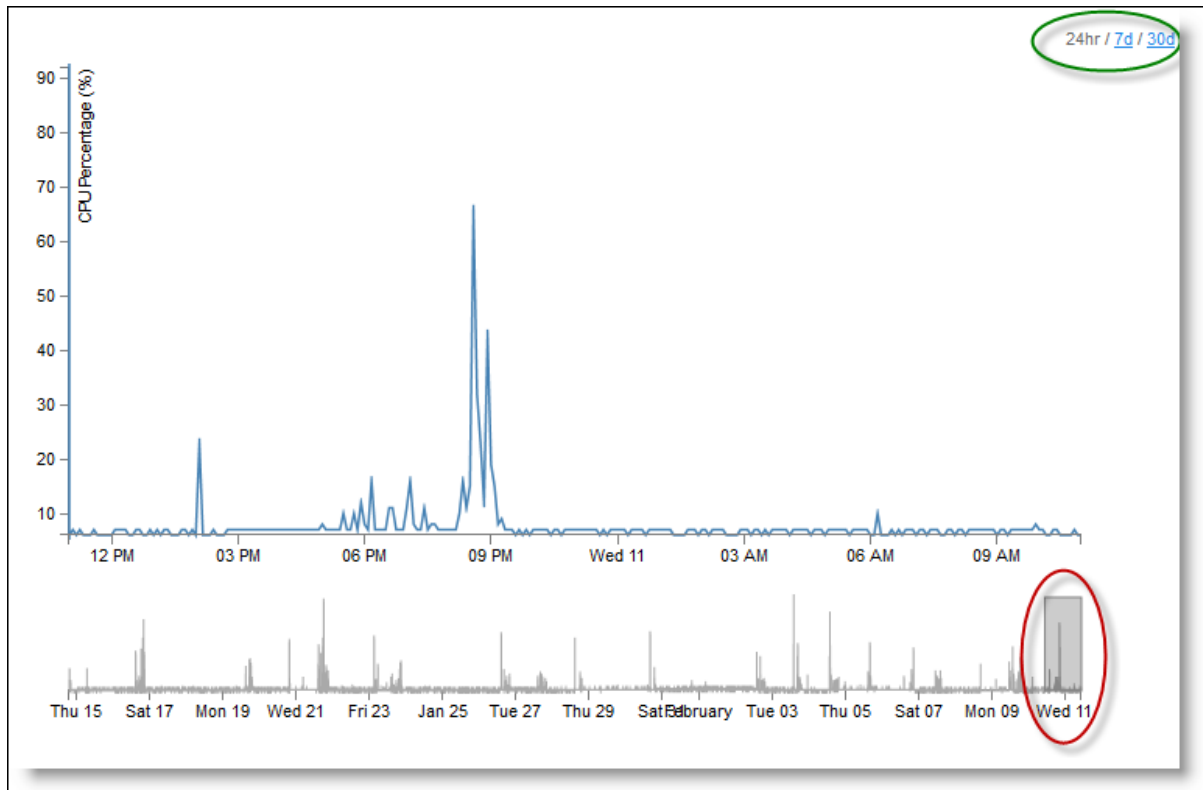
S3 Error Count

Running count of the S3 service errors (5xx status responses) returned to S3 client applications by the node. This is a cumulative count since the last restart of the S3 Service on the node.

By default, when you choose a statistic from the "Operation" drop-down list, a graph of the statistic's movement over the **past 24 hours** displays. The interface provides you two methods for adjusting this graphing interval:

- In the upper right of the page, you can click to change the graph to a 7 day or 30 day view.
- In the small graph at the bottom of the page — which shows a compressed 30 day view — you can manipulate the gray block to control the time period that's shown in the main graph. With your mouse you can click and drag the left or right edges of the block to expand or contract the time period shown in

the main graph. You can also click the block's center and drag the block to shift the main graph to an earlier or later time interval (within the bounds of the past 30 days).



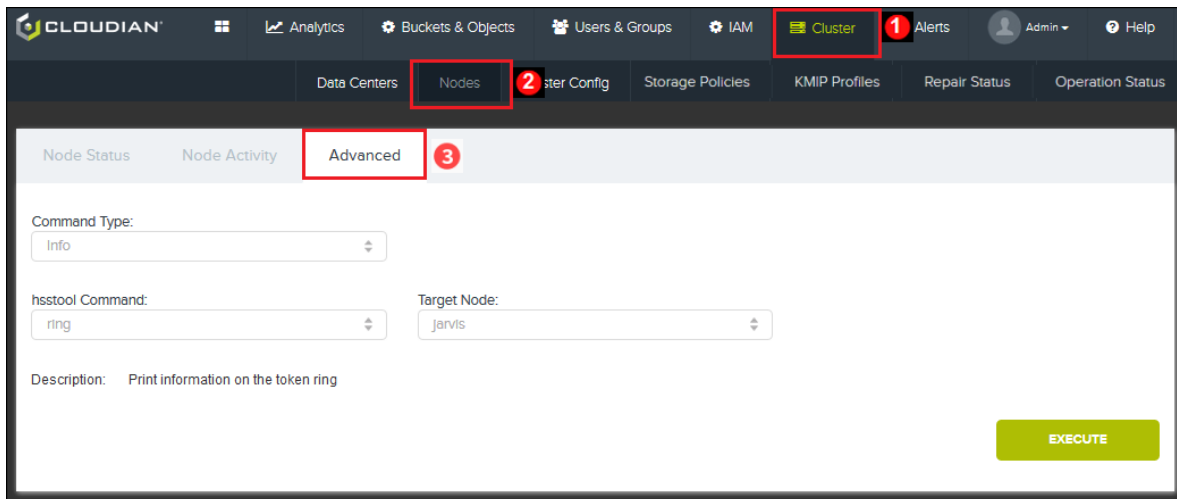
Note For statistics that measure quantities of data, the metric used on a graph's Y-axis auto-scales to units that are most appropriate for the particular quantities being conveyed. Specifically, for a given statistic the Y-axis may be expressed in terms of bytes, KiBs, MiBs, or GiBs, depending on the activity level during the full 30-day graphing interval. Pay attention to the Y-axis label to see what metric is being used.

Note for multi-region systems

If your HyperStore system has multiple service regions, a drop-down list displays at the top of the page so you can select a region first before selecting a node for which to view activity.

6.10.4. Node Advanced

Path: **Cluster** → **Nodes** → **Advanced**



Supported tasks:

- ["Info Commands"](#) (page 484)
- ["Maintenance Commands"](#) (page 484)
- ["Redis Monitor Commands"](#) (page 485)
- ["Disk Management Commands"](#) (page 485)
- ["Cassandra QoS Commands"](#) (page 485)
- ["Collect Diagnostics"](#) (page 487)
- ["Start Maintenance Mode"](#) (page 489)
- ["Uninstall Node"](#) (page 490)

6.10.4.1. Info Commands

In the CMC's [Node Advanced](#) page, with Command Type "Info" selected, you can execute any of the following *hssstool* commands which retrieve information about your system:

- [ring](#) — View vNode info for whole cluster
- [info](#) — View vNode and data load info for a physical node
- [status](#) — View summary status for whole cluster
- [opstatus](#) — View status of repair or cleanup operations
- [proactiverepairq](#) — View status of proactive repair queues
- [repairqueue](#) — View auto-repair schedule
- [ls](#) — View vNode and data load info per mount point)
- [whereis](#) — View storage location information for an S3 object
- [metadata](#) — View metadata for an S3 object
- [trmap](#) — View a list of active token range map snapshots

6.10.4.2. Maintenance Commands

In the CMC's [Node Advanced](#) page, with Command Type "Maintenance" selected, you can execute any of the following *hssstool* commands for acting on the data and metadata in your system:

- [cleanup](#) — Clean a node of replicated data that doesn't belong to it
- [cleanupec](#) — Clean a node of erasure coded data that doesn't belong to it
- [autorepair](#) — Enable or disable auto-repair
- [proactiverepair](#) -- Enable or disable or stop or start proactive repair
- [repair](#) — Repair the replicated data on a node
- [repairec](#) — Repair the erasure coded data on a node
- [repaircassandra](#) — Repair the system and object metadata on a node
- [rebalance](#) — Rebalance some data from the cluster to a newly added node

6.10.4.3. Redis Monitor Commands

In the CMC's [Node Advanced](#) page, with Command Type "Redis Monitor Operations" selected, you can execute any of the following commands for the Redis Monitor Service:

- [setClusterMaster](#) — Move the Credentials DB master role or the QoS DB master role
- [getClusterInfo](#) — Get information about the Credentials DB cluster or the QoS DB cluster

Note The Redis Monitor supports additional commands that can only be executed on the command line, not through the CMC. For more information see "[Redis Monitor Commands](#)" (page 947).

6.10.4.4. Disk Management Commands

In the CMC's [Node Advanced](#) page, with Command Type "Disk Management" selected, you can execute any of the following commands for managing HyperStore data disks:

- [disableDisk](#) — Temporarily disable a disk
- [enableDisk](#) — Re-enable a disk that is currently disabled
- [replaceDisk](#) — Replace a disk

6.10.4.5. Cassandra QoS Commands

HyperStore has a rate-limiting feature that automatically ensures that no single bucket places excessive load on the Metadata DB (Cassandra). Processing incoming S3 requests entails queries for bucket and object metadata in the Metadata DB -- typically multiple Metadata DB queries per S3 request. The system continually tracks the Metadata DB requests resulting from S3 activity associated with each bucket. Buckets that generate an excessively high load (more than a threshold number of Metadata DB queries per 5 minute interval) over a sustained period of time (five successive 5-minute intervals) will be automatically put into a restricted state. While in a restricted state, the bucket will not be allowed to execute Metadata DB queries, and S3 requests incoming to the bucket will fail with an HTTP 503 error response. This restricted state for the bucket will last 5 minutes and then the bucket will be released from restriction. Any recurrence of sustained excessive Metadata DB load from the bucket will result in the bucket going back into a 5 minute restriction period again.

The per-bucket Metadata DB request threshold is calculated dynamically, based on factors including the number of nodes in your system, the number of disks per node, the disk space usage levels in your system, and the storage policy that a bucket uses (because of this last factor, different buckets may have different thresholds due to using different storage policies). The per-bucket thresholds are updated periodically in response to changes in disk space usage levels in your system.

The objective of this Metadata DB rate limiting feature is to prevent any single bucket from negatively impacting service performance for your user population as a whole.

In the CMC's [Node Advanced](#) page the "Cassandra QoS Management" commands can be used to check on and control certain aspects of the Metadata DB rate limiting feature. In the documentation below, the commands are divided into two tables. The commands in the first table are generally applicable, but the commands in the second table are only applicable in the unlikely circumstance noted in the introduction to that table.

Command	Description	Parameters
getRestrictedBuckets	Get a list of buckets that are currently in the restricted state. These are buckets that have exceeded their threshold for five successive 5-minute intervals. While in a restricted state, these buckets will not be allowed to execute Metadata DB queries, and S3 requests incoming to the buckets will fail with an HTTP 503 error response. This restricted state for each such bucket will last 5 minutes and then the bucket will be released from restriction.	--
setRestrictedBucket	Put a bucket into a restricted state. The bucket will not be allowed to execute Metadata DB queries, and S3 requests incoming to the bucket will fail with an HTTP 503 error response. This restricted state for the bucket will last 5 minutes and then the bucket will be released from restriction.	bucket name
getQoSRunTimeInterval	Get the time interval at which the QoS process executes -- this by default is the 5 minutes (300 seconds) that has been referenced in the descriptions of threshold behavior in this documentation.	--
setQoSRunTimeInterval	Set the time interval at which the QoS process executes, in number of seconds. This would be to use an interval other than the 5 minute (300 second) default. Note that if you change this, then in all the threshold behavior descriptions in this documentation, the actual behavior will be based on your specified interval rather than the default 5 minutes.	interval in seconds
shutdownQoS	Disable the Metadata DB QoS feature, system-wide. Note that this has the effect of removing all restricted buckets from restriction, and when you re-enable the Metadata DB QoS feature those buckets will not be in restriction (unless they again exceed the Metadata DB request threshold for five successive 5-minute intervals).	--
startQoS	Enable the Metadata DB QoS feature, system-wide.	--
qosStatus	Check the current status of Metadata DB QoS feature -- this will return "true" for enabled or "false" for disabled. By default the feature is enabled.	--
getQoSMeterStatus	Check whether the Metadata DB QoS feature is enabled on a specific node. By default when the Metadata DB QoS feature is enabled in the system as a whole it is enabled on each node (a QoS "meter" runs on each node). But you can use the <i>setQoSMeterStatus</i> command to disable the QoS meter on a specific node (and to subsequently re-enable it); and the <i>getQoSMeterStatus</i> command to check the QoS meter's current status on a specific node. Note that for the Metadata DB QoS feature to be operational on a given node, the (system-wide)	node

Command	Description	Parameters
	<i>qosStatus</i> must indicate "true" and the node-specific QoS meter status must be "enabled". This is the default status in the system.	
setQoSMeterStatus	Enable or disable the Metadata DB QoS feature on a specific node. See the <i>getQoSMeterStatus</i> description for more information.	node, enable/disable
reportBucketQoSStats	Retrieve Metadata DB QoS statistics for a specific node. This is information that Cloudian Support may use if they are helping you troubleshoot a system issue.	node

The commands in this second table are applicable **only in the unlikely event that Cloudian Support has you disable the dynamic threshold calculator** (introduced in HyperStore version 8.2) so that the Metadata DB rate limiting feature reverts back to its pre-8.2 implementation which used static per-bucket thresholds rather than dynamically calculated thresholds. In the table below, references to the "threshold" mean the number of allowed Metadata DB requests per 5 minute interval.

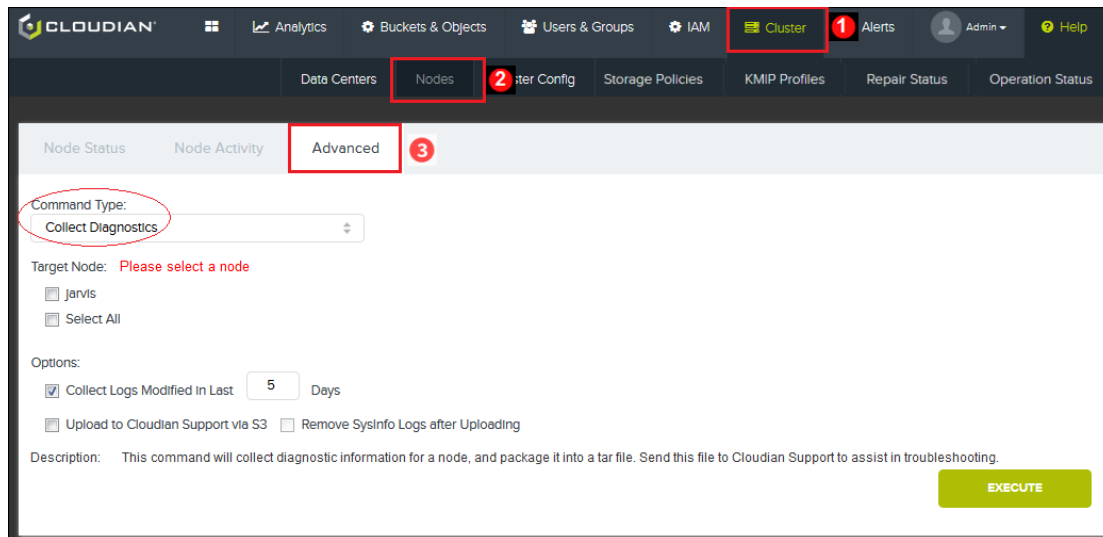
Command	Description	Parameters
setThresholdForBucket	Set a threshold for a specific bucket. This will override the default per-bucket threshold, for this specific bucket.	bucket name, threshold
getThresholdForBucket	Get the current threshold for a specific bucket.	bucket name
getDefaultBucketThreshold	Get the current default per-bucket threshold.	--
setDefaultBucketThreshold	Set a default per-bucket threshold.	threshold
removeThresholdForBucket	Remove the threshold assigned to a specific bucket, so that the bucket reverts back to using the bucket default threshold.	bucket name

6.10.4.6. Collect Diagnostics

When you are experiencing system problems, Cloudian Support is best able to help you if you provide them with comprehensive diagnostic information. Through the CMC you can easily generate a package of diagnostic information for a specified node or nodes. Each node's diagnostics package will be created on the node itself, at path `/var/log/cloudian/cloudian_sysinfo/<hostname>_<YYYYMMDDmmss>.tar.gz`. The package will include log files, configuration files, statistics, and command outputs for that particular node. Optionally, you can have the diagnostics package file get automatically uploaded via S3 to Cloudian Support or an alternative S3 destination of your choosing.

To generate diagnostics for a node or nodes:

1. In the CMC's **Node Advanced** page, from the Command Type menu select "Collect Diagnostics":



2. In the "Target Node" section, select one or more nodes for which to collect diagnostics data.
3. Select from among the diagnostics processing options:

Collect Logs Modified in the Last X Days

Use this to specify how many days' worth of system, application, and transaction logs should be collected from the target node(s). The default behavior is to collect the last 5 days' worth of logs.

Upload to Clouidian Support via S3

Select this checkbox if you want the diagnostics from the target node(s) to be automatically uploaded to Clouidian Support, immediately after creation of the diagnostics package.

Note Optionally, you can have the automatic upload go to a different S3 destination rather than Clouidian Support. To do so requires that you first implement a system configuration change. For more information see ["Configuring Smart Support and Node Diagnostics"](#) (page 120).

Remove SysInfo Logs After Uploading

This option is applicable only if you are having the diagnostics automatically uploaded to an S3 destination. If you select the "Upload to Clouidian Support via S3" checkbox then the "Remove SysInfo Logs After Uploading" becomes checked also by default. With this option checked, then after the upload to the S3 destination the local copy of the diagnostics package file will be automatically deleted. If you want to retain the local copy of the diagnostics package (under a `/var/log/cloudian/cloudian_sysinfo` directory on the target node[s]) as well as uploading a copy to the S3 destination, then uncheck the "Remove SysInfo Logs After Uploading" checkbox.

Note If the "Remove SysInfo Logs After Uploading" checkbox is not selected when you perform the diagnostics collection operation, then the diagnostics package is retained on the target node(s) for 15 days before being automatically deleted.

4. Click **Execute**.
5. In the confirmation dialog that displays, confirm that you want to proceed with the operation.

Once you confirm, the Result section of the page displays a message indicating the command has been sent. After the operation completes, the Result section displays a brief summary of the operation outcome.

6.10.4.7. Start Maintenance Mode

Through the CMC you can place a node into "maintenance mode". When a node is in maintenance mode:

- The node will not generate alerts.
- If the node is hosting a Credentials DB master or QoS DB master, the master role will be temporarily shifted to a different node.
- The S3 Service on other nodes will not direct requests to the HyperStore Service or the Metadata DB on the node that is maintenance mode.

Note When a node is in maintenance mode, a load balancer can still direct incoming S3 client requests to that node. If you do not want the node to receive incoming S3 requests from clients you can [stop the S3 Service](#) on the node.

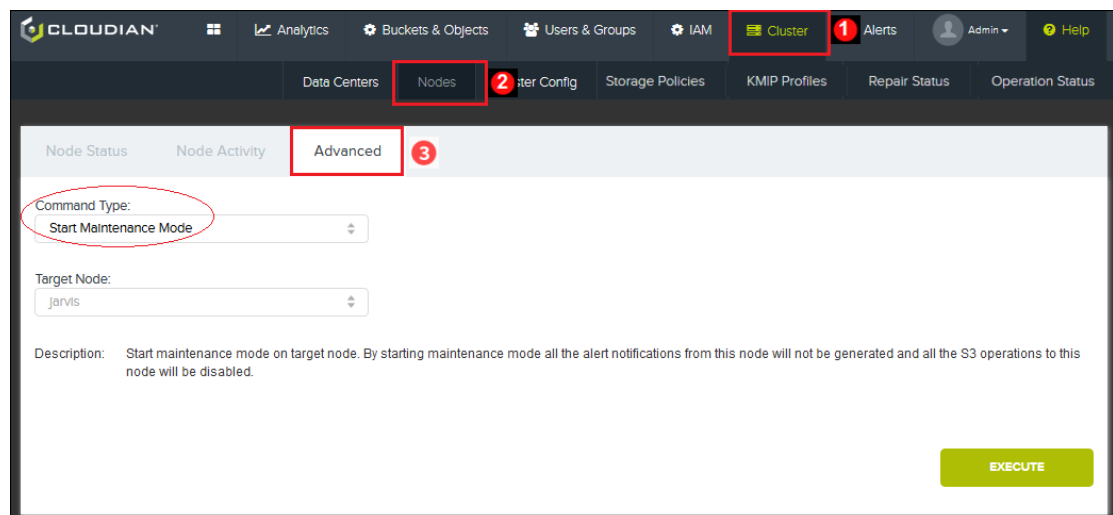
Within a service region, you can have no more than one node in maintenance mode at a time. Also, the system will not allow you to put a node into maintenance mode if:

- Any other node in the service region is down or unreachable
- Any service is down on any other node in the service region
- Any other node in the region is in stop-write mode

To check whether any other nodes are in any of these conditions use the CMC's [Data Centers](#) page.

To put a node into maintenance mode:

1. In the CMC's **Node Advanced** page, from the Command Type menu select "Start Maintenance Mode":



2. From the "Target Node" list, select the node that you want to put into maintenance mode.
3. Click **Execute**.

Note If any requests to the HyperStore Service or Metadata DB are in processing on the target node when you submit the Start Maintenance Mode command, the processing of those requests will be completed before the node goes into maintenance mode.

While a node is in maintenance mode, in the CMC's [Data Centers](#) page a special blue icon will indicate that the node is in maintenance mode. Also, when a node is in maintenance mode, the system supports using fall-back consistency levels for S3 write or read requests for which the node is an endpoint (if you have configured ["Dynamic Consistency Levels"](#) (page 80).)

Stopping Maintenance Mode

To take a node out of maintenance mode and put it back into regular service:

1. In the CMC's **Node Advanced** page, from the Command Type menu select "Stop Maintenance Mode".
2. From the "Target Node" list, select the node that you want to take out of maintenance mode
3. Click **Execute**.

Note If when you put the node into maintenance mode you also stopped the S3 Service on the node, remember to start the S3 Service again when you take the node out of maintenance mode.

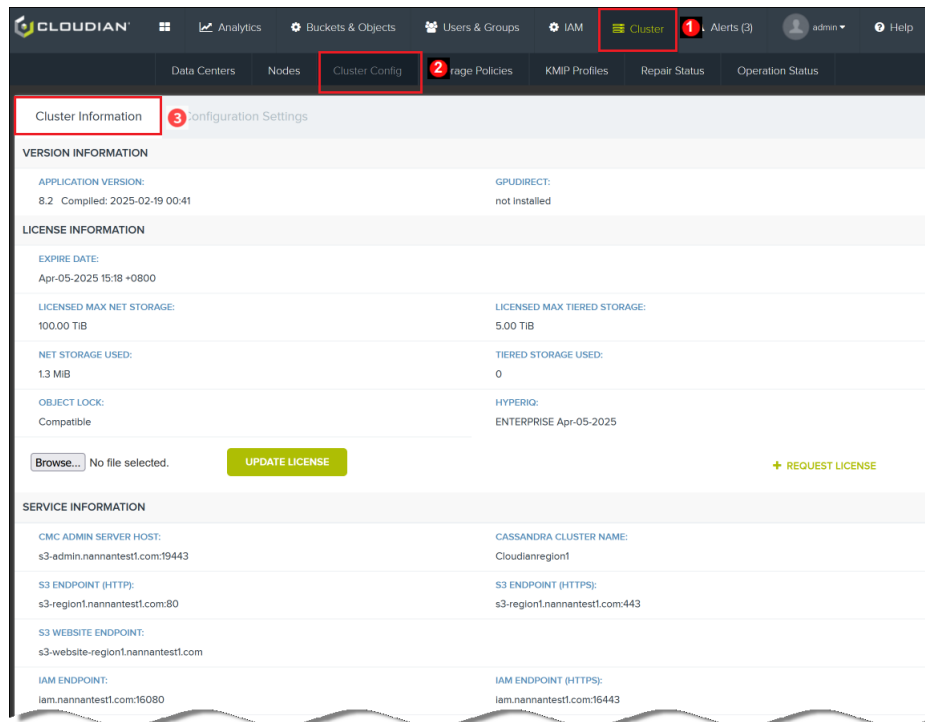
6.10.4.8. Uninstall Node

In the CMC's [Node Advanced](#) page, you can use the Command Type "Uninstall Node" to remove a node from your cluster.

For the full procedure including important steps to take before removing a node, see ["Removing a Node"](#) (page 603).

6.10.5. Cluster Information

Path: **Cluster** → **Cluster Config** → **Cluster Information**



Supported tasks:

- View Cluster Summary Information:
 - ["Version Information"](#) (page 491)
 - ["License Information"](#) (page 492)
 - ["Service Information"](#) (page 494)
- ["Renew and Install a License"](#) (page 497)

6.10.5.1. Version Information

Application Version

Your current HyperStore software version

GPUDirect

This field indicates the status of the NVIDIA® GPUDirect® Storage (GDS) feature in your HyperStore system.

Possible statuses are:

- Not installed -- The NVIDIA CUDA® object (*cuobject*) libraries required for the GDS feature are not installed on your HyperStore nodes.
- Disabled -- The NVIDIA CUDA object libraries required for the GDS feature are installed on your HyperStore nodes, but you have not enabled the GDS feature by configuration. The GDS feature is disabled by default. If the CUDA object libraries are installed you can enable GDS by using the configuration setting ["common.rdma.enabled"](#) (page 749).
- Enabled -- The NVIDIA CUDA object libraries required for the GDS feature are installed on your HyperStore nodes and you have enabled the GDS feature by configuration.

If the GPUDirect field shows "N/A" that means the CMC was unable to retrieve the GPUDirect status for your system due to an error.

For more information on the GPUDirect Storage feature see ["GPUDirect Storage Feature Overview"](#) (page 143) and ["GDS Setup"](#) (page 146).

6.10.5.2. License Information

Expire Date

Date on which your license expires.

If you reach the **warning period** preceding your license expiration, then when you use any part of the CMC, the top of the interface displays a warning that your license expiration date is approaching.

If you reach your license **expiration date** you enter a grace period, per the terms of your contract. During the grace period:

- In the CMC, the top of the console screen displays a warning indicating that your license has expired.
- The system still accepts and processes all types of incoming S3 requests, but every S3 response returned by the S3 Server includes an extension header indicating that the system license has expired (header name: *x-gemini-license*; value: *Expired: <expiry_time>*)

If you reach the **end of your grace period** after the license expiration date:

- No S3 service is available for end users except for GET and HEAD operations (so existing can still be listed and read). All other types of incoming S3 requests -- such as PUT operations associated with creating buckets or uploading objects -- will be rejected with a "403 Forbidden" error response. The response also includes the expiration header described above.
- The top of the CMC screen will display an error message indicating that your license has expired. Also, in the CMC's [Dashboard](#) page the Cluster Health panel will indicate that the system is "Disabled". (More specifically the system is only partially disabled, in that it allows only S3 GET and HEAD operations and no other types of S3 operations.)

For information on renewing your license, see ["Renew and Install a License"](#) (page 497).

Licensed Max Net/Raw Storage

The maximum amount of object data storage your HyperStore system license allows for. This will be in terms of either Net storage or Raw storage, depending on your particular license terms.

With a license based on Net storage, the limit is on total object storage bytes excluding overhead from object replication or erasure coding. For example if a 1GiB object is replicated three times in your system it counts as only 1GiB toward the Net Storage limit.

With a license based on Raw storage, the limit is on total bytes stored on formatted disks in your system. This encompasses object storage bytes including overhead from object replication or erasure coding, as well as bytes of stored metadata. Note that with this type of storage limit, if a 1GiB object is replicated three times in your system it counts as 3GiB toward the limit.

In a multi-region HyperStore system, the licensed storage limit is for the system as a whole (all regions combined).

For more information on licensing see ["Licensing and Auditing"](#) (page 36).

Licensed Max Tiered Storage

The maximum amount of tiered data storage your HyperStore system license allows for.

All auto-tiered data stored in any destination system **other than HyperStore** counts toward this limit. Data auto-tiered from one of your HyperStore regions to another region, or from your HyperStore system to an external HyperStore system, does not count toward this limit.

If this value is "Unlimited" then the license places no limit on tiered data volume.

For more information on licensing see ["Licensing and Auditing" \(page 36\)](#).

Net Storage Used / Raw Storage Used

If your HyperStore license is based on Net storage, this field is "Net Storage Used" and the number displayed is your current total storage usage excluding overhead from object replication or erasure coding. For example if a 1MiB S3 object is replicated three times in your system it counts as only 1MiB toward the "Number of Stored Bytes" count.

If your HyperStore license is based on Raw storage, this field is "Raw Storage Used" and the number displayed is your current total raw storage including overhead from object replication or erasure coding. For example if a 1MiB S3 object is replicated three times in your system it counts as 3MiB toward the "Raw Storage Used" count.

The storage usage count for your system is updated at the top of each hour. In a multi-region HyperStore system, the count is for the system as a whole (all regions combined).

This field displays a warning message if your current bytes count exceeds 70% of your licensed usage maximum; and a critical message if your current bytes count exceeds 90% of your licensed usage maximum

Note If some users have **versioning** enabled on their buckets -- so that the system retains rather than overwriting older versions of an object when the user uploads a new version of the object -- then each stored object version (the older versions as well as the current version) counts toward your system usage count.

Also, if some users use the **cross-region replication** feature to replicate objects from one HyperStore bucket to another HyperStore bucket in the same HyperStore system, then the original source objects and the object replicas in the destination bucket both count toward your system bytes count. For more information see ["Cross-Region Replication Impact on Usage Tracking" \(page 215\)](#)

Tiered Storage Used

The current tiered storage usage level in external systems other than HyperStore.

Data auto-tiered from one of your HyperStore regions to another region in the same HyperStore system, or from your HyperStore system to an external HyperStore system, does not count toward this figure. Auto-tiered data stored in any other type of external system -- such as the Amazon, Azure, or Google clouds -- does count toward this figure.

Note A warning message displays here if this figure exceeds 70% of your licensed maximum tiered storage limit, or a critical message if this figure exceeds 90% of your licensed maximum tiered storage limit.

Object Lock License

Your HyperStore license's type of support for the Object Lock (WORM) feature:

- Disabled -- Your license does not support the Object Lock feature.
- Compatible -- Your license supports the "Compatible" Object Lock type. Appropriate if you want Object Lock capabilities but you are not subject to the data protection mandates of U.S. SEC-17a or a comparable regulatory regime and you want more flexibility than is allowed by the "Certified" Object Lock type.
- Certified-- Your license supports the "Certified" Object Lock type. Appropriate if you want Object Lock capabilities and you are subject to the data protection mandates of U.S. SEC-17a or a comparable regulatory regime.

Note For Certified Object Lock to work, you must enable the HyperStore Shell and disable password-based root access to HyperStore nodes (as described in ["Preparing the Object Lock Feature"](#) (page 224)). If your license supports Certified Object Lock but you have not yet taken these steps, in the **Cluster Information** page the "Object Lock License" field displays "Certified - Object lock feature requires HyperStore Shell(HSH) to be locked."

For information about the Object Lock feature including full descriptions of the "Compatible" and "Certified" types of licensed Object Lock support, see ["Object Lock Feature Overview"](#) (page 221).

HyperIQ License

Cloudian HyperIQ is a solution for dynamic visualization and analysis of HyperStore system monitoring data and S3 service usage data. HyperIQ is a separate product available from Cloudian that deploys as virtual appliance on VMware or VirtualBox and integrates with your existing HyperStore system. For more information about HyperIQ contact your Cloudian representative.

The HyperIQ License field in the **Cluster Information** page indicates what level of HyperIQ functionality will be available to you **if you acquire and set up HyperIQ**.

- Basic -- HyperIQ dashboards for hardware, OS, and service status monitoring are supported indefinitely.
- Enterprise <expiration date> -- HyperIQ dashboards for hardware, OS, and service status monitoring are supported indefinitely, and also an S3 analytics dashboard is supported until the Enterprise expiration date. The presence of the S3 analytics dashboard is what distinguishes Enterprise level HyperIQ support from Basic HyperIQ support.

6.10.5.3. Service Information

Note If you have a [multi-region HyperStore system](#), a drop-down list lets you choose a region for which to view Service Information.

For information about moving service roles from one HyperStore node to another, see ["Change Node Role Assignments"](#) (page 616).

CMC Admin Server Host

Admin Service endpoint to which the CMC connects in order to submit Admin Service calls. The CMC makes calls to the [Admin Service](#) when you use the CMC to perform administrative tasks like adding users, configuring rating plans, or viewing system monitoring data. In a multi-region system there is just one Admin Service endpoint for the whole system.

All of the HyperStore nodes in the [default service region](#) should service requests to this endpoint. In a production environment, typically you would have the Admin service endpoint resolve to the virtual IP(s) of one or more load balancers in the default service region, and the load balancers would in turn distribute traffic across all the HyperStore nodes in the default region. For more information see ["DNS Set-Up"](#) (page 53).

Note The label "CMC Admin Server Host" is somewhat misleading. In the current version of HyperStore this is a service endpoint, not a single host or IP address as it was in earlier HyperStore versions.

Cassandra Cluster Name

Name of the [Cassandra](#) (Metadata DB) cluster in this service region. In a multi-region HyperStore system, each region has an independent Cassandra cluster. The system automatically derives a Cassandra name from the local region name.

S3 Endpoint (HTTP) and S3 Endpoint (HTTPS)

S3 endpoint(s) for this service region. In a multi-region system, each region has its own S3 endpoint -- for HTTP, and optionally HTTPS. S3 client applications connect to these endpoints to submit S3 API calls.

Note HyperStore by default derives the S3 service endpoint(s) as `s3-<regionname>.<domain>`, based on the domain and region name that you specify during installation. However HyperStore also supports the format `s3.<regionname>.<domain>` (with a dot after the "s3" rather than a dash) if you specify custom S3 endpoints with this format during your HyperStore installation.

Note For information about using HTTPS for the S3 Service, see ["Security and Privacy Features"](#) (page 177).

S3 Website Endpoint

S3 website endpoint for this service region. Web browsers use the S3 website endpoint to access content in buckets that are [configured as static websites](#). In a multi-region system, each region has its own S3 website endpoint. For S3 website endpoints, only HTTP is supported -- not HTTPS.

IAM Endpoint (HTTP) and IAM Endpoint (HTTPS)

Identity and Access Management (IAM) service endpoints for the system. In a multi-region system, all regions use these same endpoints. IAM client applications connect to these endpoints to submit IAM API calls.

Redis Credentials Master Host

Host on which the [Credentials DB Master node](#) is located. In a multi-region system, there is just one Credentials DB master for the whole system.

If the one Credentials DB master is located in a service region other than the one that you are currently viewing, the display will show "Not installed".

Redis Credentials Slave Host(s)

Hosts on which the [Credentials DB slave nodes](#) are located. This is typically two nodes per data center.

Redis QoS Master Host

Host on which this service region's [QoS DB Master node](#) is located. In a multi-region system, each region has

its own QoS DB Master.

Redis QoS Slave Host(s)

Hosts on which the [QoS DB slave nodes](#) are located. This is typically one node per data center.

Redis Monitor Primary Host

Host on which the primary [Redis Monitor](#) instance is located. This is just one host per whole system, even for a multi-region system.

If the one Redis Monitor primary instance is located in a service region other than the one that you are currently viewing, the display will show "Not installed".

Redis Monitor Backup Host

Host on which the backup [Redis Monitor](#) instance is located. This is just one host per whole system, even for a multi-region system. If the Redis Monitor primary instance fails, the Redis Monitor services automatically fail over to the backup host.

If the one Redis Monitor backup instance is located in a service region other than the one that you are currently viewing, the display will show "Not installed".

System Monitoring / Cronjob Primary Host

In this service region, the host on which the primary instance of the HyperStore [Monitoring Data Collector](#) and the primary instance of the HyperStore [system maintenance crontab](#) are located. In a multi-region system, each region has one System Monitoring / Cronjob Primary Host.

System Monitoring / Cronjob Backup Host

In this service region, the host on which the backup instance of the HyperStore [Monitoring Data Collector](#) and the backup instance of the HyperStore [system maintenance crontab](#) are located. In a multi-region system, each region has one System Monitoring / Cronjob Backup Host. If the System Monitoring / Cronjob Primary Host fails, the system monitoring and cron job services automatically fail over to the backup host.

External NTP Host(s)

In each DC in this service region, the external NTP servers to which the HyperStore system's internal NTP servers connect. For information about how HyperStore automatically implements a robust time-synchronization set-up using NTP, see ["NTP Automatic Set-Up" \(page 857\)](#).

Internal NTP Host

In each DC in this service region, the hosts acting as internal NTP servers. There will be four internal NTP hosts per data center. (If a HyperStore data center has only four or fewer nodes, then all the nodes in the data center are configured as internal NTP servers.) For information about how HyperStore automatically implements a robust time-synchronization set-up using NTP, see ["NTP Automatic Set-Up" \(page 857\)](#).

Note If you are running any HyperStore File Service or HyperStore Search Service nodes in a data center, up to two of those nodes will also be acting as internal NTP servers, but they will not be listed in the "Internal NTP Host" display in the CMC's **Cluster Information** page.

Config Controller Primary Host

Host on which the active [Config Controller](#) is located. This is just one host per whole system, even for a multi-region system.

Config Controller Backup Host

Host that is configured to be the backup host for the [Config Controller](#). This is just one host per whole system, even for a multi-region system. If the node on which the active Config Controller is located fails, you can **manually** fail over the Config Controller role to the backup host. Automatic fail-over of this role is not supported. For instructions see ["Move the Config Controller Primary or Backup Role"](#) (page 624).

If the one Config Controller backup host is located in a service region other than the one that you are currently viewing, the display will show "Not installed".

Installation Staging Directory

On the Config Controller primary host, the installation staging directory path. In this directory resides the *cloudianInstall.sh* script, which you can use for certain cluster management tasks such as apply configuration file edits to the cluster or [restarting services](#). By default the installation staging directory for HyperStore version 8.2 is */opt/cloudian-staging/8.2*.

Number of Datacenters

Number of data centers in which you are running the HyperStore system.

Number of Hosts in Each Datacenter

Number of HyperStore hosts in each data center, in format *<DCName>: <#hostsInDC>*.

6.10.5.4. Renew and Install a License

In the CMC's [Cluster Information](#) page you can request a HyperStore license renewal, and install a new or renewed license file.

Request a License Renewal

Click **Request License** to send an email to *cloudian-license@cloudian.com* to initiate the process of obtaining a new license file. Clicking this button should open your default email application (for instance, Gmail).

If nothing happens when you click **Request License**, your browser is not configured for launching your email application. You can instead open your email application manually and send a license renewal request to *cloudian-license@cloudian.com*.

Install a New License File

Note This feature is for installing a cluster-wide license, not a license specific to a particular HyperStore Appliance machine.

After you've obtained a new license file from Cloudian, you need to apply the new license file to your HyperStore system. You can do so through the **Cluster Information** page:

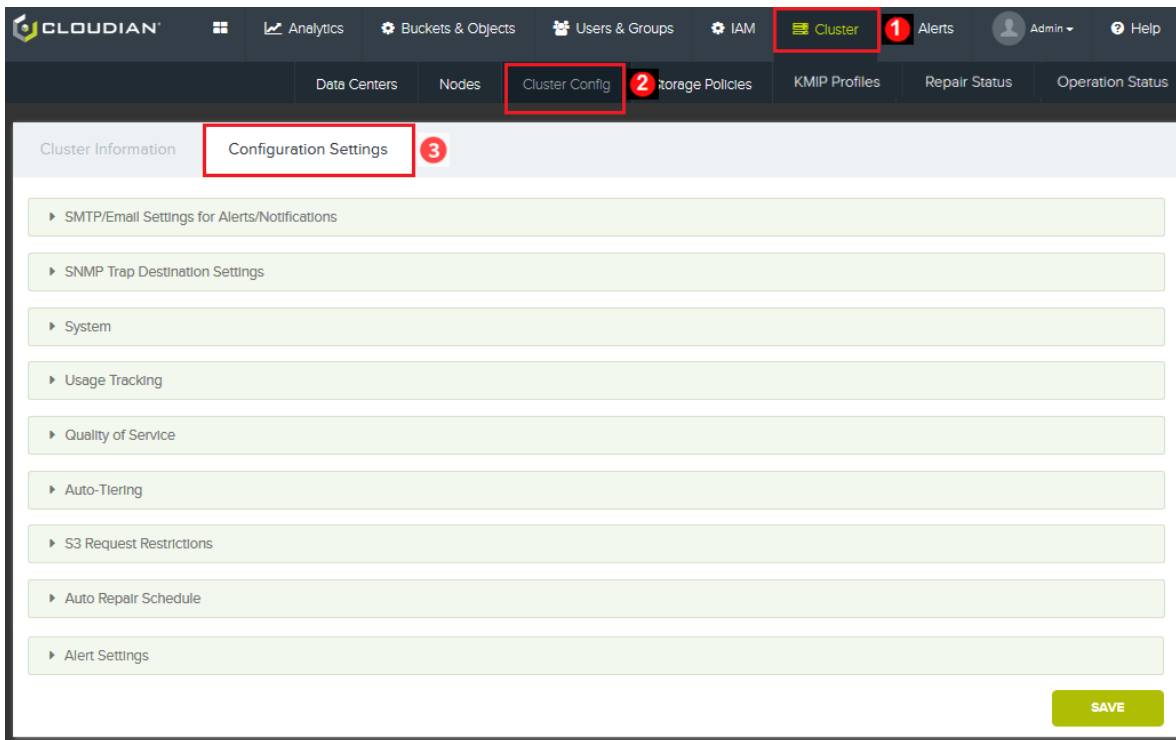
1. Put the license file on the computer from which you are accessing the CMC (the computer on which your browser is running).
2. On the left side of the **Cluster Information** page click **Browse** (if your browser is Firefox) or **Choose File** (if your browser is Chrome). On your local machine, browse to and select the license file.
3. Click **Update License**.

The CMC pushes the license file to the Config Controller node, which in turn propagates the file to all your HyperStore nodes. The Config Controller then triggers your HyperStore nodes to dynamically reload the license file. You do not need to restart any services.

After the process completes, if you refresh the **Cluster Information** page you should see updated information in the **License Information** section.

6.10.6. Configuration Settings

Path: **Cluster** → **Cluster Config** → **Configuration Settings**



Supported task:

- Edit HyperStore configuration settings

When you change settings in the CMC's **Configuration Settings** page the system applies your configuration changes dynamically — no service restart is required. The system also automatically updates and pushes the relevant configuration file settings so that your configuration changes persist across any future restarts of HyperStore services.

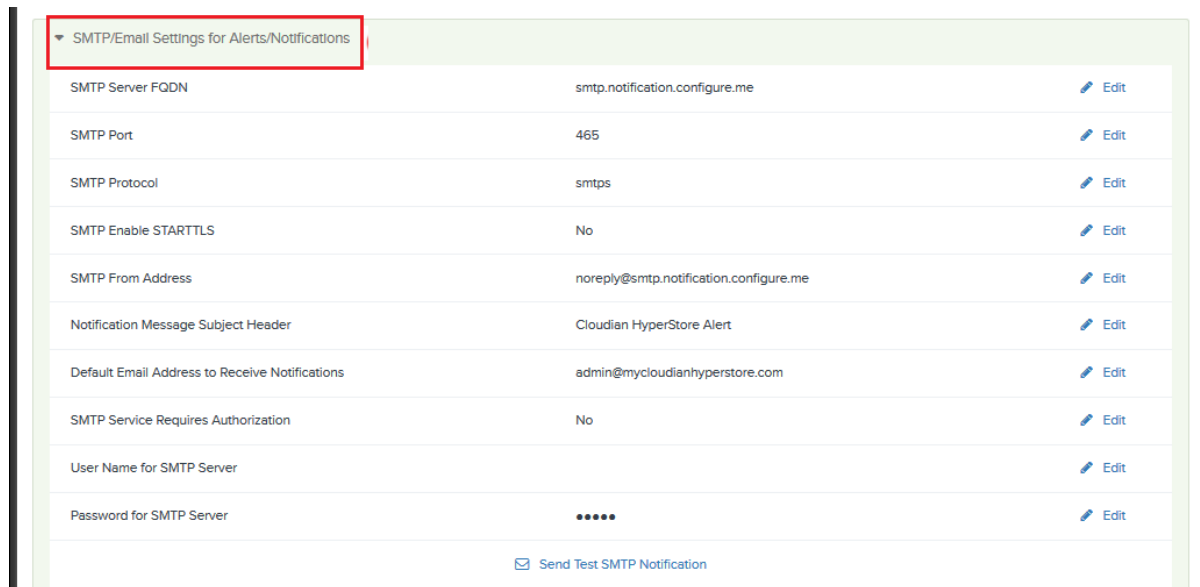
You can edit any of the following types of settings. **Be sure to click Save at the bottom of the page** to save your changes.

Note In a multi-region system, the CMC's **Configuration Settings** page does not support selecting a region. Instead, these settings apply to the whole system. They are not region-specific.

SMTP/Email Settings for Alerts/Notifications

The settings in this section of the CMC's **Configuration Settings** page pertain to the SMTP server that you want the HyperStore system to use when it sends alert notification emails to system administrators. Providing the

system with this information is essential for proper system monitoring and administration.



SMTP/Email Settings for Alerts/Notifications		
SMTP Server FQDN	smtp.notification.configure.me	Edit
SMTP Port	465	Edit
SMTP Protocol	smtps	Edit
SMTP Enable STARTTLS	No	Edit
SMTP From Address	noreply@smtp.notification.configure.me	Edit
Notification Message Subject Header	Cloudian HyperStore Alert	Edit
Default Email Address to Receive Notifications	admin@mycloudianhyperstore.com	Edit
SMTP Service Requires Authorization	No	Edit
User Name for SMTP Server		Edit
Password for SMTP Server	•••••	Edit

[Send Test SMTP Notification](#)

To use this feature you will need information about your organization's mail server, and at least one system administrator email account must have already been set up with the mail server and be able to receive email.

After making any edits in the SMTP/Email Settings section, click **Save** at the bottom of the page to dynamically apply the configuration change to the HyperStore system. Then open the SMTP/Email Settings section again and click **Send Test SMTP Notification** to send a test email to the system administrator email address that you configured. Then confirm that the test email was received at that address. (In the event of a failed test, on the CMC node on which you conducted the test check `/var/log/cloudian/cloudian-ui.log` for error messages.)

Note You can set alert notification triggers in the [Alert Rules](#) page.

Note Alert emails are sent by the [HyperStore System Monitoring / Cron Job host](#), using your specified SMTP server.

SMTP Server FQDN

Fully qualified domain name (FQDN) of the SMTP server that the HyperStore system should utilize for sending alert notification emails to system administrators.

Default = smtp.notification.configure.me

SMTP Port

Listening port of the SMTP service that the HyperStore system should utilize for sending alert notification emails to system administrators.

Default = 465

SMTP Protocol

Protocol to use when sending alert notification emails. Options are *smtp* or *smtps*.

Default = smtps

SMTP Enable STARTTLS

Whether to use STARTTLS when sending alert notification emails. Options are Yes or No.

If you enable this option you should also set the SMTP Protocol setting to "smtp" (not "smtps") and set the SMTP Port setting to 587 (if your email server is using the typical STARTTLS port).

Default = No

Note Even if you enable STARTTLS, if the recipient email account is a Gmail account the account user will still need to set the 'Allow less secure apps' option on the Gmail dashboard.

SMTP From Address

The "From" address to use when sending alert notification emails.

Default = noreply@smtp.notification.configure.me

Notification Message Subject Header

The "Subject" to use when sending alert notification emails.

Default = Cloudian HyperStore Alert

Default Email Address to Receive Notifications

Default system administrator email address(es) to which to send alert notification emails. If you want alert notification emails to go to multiple email addresses, enter the addresses as a comma-separated list. Note that this default recipient address can optionally be overridden for specific alerts when you configure alert rules in the CMC's **Alert Rules** page.

Default = admin@mycloudianhyperstore.com

SMTP Service Requires Authorization

Whether the SMTP server that the HyperStore system will use requires clients to use SMTP Authentication. Options are Yes or No.

Default = No

User Name for SMTP Server

If SMTP Authentication is required by the SMTP server, the username to submit with requests to the server. Comma (,) and double-quote (") are not supported in this setting's value.

Default = No default

Password for SMTP Server

If SMTP Authentication is required by the SMTP server, the password to submit with requests to the server. Comma (,) and double-quote (") are not supported in this setting's value.

Default = No default

SNMP Trap Destination Settings

The settings in this section of the CMC's **Configuration Settings** page configure an SNMP trap destination, in support of having HyperStore system alerts sent as SNMP traps.

Note For information about enabling SNMP trap sending for individual alert types, see ["Alert Rules" \(page 560\)](#). For traps to be sent, you must configure the SNMP trap destination settings and then also enable trap sending for individual alert types.

In the traps that HyperStore generates and sends to your specified destination, the OID is *enterprises.16458.4.1.1.1*. The trap payload also indicates the specific HyperStore host on which the trap-triggering event occurred. HyperStore uses SNMP version 2c. Trap are sent by the [HyperStore System Monitoring / Cron Job host](#).

Destination Endpoint

IP address or FQDN of the SNMP manager to which the CMC will send system alerts (for alert types for which you've enabled SNMP trap sending in the CMC's **Alert Rules** page).

This setting does not support entering multiple endpoints. Enter only one endpoint.

Default = None

Destination Port

Listening port used by the SNMP manager.

Default = 162

System Settings

The settings in this section of the CMC's **Configuration Settings** page configure storage system operations.

HyperStore Disk Failure Action

The automated action for the system to take in the event that read or write errors occur for a particular HyperStore data disk. Supported automated actions are:

- **Disable Disk + Move its Tokens** — With this setting, if disk errors are detected the system will automatically unmount the disk and mark it as disabled. The system will also automatically transfer all of the disabled disk's storage tokens to the remaining disks on the host, in an approximately balanced way. The data from the disabled disk will **not** be recreated on the other disks (repair operations will not be automatically triggered, and even if you triggered repair operations manually the data from the disabled disk would not be recreated on the other disks because with this disk disabling approach the moved tokens are assigned updated timestamps).

When a disk is disabled in this way, writes of new or updated S3 object data that would have gone to

the disabled disk will go to the other disks on the host instead. Meanwhile existing S3 object data will be unreadable on the host. Whether the system as a whole can still provide S3 clients with read access to the affected S3 objects depends on the storage policies with which the objects are associated, and on the availability of other replicas or erasure coded fragments within the cluster.

When a disk is disabled in this way, an alert is generated and the disk will show as disabled in the Disk Detail Info section of the **Node Status** page.

Note When you subsequently perform the ["Enabling a HyperStore Data Disk" \(page 640\)](#) operation or the ["Replacing a HyperStore Data Disk" \(page 642\)](#) operation, the tokens will automatically be moved back to the re-enabled or replacement disk. Data written in association with the tokens when they were on the other disks will remain on those other disks and does not need to be moved to the re-enabled or replacement disk. For information on how HyperStore tracks token location over time so that objects can be written to and read from the correct disks, see ["Dynamic Object Routing" \(page 127\)](#).

- **None** — With this option the system will not automatically disable a disk for which errors have occurred. Each disk error will trigger an alert, however.

By default, your configured automatic disk failure action will be triggered if 10 "HSDISKERROR" error messages for the disk occur in the HyperStore Service application log within a 5 minute span.

Your configured automatic disk failure action will also be triggered if the HyperStore drive audit feature detects that a disk is in a read-only condition.

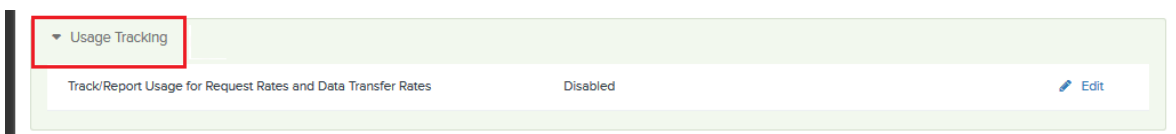
Note The automatic disk disabling feature works only if you have multiple HyperStore data disks on the host. If there is only one HyperStore data disk on the host, the system will not automatically disable the disk even if errors are detected.

Also, the automatic disk failure handling feature does not work correctly in Xen, Logical Volume Manager (LVM), or Amazon EC2 environments. Contact Cloudbian Support if you are using any of these technologies.

Default = Disable Disk + Move Its Tokens

Usage Tracking Settings

The settings in this section of the CMC's **Configuration Settings** page configure the HyperStore system's functionality for tracking service usage by users and groups.



Note For an overview of the HyperStore usage reporting feature, see ["Usage Reporting Feature Overview" \(page 252\)](#).

Track/Report Usage for Request Rates and Data Transfer Rates

Whether to have the system maintain counts for number of HTTP requests, number of bytes-in, and number of

bytes-out for each user and for each user group. By default this functionality is disabled, and the system only tracks and reports on stored bytes and stored object counts per user and group.

You must change this setting to "Enabled" if you want the system to support usage reports that report on number of HTTP requests (GETs, PUTs, and DELETES), bytes-in (upload) volume, and bytes-out (download) volume per user and group. This is important if you want to bill or charge-back users or groups based in part on request volume and/or data transfer volume.

Note that enabling this feature does not produce request counts and data transfer counts for user activity that occurred during the period when the feature was disabled. Rather, if you enable this feature then request tracking and data transfer tracking starts from that point forward.

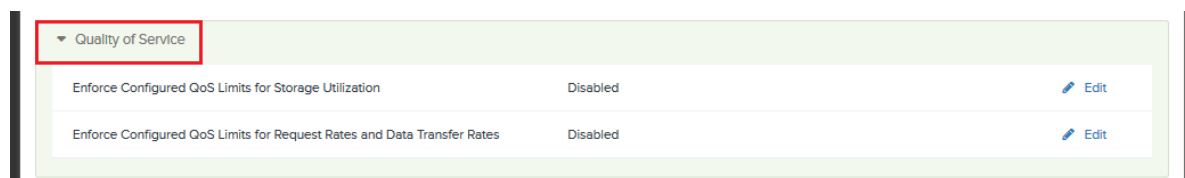
Default = Disabled

Note This setting applies only to per-user and per-group usage statistic tracking. It does not apply to per-bucket usage tracking. If you have [enabled per-bucket usage tracking](#), then all usage types -- including request rates and data transfer rates -- are available on a per-bucket basis regardless of this setting.

Note If you enable this setting, then the retention period for raw usage data in the Metadata DB is automatically reduced from seven days to one day. This is due to the large amount of raw usage data that can accumulate quickly if request and data transfer usage tracking is enabled. Note that each hour, raw usage data gets automatically rolled into hourly roll-up usage data -- which has a default retention period of 65 days.

Quality of Service Settings

The settings in this section of the CMC's **Configuration Settings** page configure the HyperStore system's quality of service (QoS) feature.



Note For an overview of the HyperStore QoS feature, see ["Quality of Service \(QoS\) Feature Overview" \(page 246\)](#).

Enforce Configured QoS Limits for Storage Utilization

If this setting is **enabled** the system will enforce Quality of Service (QoS) limits for number of stored bytes and number of stored objects, if you have [set such limits for users and groups](#).

If this setting is **disabled** the system will not enforce any QoS limits, even if you have set such limits for users and groups.

Note To enforce QoS limits for stored bytes and number of stored objects **but not** QoS limits for HTTP request rate, bytes-in rate, and bytes-out rate, set:

Enforce Configured QoS Limits for Storage Utilization = enabled

Enforce Configured QoS Limits for Request Rates and Data Transfer Rates = disabled

To enforce QoS limits for stored bytes and number of stored objects **and also** QoS limits for HTTP request rate, bytes-in rate, and bytes-out rate, set:

Enforce Configured QoS Limits for Storage Utilization = enabled

Enforce Configured QoS Limits for Request Rates and Data Transfer Rates = enabled

Enforcing QoS for traffic rates but not for stored bytes and objects is not supported at the system configuration level. If you want to use QoS in this way, set both of the QoS enforcement settings to enabled, then when you're configuring QoS limits for groups and users set the stored bytes and objects controls to unlimited and the rate controls to your desired levels.

Default = Disabled

Enforce Configured QoS Limits for Request Rates and Data Transfer Rates

If this setting is **enabled** the system will enforce QoS limits for HTTP request rate, bytes-in rate, and bytes-out rate, if you have [set such limits for users and groups](#).

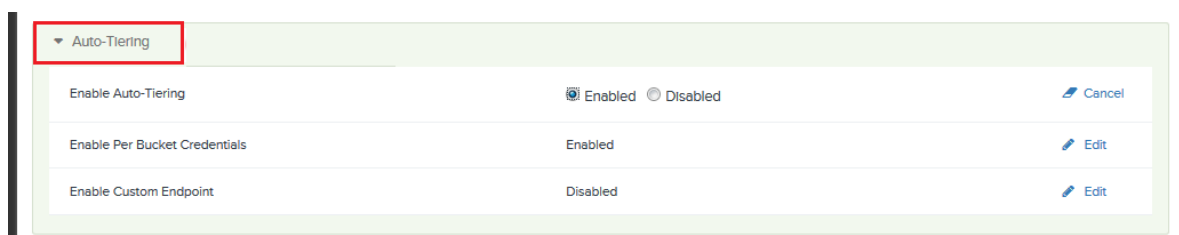
If this setting is **disabled**, then the system will not enforce HTTP request rate, bytes-in rate, and bytes-out rate limits, even if you have set such limits for users and groups.

Enabling this setting is supported only if the *Enforce Configured QoS Limits for Storage Utilization* setting is also enabled.

Default = Disabled

Auto-Tiering Settings

The settings in this section of the CMC's **Configuration Settings** page configure the HyperStore system's auto-tiering feature.



▼ Auto-Tiering		
Enable Auto-Tiering	☒ Enabled ☐ Disabled	Cancel
Enable Per Bucket Credentials	Enabled	Edit
Enable Custom Endpoint	Disabled	Edit

Note For an overview of the HyperStore auto-tiering feature, see ["Auto-Tiering Feature Overview" \(page 203\)](#). For a high level view of how these settings work together in combination -- or if you want to support auto-tiering to a HyperStore region or to a different HyperStore system -- see ["Preparing the Auto-Tiering Feature" \(page 209\)](#).

Enable Auto-Tiering

This setting controls whether or not the CMC will support configuring auto-tiering rules for buckets. If you set

"Enable Auto-Tiering" to **Enabled**, then the [CMC interface for configuring a bucket lifecycle](#) will include an option for configuring an auto-tiering rule. If "Enable Auto-Tiering" is set to **Disabled** (as it is by default), then the bucket lifecycle configuration interface will not display an option for configuring an auto-tiering rule.

Note that this setting only controls whether bucket lifecycle rules for auto-tiering can be configured **through the CMC**. Even if you have this setting disabled, bucket lifecycle rules for auto-tiering can still be configured through HyperStore extensions to the S3 API method [PutBucketLifecycle](#), by other S3 client applications that call that API method.

If you set "Enable Auto-Tiering" to Enabled then additional settings (described below) will display in the "Auto-Tiering" section of the **Configuration Settings** page, allowing you control over the specific auto-tiering options that will be available to users configuring bucket lifecycle auto-tiering rules through the CMC.

Default = Disabled

Enable Per Bucket Credentials

This setting displays only if "Enable Auto-Tiering" is set to Enabled.

If "Enable Per Bucket Credentials" is **Enabled** (as it is by default), then users configuring bucket lifecycle auto-tiering rules through the CMC will be able to choose from several different options for tiering destination, and they will provide (through the CMC interface) their own security credentials for their selected tiering destination. By default the tiering destinations that will be available to users are:

- AWS S3 (default endpoint = <https://s3.amazonaws.com>)
- AWS Glacier (default endpoint = <https://s3.amazonaws.com>)
- Google Cloud Storage (default endpoint = <https://storage.googleapis.com>)
- Azure (default endpoint = <https://blob.core.windows.net>)
- Spectra BlackPearl (default endpoint = <https://bplab.spectrallogic.com>) (note that Spectra BlackPearl is in the default list of tiering destinations only if original installed HyperStore version was older than 7.1.4).

Note You can change this list of destinations and/or their endpoints if you wish. For details see "[Configure Tiering Destinations](#)" (page 210).

If you set "Enable Per Bucket Credentials" to **Disabled**, then all users configuring bucket lifecycle auto-tiering rules through the CMC will be presented with just one, system-default tiering destination, and all tiering to that destination will use a default set of security credentials that you supply to the system. You set this one default tiering destination and the one set of security credentials with the "Default Tiering URL" and "Default Tiering Credentials" settings that will display if you set "Enable Per Bucket Credentials" to Disabled. These settings are described further below.

Default = Enabled

Enable Custom Endpoint

This setting displays only if "Enable Auto-Tiering" and "Enable Per Bucket Credentials" are both set to Enabled.

If you set "Enable Custom Endpoint" to **Enabled**, then users configuring bucket lifecycle auto-tiering rules through the CMC will be able to choose from among the several tiering destinations described above under "Enable Per Bucket Credentials" **and also** they will have the option to enter an endpoint URL for an S3-compliant destination of their own choosing. Users will enter their own security credentials for their specified tiering destination. HyperStore will attempt to connect to the specified custom domain using the user's supplied credentials, and if the connection is successful the user will be able to auto-tier to that domain.

If "Enable Custom Endpoint" is **Disabled** (as it is by default), then users configuring bucket lifecycle auto-tiering rules through the CMC will not be able to specify a custom tiering destination endpoint. Instead they will choose among the several system-configured destinations (the default list of destination is described under "Enable Per Bucket Credentials" above).

Default =Disabled

Default Tiering URL

This setting displays only if "Enable Auto-Tiering" is set to Enabled and "Enable Per Bucket Credentials" is set to Disabled.

Configure a "Default Tiering URL" if you want all users setting bucket lifecycle auto-tiering rules through the CMC to tier to the same tiering destination, using the same account credentials. For example, if you want all users to tier to the same corporate account with Amazon S3 you could set the "Default Tiering URL" to <https://s3.amazonaws.com> -- or if your organization is based in California, USA, <https://s3.us-west-1.amazonaws.com>. Be sure to include the <https://> part when configuring your default tiering URL.

If you use the "Default Tiering URL" feature, you **must** set default tiering credentials.

Also, if you have a multi-region HyperStore system, note that the "Default Tiering URL" will apply to all of your service regions.

Note If you set "Enable Per Bucket Credentials" to Disabled and set a "Default Tiering URL", then **only** your specified "Default Tiering URL" will display for users who are using the CMC to configure auto-tiering for their buckets.

Default Tiering Credentials

This setting displays only if "Enable Auto-Tiering" is set to Enabled and "Enable Per Bucket Credentials" is set to Disabled.

Default tiering credentials (access key and secret key) to use with the "Default Tiering URL". If you configure a default tiering URL you must also configure default tiering credentials. HyperStore will then use those credentials and that tiering URL whenever users configure auto-tiering on their buckets through the CMC.

IMPORTANT ! If you subsequently change your security credentials at the tiering destination -- for example if you change your Amazon or Azure security credentials -- be sure to update the Default Tiering Credentials setting so that HyperStore has the correct current credentials. If you do not keep HyperStore up to date with your current security credentials for the destination system, HyperStore will not be able to access the destination and will not be able to tier your objects to the destination.

S3 Request Restriction Settings

The settings in this section of the CMC's **Configuration Settings** page place restrictions on the behaviors of S3 client applications that interface with the HyperStore system.

S3 Request Restrictions		
Put Object Maximum Size (Bytes)	5368709120	Edit
Multipart Upload Maximum Parts	10000	Edit
Maximum Buckets Per User	100	Edit

Put Object Maximum Size (Bytes)

The maximum single-part object size that S3 clients will be allowed to upload to the system through an S3 *PutObject* request (or *POST Object* request), in number of bytes. If an S3 client submits a *PutObject* or *POST Object* request with an object size larger than this many bytes, the S3 Service rejects the request.

In the case of Multipart Upload operations, this maximum size restriction applies to each individual uploaded part. If a single part is larger than Put Object Maximum Size, the S3 Service rejects the S3 *UploadPart* request.

For best upload performance it's recommended for S3 clients to use the S3 *CreateMultipartUpload* operation for objects larger than 100MB (rather than *PutObject* or *POST Object*).

Default = 5368709120 (5 GiB)

Note In HyperStore version 7.1.x and older, the default was 5000000000 (5 GB rather than 5 GiB). If you upgraded from HyperStore 7.1.x to later versions, your system retains the default of 5000000000 unless you change the setting value.

Multipart Upload Maximum Parts

Maximum number of parts to allow per [CreateMultipartUpload](#) request.

Each individual part can be no larger than the limit configured by the "Put Object Maximum Size (Bytes)" setting.

Default = 10000

Maximum Buckets Per User

Maximum number of S3 storage buckets to allow per user. When a user has this many buckets and tries to create an additional bucket, the request is rejected with an error response.

Default = 100

Auto-Repair Schedule Settings

The settings in this section of the CMC's **Configuration Settings** page configure the scheduling of the HyperStore auto-repair feature.

Auto Repair Schedule		
Replicas Repair Interval (Minutes)	43200	Edit
EC Repair Interval (Minutes)	41760	Edit
Cassandra Full Repair Interval (Minutes)	10080	Edit

Note For more information about the auto-repair feature see "[Automated Data Repair Feature Overview](#)" (page 130)

Replicas Repair Interval (Minutes)

For each HyperStore node, the interval (in minutes) between scheduled auto-repairs of replicated object data (associated with Replication storage policies). The auto-repair feature automatically executes the [hsstool repair](#) operation to repair each node at the configured interval. For each node the repair is for the node's primary token range only, and the system allows auto-repairs to run on multiple nodes concurrently.

Note that when a replica repair operation is running on a target node, the scope of repair activity will extend to other nodes as well. In particular, repair of a target node will also make sure that for objects that fall within the target node's primary token range, the objects' replicas also are present on the other nodes where they are supposed to be.

Auto-repair of the replicated object data in your cluster is a resource-intensive operation and should be configured to run infrequently.

Default = 43200 (every 30 days)

Note If you wish, you can have some or all of the auto-repairs of replica data use the "computedigest" option to combat bit rot. This feature is controlled by the configuration setting "[hyperstore.repair.scheduled.computeDigest.frequency](#)" (page 776). By default "computedigest" is not used in auto-repair runs.

EC Repair Interval (Minutes)

For each HyperStore node, the interval (in minutes) between scheduled auto-repairs of erasure coded object data (associated with EC storage policies). The auto-repair feature automatically executes the [hsstool repairec](#) operation to repair each node at the configured interval.

Note that for erasure coded object data repair, for single data center storage policies and for multi- data center replicated EC storage policies, repair of a target node has the effect of assessing and repairing all erasure coded data on all nodes within the data center where the target node resides. And for multi- data center distributed EC storage policies, repair of a target node has the effect of assessing and repairing all erasure coded data in all of the participating data centers.

EC auto-repair also repairs any replicated object data associated with EC Hybrid storage policies.

EC auto-repair is a resource-intensive operation and should be configured to run infrequently.

Default = 41760 (every 29 days)

Note If you wish, you can have some or all of the auto-repairs of erasure coded data use the "computedigest" option to combat bit rot. This feature is controlled by the configuration setting "[hyperstore.repair.scheduled.computeDigest.frequency](#)" (page 776). By default "computedigest" is not used in auto-repair runs.

Cassandra Full Repair Interval (Minutes)

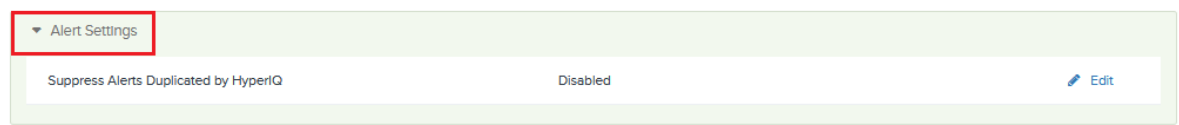
For each HyperStore node, the interval (in minutes) between automatically running a full repair of the node's Cassandra data (Metadata DB) . It is essential to run a full Cassandra data repair on each node at an interval

less than 10 days (less than the Cassandra `gc_grace_seconds` interval, which by default is 10 days). For each node the repair is for the node's primary token range only, and the system allows auto-repairs to run on multiple nodes concurrently.

Note that when a Cassandra repair operation is running on a target node, the scope of repair activity will extend to other nodes as well. In particular, repair of a target node will also make sure that for metadata row keys that fall within the target node's primary token range, the metadata's replicas also are present on the other nodes where they are supposed to be.

Default = 10080 (every seven days)

Alert Settings



Suppress Alerts Duplicated by HyperIQ

This setting is applicable only if you have [Cloudian HyperIQ](#).

The CMC and HyperIQ are both capable of generating alerts based on customizable alert rules (for more about this CMC capability see ["Alert Rules" \(page 560\)](#); for more about this HyperIQ capability see the [Cloudian HyperIQ User's Guide](#)). There is some overlap in the alerts that the CMC and HyperIQ cover. If you enable the "Suppress Alerts Duplicated by HyperIQ" setting, then the CMC will not notify you about alerts that HyperIQ notifies you about. With this setting enabled, alerts that are covered by HyperIQ will not display in the CMC's **Alerts** page and the CMC will not send emails or SNMP traps in regard to such alerts. This keeps you from getting duplicate notifications from the CMC and HyperIQ about the same system events or conditions.

This setting does not apply to alerts triggered by the appearance of warnings or errors in HyperStore application logs, since HyperIQ does not currently monitor for this type of alert. Only the CMC monitors and notifies you about this type of alert.

Default = Disabled (the duplicate suppression function is not turned on)

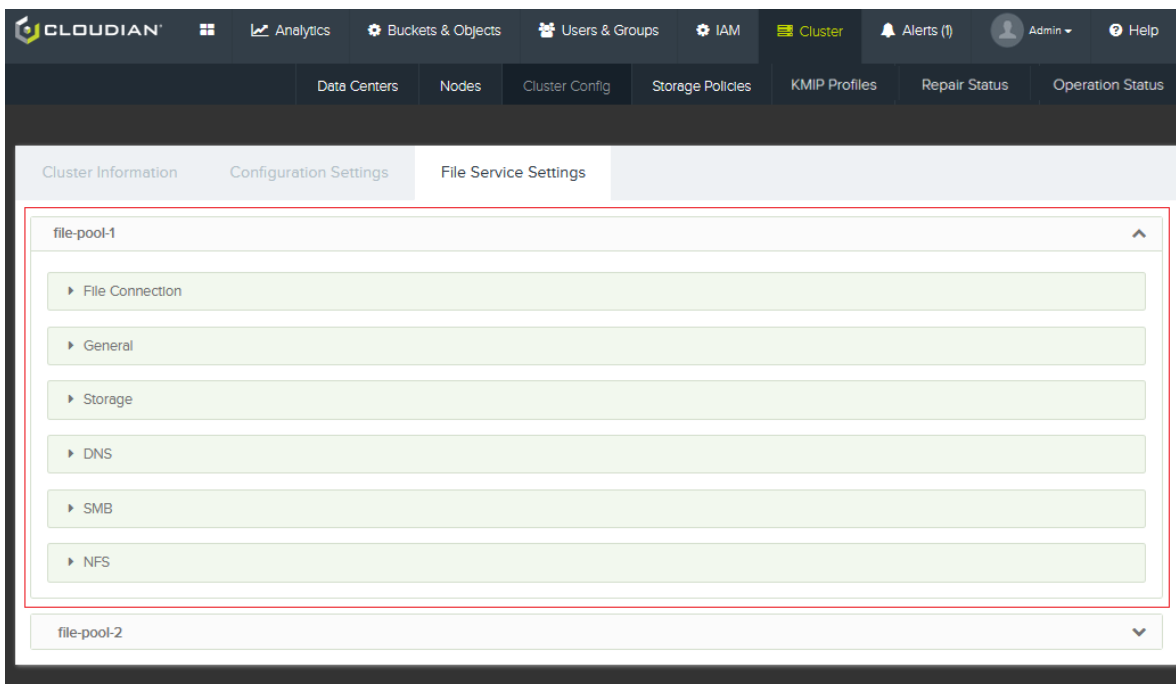
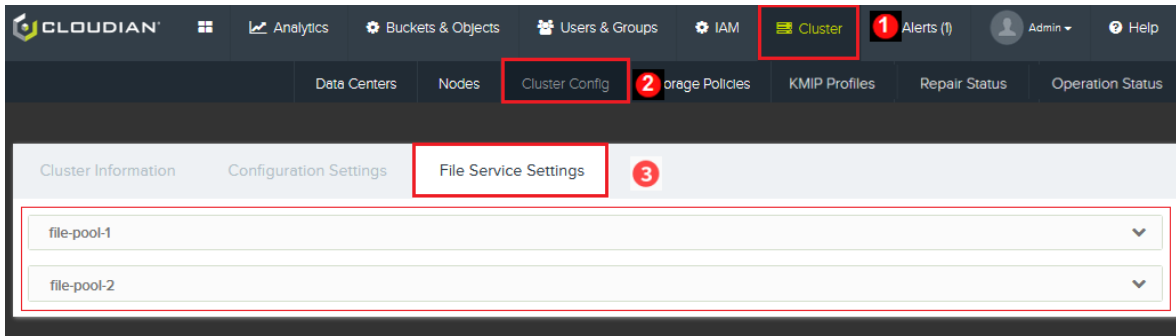
Note: The specific types of alerts that will be suppressed in the CMC if you enable the "Suppress Alerts Duplicated by HyperIQ" setting are listed below. For more information on these alert types see ["Alert Rules" \(page 560\)](#).

- Number of GET transactions per second
- Number of PUT transactions per second
- Throughput for GET operations
- Throughput for PUT operations
- Network throughput (incoming)
- Network throughput (outgoing)
- Disk space available in each device
- Disk error
- CPU utilization
- Admin Service status
- Cassandra Service status

- HyperStore Service status
- Redis QoS Service status
- Redis Credentials Service status
- Redis Monitor Service status
- S3 Service status

6.10.7. File Service Settings

Path: **Cluster** → **Cluster Config** → **File Service Settings**



Supported tasks:

- Configure the HyperStore File Service

If you've set up the HyperStore File Service as part of your HyperStore system, HyperStore buckets can serve as the storage back-end for file shares that SMB or NFS clients can access through the HyperStore File Service. HyperStore system administrators decide which HyperStore users will be allowed to create SMB or NFS files shares that are mapped to those users' own storage buckets. (For overview information about the file ser-

vices feature see ["File Services Feature Overview"](#) (page 238) and ["Preparing the File Services Feature"](#) (page 241). The latter selection includes instructions for enabling file services functionality for selected users.)

If your HyperStore system includes the HyperStore File Service, the CMC displays a **File Service Settings** tab under **Cluster Config**. For most settings in the **File Service Settings** tab you are configuring **system default values** for file services. You can override these system default values with customized settings for individual users for whom you enable file services if you wish (in the **Manage Users** section of the CMC). And also, users for whom you enable file services can override these defaults for their own file shares if they wish.

If you have set up multiple [File Service pools](#) as part of your HyperStore system, then in the CMC's **File Service Settings** page you can configure system default values for each File Service pool independently. In your initial view of the **File Service Settings** page a panel will appear for each File Service pool. Click a pool panel to display the configurable settings for that pool.

Note Making a configuration change to multiple File Service pools in a single action is not supported. If you want to make the same configuration change to multiple File Service pools you must do it separately for each pool.

6.10.7.1. File Service Setting Descriptions

For descriptions of the individual File Service settings see ["File Services Configuration Setting Matrix"](#) (page 384).

6.10.7.2. File Connection

The **File Connection** section of the settings interface shows the status of the CMC's connection to the File Service API. The CMC connects to the File Service API automatically, and the status should show "Connected". The CMC uses this connection to support configuration management for the File Service and also to support HyperStore users' creation and management of file shares.

Typically you should not need to use the **File Connection** section of the settings interface, but in the event that the CMC becomes disconnected from the File Service API and fails to automatically reconnect, you can use the **File Connection** section to re-establish the connection.

The user name for the File Service API connection is "admin" and the password is system-generated when you install the File Service. You can retrieve the current File Service API password by logging into the HyperStore Config Controller node and running this `hsctl` command:

```
hsctl config get cluster.zones.<file-pool-name>.file.auth.password
```

If you wish you can change the password by running this `hsctl` command:

```
hsctl file passwd <PASSWORD> -n <file-pool-name>
```

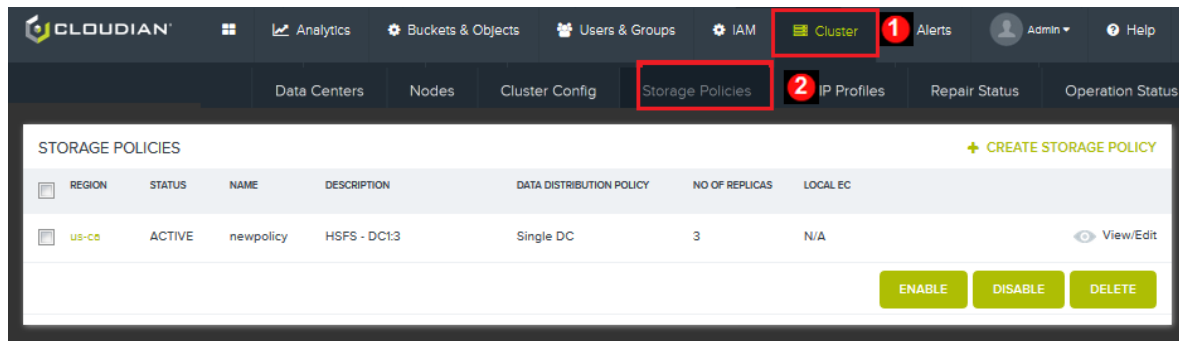
where `<PASSWORD>` is a password of your choosing, that meets these criteria:

- Must be at least 9 characters long and no more than 36 characters
- Must include at least one upper case letter, one number, and one special character

If you have multiple File Service pools each pool has its own File Service API password and its own separate connection from the CMC to the pool's File Service API.

6.10.8. Storage Policies

Path: **Cluster** → **Storage Policies**



Supported tasks:

- ["Add a Storage Policy"](#) (page 513)
- ["Edit a Storage Policy"](#) (page 542)
- ["Designate a Default Storage Policy"](#) (page 543)
- ["Disable a Storage Policy"](#) (page 544)
- ["Delete a Storage Policy"](#) (page 544)

IMPORTANT ! Erasure coding storage policies created with HyperStore versions 5.x or 6.x included a functionality that has been deprecated. The functionality, which identified more than $k+m$ endpoints as possible write destinations for an erasure coded object's $k+m$ fragments, ultimately was found to pose potential problems that outweighed the benefits (and so erasure coding storage policies created in HyperStore 7.0.1 and later returned to using the more conventional approach of identifying only $k+m$ endpoints for erasure coded writes). Support for legacy storage policies with the deprecated functionality will end in a future HyperStore release (the exact timing is still to be determined). Starting with HyperStore 7.5.2, the system automatically detects if you have any of these deprecated storage policies in your system. If you do have any of these deprecated storage policies, then:

* A warning will display in the CMC **Dashboard**. The warning will indicate that you have one or more deprecated storage policies in your system and that you will need to contact your Clouidian Sales Representative to plan for migrating data from those deprecated storage policies to supported storage policies.

* In the CMC's **Storage Policies** page -- which lists all your storage policies -- a warning symbol will display next to the names of the deprecated storage policy or policies. Hover text shows the warning description.

* Users creating new buckets will not be able to use a deprecated storage policy (such policies will not display in the list of available storage policies that users see when creating a new bucket).

Note For detailed information on S3 write and read availability under various combinations of cluster size, storage policy configuration, and number of nodes down, see ["Storage Policy Resilience to Downed Nodes"](#) (page 156).

6.10.8.1. Add a Storage Policy

Storage policies are ways of protecting data so that it's durable and highly available to users. The HyperStore system lets you configure one or more storage policies. When users create a new storage bucket they choose which of your configured storage policy to use to protect data in that bucket. **Users cannot create buckets until you have created at least one storage policy.**

For each storage policy that you create you can choose and configure either of two data protection methods: replication or erasure coding. If your HyperStore system spans multiple data centers, for each storage policy you can also choose how data is allocated across your data centers.

Before creating a storage policy:

- You may want to review the ["Storage Policies Feature Overview" \(page 148\)](#) if you haven't already done so.
- If you intend to create a storage policy that includes support for object metadata search, first review the ["Object Metadata and Search Feature Overview" \(page 230\)](#) and complete the preparation steps described in ["Preparing the Metadata Search Feature" \(page 236\)](#).
- If you intend to create a storage policy that includes server-side encryption, first review the ["Server-Side Encryption" \(page 186\)](#) and complete the preparation steps described in ["Using Regular SSE" \(page 188\)](#) or (if applicable to your use case) ["Using SSE-KMIP" \(page 191\)](#).

While creating a storage policy, keep in mind:

- In your system you must have a [default storage policy](#). If you have not yet created any storage policies, the first policy that you create must be configured to be visible to all groups, and it will automatically become the default policy. Subsequently, after creating additional policies, you can designate a different policy as the default policy if you wish.
- If you have a multi-region system, each region must have its own storage policy or policies. When you create storage policies, while configuring each policy's "Data Center Assignment" you will specify the region with which the policy is associated. Each region must have a default storage policy.
- By default the system allows a maximum of 25 storage policies. This maximum is controlled by the configuration setting ["hyperstore.storagePolicy.max" \(page 779\)](#). Edit this setting if you want to have more than 25 storage policies in your HyperStore system.

To add a storage policy:

1. In the CMC's [Storage Policies](#) page click **Create Storage Policy**. This opens the **Create New Policy** interface.

STORAGE POLICIES

+ CREATE STORAGE POLICY

CREATE NEW POLICY

Policy Name

Policy Name

Policy Description

Policy Description

☐ Enable Metadata Search

NUMBER OF DATACENTERS

1

DATA DISTRIBUTION SCHEME

☒ Replicas Within Single Datacenter

☐ EC Within Single Datacenter





NUMBER OF REPLICAS

3

DATACENTER ASSIGNMENT

REGION	DATACENTER	REPLICA	LOCAL EC
region	DC1	1 of 3	
		2 of 3	disabled
		3 of 3	

CONSISTENCY SETTING

CONSISTENCY LEVEL	READ	WRITE
ALL	☐	☐
QUORUM	☒	☒
ONE	☐	☐

GROUP VISIBILITY

Please select a Group

ADD

Compression Type

NONE

Server-side Encryption

NONE

Workload Type

DEFAULT

SAVE

CANCEL

2. In the **Create New Policy** interface set your desired storage policy configuration:

- [Policy Name and Description](#)
- [Metadata Search \(optional\)](#)
- [Data Distribution Scheme and Data Center Assignment](#)
- [Consistency Setting](#)
- [Group Visibility](#)
- [Compression \(optional\)](#)
- [Server-Side Encryption \(optional\)](#)

- [Workload Type](#)
 - Additional Settings That Display Only For EC Storage Policies
 - [Hybrid Policy Threshold](#)
 - [Hybrid Policy RF Count](#)
 - [Total Metadata Replicas](#)
3. Click **Save** to create the new storage policy in the system. The policy will then appear in your storage policy list in the CMC's **Storage Policies** page. The new policy will initially show a status of PENDING, but if you check again a few minutes later it should have a status of ACTIVE, at which point the policy is available to users.

Note In the unlikely event that the new policy fails to be created in the system, in your policy list the policy will appear with status FAILED. In this case you can delete the failed policy and then try again to configure and save a new policy with your desired settings.

6.10.8.2. Create Storage Policy -- Policy Name and Description

Enter a Policy Name (only letters, numbers, dashes, and underscores are allowed; maximum 32 characters) and also a Policy Description that will be meaningful to your users (maximum 64 characters). The Policy Name and Policy Description are important because users will see this text when they are choosing a storage policy to apply to a newly created storage bucket.

How End Users Will See the Policy Name and Description

In the CMC's **Buckets** interface, a user creating a bucket will see a drop-down menu listing all the pre-configured storage policies by name, and when they select a policy they will see its description (before they complete the process of creating the bucket). So, create policy names and descriptions that will be meaningful to your service users.

For example, suppose your HyperStore spans two data centers, one in Chicago and one in Kansas City. You create one storage policy that uses replication and stores the data in both of your data centers, and another policy that uses erasure coding and stores data only in Kansas City. If the people who will be creating storage buckets are fairly technical, then in creating these two policies you might use names and descriptions like:

Policy Name	Policy Description
Replication_Chicago-3X_KC-2X	3 replicas stored in Chicago and 2 in Kansas City
Erasure_coding_KC-only	Objects are erasure coded and stored in Kansas City only

If your users are non-technical, you might gear your policy descriptions more towards the type of data that users intend to store. Since replication is commonly used for more active data while erasure coding is suitable to "cold" data, you might do something like:

Policy Name	Policy Description
Active_High-Availability	For "live" data that is accessed often
Cold_Archival_Storage	For "cold" data that is rarely accessed

Note that if your HyperStore system has multiple service regions and/or multiple data centers, it may be important to communicate data storage location information to end users through the Policy Name and Policy Description — especially if your system spans multiple countries.

6.10.8.3. Create Storage Policy -- Enable Metadata Search

Policy Name Policy Name	Policy Description Policy Description
☐ Enable Metadata Search	

If you have set up the HyperSearch Service in your HyperStore system, then in the [storage policy creation interface](#) an "Enable metadata search" checkbox will appear directly below the "Policy Name" field. Select this checkbox if you want to enable object metadata search for this storage policy. If metadata search is enabled for the storage policy, then users who create buckets that use this storage policy will be able to use the CMC to perform metadata-based searches for objects in those buckets. This includes searching by user-defined object metadata.

Note The CMC does not support the creation of user-defined object metadata. But if users create object metadata with a third party S3 client application, then they will be able to use the CMC to search for objects based on that metadata -- if their buckets use a storage policy for which metadata search is enabled.

For more details about the object metadata search feature see:

- ["Object Metadata and Search Feature Overview"](#) (page 230)
- ["Preparing the Metadata Search Feature"](#) (page 236)

6.10.8.4. Create Storage Policy -- Data Distribution Scheme and Data Center Assignment

In the "Data Distribution Scheme" and "Data Center Assignment" sections of the [storage policy creation interface](#) choose and configure a scheme for protecting S3 object data and object metadata.

First, from the "Number of Data Centers" drop-down list, choose the **number of data centers in which this new policy should store data**. If you want you can specify a number here that's smaller than your total number of data centers -- for example, if you have three DCs in your system but you want to create a policy that stores data in only two of those DCs, then select 2 here. Later in the policy creation process you will be able to specify exactly which DCs to use for this policy.

Next, choose one of the following data distribution schemes. Only the schemes appropriate to your specified number of DCs will display in the interface.

Note For detailed information on S3 write and read availability under various combinations of cluster size, data distribution scheme, consistency level settings, and number of nodes down, see ["Storage Policy Resilience to Downed Nodes"](#) (page 156).

Replicas Within Single Data Center

NUMBER OF DATACENTERS

1

DATA DISTRIBUTION SCHEME

☒ Replicas Within Single Datacenter
☐ EC Within Single Datacenter

NUMBER OF REPLICAS

3

DATACENTER ASSIGNMENT

REGION	DATACENTER	REPLICA	LOCAL EC
region1	DC1	1 of 3	disable
		2 of 3	
		3 of 3	

This scheme protects S3 object data by replicating it within a single data center.

When you choose this option, the interface displays a "Number of Replicas" field in which you can specify how many replicas you want. For example, if you specify 3, then with this storage policy the system will maintain a total of 3 copies of each S3 data object, each on a different node.

The system will also maintain the same number of replicas of each S3 object's metadata (in Cassandra), again with each replica on a different node. The object metadata key has its own hash value (token) and thus HyperStore's token-based data distribution mechanism may allocate the object metadata to different nodes than the object data.

You **cannot choose a number of replicas that is greater than the number of HyperStore nodes in the data center**. For example, if you have only 3 nodes in the data center, then 3 is the maximum number of replicas that you can choose.

Also, you **cannot choose fewer than 3 replicas** unless you have only 1 or 2 nodes in your system (such as when doing simple evaluation or testing of HyperStore). In a production environment, 1X or 2X replication provide inadequate data protection and consequently those settings are not allowed.

If you have only one DC in your system, then there is nothing that you need to do in the "Data Center Assignment" section of the interface. If you have more than one DC in your system, use the "Data Center Assignment" section to select which one of your DCs to use for this policy.

EC Within Single Data Center

NUMBER OF DATACENTERS

1

DATA DISTRIBUTION SCHEME

☐ Replicas Within Single Datacenter
 ☒ EC Within Single Datacenter

ERASURE CODING K+M VALUE

4+2

DATACENTER ASSIGNMENT

REGION	DATACENTER	REPLICA	LOCAL EC
region1	DC1	1 of 1	4+2

This scheme protects S3 object data by erasure coding the data, within a single data center.

When you choose this option, the interface lets you select which of the supported EC "k"+"m" configuration schemes to use. For example:

- **4+2** — Each object will be encoded into 4 data fragments plus 2 parity fragments, with each fragment stored on a different node. Objects can be read so long as any 4 of the 6 fragments are available.
- **6+2** — Each object will be encoded into 6 data fragments plus 2 parity fragments, with each fragment stored on a different node. Objects can be read so long as any 6 of the 8 fragments are available.

In addition to the options above, the system by default also supports:

- **8+2**
- **9+3**
- **12+4**

The choice among these supported EC configurations is largely a matter of how many HyperStore nodes you have in the data center. For example, compared to a 4+2 configuration, 6+2 EC provides the same degree of data availability assurance (stored objects can be read even if 2 of the involved nodes are unavailable), while delivering a higher level of storage efficiency based on the parity fragments as a percentage of the data fragments (4+2 incurs 50% overhead whereas 6+2 incurs only 33% overhead). So 6+2 may be preferable to 4+2 if you have at least 8 HyperStore nodes in the data center.

Likewise, 9+3 EC provides a higher degree of protection and availability than 6+2 EC (since with 9+3 EC, stored objects can be read even if 3 of the involved nodes are unavailable) while delivering the same level of storage efficiency (both 6+2 and 9+3 incur 33% storage overhead). So 9+3 may be preferable to 6+2 if you have at least 12 HyperStore nodes in the data center.

You **cannot choose a $k+m$ configuration that totals to more than the number of HyperStore nodes in the data center**. For example, if you have 7 nodes in the data center, you can choose 4+2 as your $k+m$ configuration, but not 6+2.

If you have only one DC in your system, then there is nothing that you need to do in the "Data Center Assignment" section of the interface. If you have more than one DC in your system, use the "Data Center Assignment" section to select which one of your DCs to use for this policy.

Note With erasure coding storage policies, the object data is erasure coded but the object metadata is protected by replication rather than erasure coding (this is because each object's metadata is very small in size and so erasure coding is inappropriate for metadata). As part of storage policy creation you can configure the object metadata replication factor with the [Total Metadata Replicas](#) setting.

Note If you want to use a $k+m$ configuration other than those listed in the CMC interface, contact Cloudian Support or your Cloudian Sales representative to see whether your desired configuration can be supported.

Replication Across Data Centers

NUMBER OF DATACENTERS

2

DATA DISTRIBUTION SCHEME

☒ Replication Across Datacenters

☐ Replicated EC

NUMBER OF REPLICAS

5

DATACENTER ASSIGNMENT

REGION	DATACENTER	REPLICA	LOCAL EC
region1	DC1	1 of 5	disable
	DC1	2 of 5	
	DC1	3 of 5	
	DC1	4 of 5	
	DC1	5 of 5	

This distribution scheme is supported only for HyperStore systems that span multiple data centers. Choose this option if you want S3 objects to be replicated and to have those replicas distributed across DCs.

When you choose this option, the interface displays a "Number of Replicas" field in which you can specify the **total** number of replicas that you want, across multiple data centers. For example, if for each S3 object you want 3 replicas stored in DC1 and 2 replicas in DC2, enter 5 in the "Number of Replicas" field.

You can then use the "Data Center" drop-down lists to select which one of your data centers will house **each replica**. This interface allows you to define how many replicas will be stored in each specific data center. In the example below, 3 replicas will be stored in DC1 and 2 replicas will be stored in DC2.

DATACENTER ASSIGNMENT			
REGION	DATACENTER	REPLICA	LOCAL EC
region1	DC1	1 of 5	disable
	DC1	2 of 5	
	DC1	3 of 5	
	DC2	4 of 5	
	DC2	5 of 5	

Note that the ordering of the replicas doesn't matter — what matters is the total number of replicas that you assign to each data center. (If you have a multi-region system, first choose a region then choose the DCs from within that region).

Note The system will also maintain the same number of replicas of each S3 object's metadata (in Cassandra), with the same distribution across data centers. The object metadata key has its own hash value (token) and thus HyperStore's token-based data distribution mechanism may allocate the object metadata to different nodes than the object data.

Replicated EC

NUMBER OF DATACENTERS

2

DATA DISTRIBUTION SCHEME

Replication Across Datacenters

OBJECT

↓

REPLICAS

DC-1

DC-2

...

DC-n

Replicated EC

OBJECT

↓

REPLICAS

↓

EC FRAGMENTS

DC-1

DC-2

...

DC-n

ERASURE CODING K+M VALUE

4+2

DATACENTER ASSIGNMENT

Please select the correct amount of DCs.

REGION	DATACENTER	SELECTED
region1	DC1	☐
	DC2	☐
	DC3	☐

With Replicated EC, each object is encoded into "k" data fragments + "m" parity fragments and that set of "k"+"m" fragments is replicated in multiple data centers. With this approach, each participating data center stores the full set of an object's erasure coded fragments (each participating data center stores "k" + "m" fragments). An object can be read so long as a combined total of at least "k" fragments are available across the participating DCs.

In the "Erasure Coding k+m Value" drop-down list you can choose from among these options that the system supports by default:

- 4+2
- 6+2
- 8+2
- 9+3
- 12+4

Next, in the "Data Center Assignment" section of the interface choose which of your DCs you want to participate in this storage policy. (If you have a multi-region system, first choose a region then choose the DCs from within that region).

Note that:

- For this type of distribution scheme, each participating data center will use the same "k"+"m" configuration (the configuration you selected from the "Erasure Coding k+m Value" drop-down list). You cannot create a policy that uses 4+2 in one data center while using 6+2 in a different data center, for instance.
- In each participating data center, for a given object each of the "k"+"m" erasure coded fragments will be stored on a different node. Therefore you cannot choose a "k"+"m" configuration that exceeds the number of nodes in any participating data center. For example, if one of the data centers that you want to use for the storage policy has 8 nodes and the second data center has 6 nodes, you can use replicated 4+2 erasure coding but not replicated 6+2 erasure coding (since the second data center doesn't have enough nodes to support 6+2 erasure coding).
- As compared to the "EC Across Data Centers" option, the "Replicated EC" option:
 - Is less efficient in using storage capacity, since there is more storage overhead per stored object. This is because for each object each DC will have a full "k"+"m" set of fragment replicas, whereas for EC Across Data Centers there is just one "k"+"m" set of fragments which is spread across multiple DCs. Mathematically, the storage overhead percentage (redundant data size as a percentage of original data size) for Replicated EC is $\#DCs(m/k)$ whereas for EC Across Data Centers it's m/k .
 - Reduces the chance of objects being unavailable, since there are additional redundant fragments and objects are available so long as a total of "k" fragments are readable across the participating data centers. For example, with 4+2 EC replicated in two DCs there are a total of 12 stored fragments and objects remain readable even if 8 of those fragments are unreachable. By contrast, for EC Across Data Centers an object will be unreadable if more than "m" endpoints in total are unavailable across all the participating DCs.
 - Reduces read latency since typically an object will be decodable (readable) from the fragments within the local data center (the data center processing the S3 request from a client application).

Note With erasure coding storage policies, the object data is erasure coded but the object metadata is protected by replication rather than erasure coding (this is because each object's metadata is very small in size and so erasure coding is inappropriate for metadata). As part of storage policy creation you can configure the object metadata replication factor with the [Total Metadata Replicas](#) setting.

Note If you want to use a $k+m$ configuration other than those listed in the CMC interface, contact Cloudian Support or your Cloudian Sales representative to see whether your desired configuration can be supported.

EC Across Data Centers

NUMBER OF DATACENTERS

3

DATA DISTRIBUTION SCHEME

Replication Across Datacenters

Replicated EC

EC Across Datacenters

ERASURE CODING K+M VALUE

7+5

DATACENTER ASSIGNMENT
Please select the correct amount of DCs.

REGION	DATACENTER	SELECTED
region1	DC1	☐
	DC2	☐
	DC3	☐

With EC Across Data Centers (also known as "Distributed EC"), each object is encoded into "k" data fragments + "m" parity fragments and then that one set of "k"+"m" fragments is spread out evenly across the participating data centers. An object can be read so long as a combined total of at least "k" fragments are available across the multiple DCs.

In the "Erasure Coding k+m Value" drop-down list you can choose your desired "k"+"m" configuration. Your options are determined by the number of DCs that you specified in the "Number of Data Centers" drop-down list. The table below shows the default supported options and how the fragments will be distributed.

# of Participating DCs	Supported "k"+"m"	How Fragments Will Be Distributed
3	5+4	3 fragments per DC
	7+5	4 fragments per DC
4	8+4	3 fragments per DC
5	6+4	2 fragments per DC
6	8+4	2 fragments per DC
	7+5	2 fragments per DC
7	10+4	2 fragments per DC
8	10+6	2 fragments per DC
9	10+8	2 fragments per DC

After selecting your "k"+"m" configuration, use the "Data Center Assignment" section of the interface to choose which of your DCs you want to participate in this storage policy. (If you have a multi-region system, first choose a region then choose the DCs from within that region).

Note that:

- You must have at least 3 data centers within a region to use this type of data distribution scheme.
- The number of nodes in each participating DC must be at least as many as the number of fragments that will be distributed to each DC. For example, to use 7+5 erasure coding spread across 3 data centers you need at least 4 nodes in each of those DCs (since each DC will be allocated 4 fragments from each S3 object and each fragment must be stored on a different node).
- The supported options are such that objects are readable even if one of the participating DCs goes down, so long as a sufficient number of endpoints are live in the remaining DCs. (Put differently, in all the supported options the number of fragments per DC is never greater than "m".)
- As compared to the "Replicated EC" option, the "EC Across Data Centers" option:
 - Is more efficient in using storage capacity, since there is less storage overhead per stored object. This is because there is just one "k"+"m" set of fragments which is spread across multiple DCs, rather than each DC having a full "k"+"m" set of fragments (as is the case with Replicated EC). Mathematically, the storage overhead percentage (redundant data size as a percentage of original data size) for EC Across Data Centers is m/k whereas for Replicated EC it's $\#DCs(m/k)$.
 - Entails a somewhat higher chance of objects being unavailable, since for a given object if more than a total of "m" endpoints (nodes on which the object's fragments are stored) are down across the multiple DCs, the system will not be able to decode the object. By contrast, with Replicated EC there is additional redundancy of fragments and a greater percentage of an object's fragments can be lost or unreachable without impacting the object's readability.
 - Entails a somewhat higher read latency since all object reads will necessarily require communication across DCs (since no one DC has the entire "k" set of fragments that's required for an object read).

Note With erasure coding storage policies, the object data is erasure coded but the object metadata is protected by replication rather than erasure coding (this is because each object's metadata is very small in size and so erasure coding is inappropriate for metadata). As part of storage policy creation you can configure the object metadata replication factor with the [Total Metadata Replicas](#) setting.

Note If you want to use a $k+m$ configuration other than those listed in the CMC interface, contact Cloudian Support or your Cloudian Sales representative to see whether your desired configuration can be supported.

6.10.8.5. Create Storage Policy -- Consistency Setting

In the "Consistency Setting" section of the [storage policy creation interface](#) configure data consistency levels to apply to this storage policy. Consistency levels impose requirements as to **what portion of the data and metadata reads or writes associated with a given S3 request must be successfully completed within the cluster before the system can return a success response to the S3 client**. If the consistency requirements cannot be met for a given S3 request at a given time -- for example, due to one or more endpoint nodes being inaccessible -- an HTTP 503 error response is returned to the client.

Note For detailed information on S3 write and read availability under various combinations of cluster size, data distribution scheme, and consistency level settings, see "[Storage Policy Resilience to Downed Nodes](#)" (page 156).

Your consistency level options depend on which data distribution scheme you chose for the policy you are creating:

Replicas Within Single Data Center

With a "Replicas within Single Data Center" scheme you can choose from the following consistency levels.

Read	Write
ALL	ALL
QUORUM (default)	QUORUM (default)
ONE	--

EC Within Single Data Center

With an "EC within Single Data Center" scheme you can choose from the following consistency levels.

Read (metadata only)	Write
ALL	ALL
QUORUM (default)	QUORUM (default)

Note For this type of scheme the consistency requirement for reading object data is always "k" unique fragments. Your "Read" consistency setting impacts only the reading of object metadata.

Replication Across Data Centers

With a "Replicas Across Data Centers" scheme you can choose from the following consistency levels.

Read	Write
ALL	ALL
--	EACH QUORUM
QUORUM (default)	QUORUM (default)
LOCAL QUORUM	LOCAL QUORUM
ONE	--

Replicated EC

With a "Replicated EC" scheme you can choose from the following consistency levels.

Read	Write
ALL	ALL
--	EACH QUORUM
LOCAL QUORUM	LOCAL QUORUM

Read	Write
ANY QUORUM (default)	ANY QUORUM (default)

Note For this type of scheme the consistency requirement for reading object data is always "k" unique fragments. If you choose "Local Quorum" for read, the system will only read from the local data center when trying to get "k" unique fragments and a local quorum of object metadata. With the other settings the system will read from all data centers when trying to get "k" unique fragments and a quorum of object metadata.

EC Across Data Centers

With an "EC Across Data Centers" scheme you can choose from the following consistency levels.

Read (metadata only)	Write
ALL	ALL
QUORUM (default)	QUORUM (default)

Note For this type of scheme the consistency requirement for reading object data is always "k" unique fragments. Your "Read" consistency setting impacts only the reading of object metadata.

Note When configuring consistency levels you would typically choose just one consistency level for each operation type, by selecting one consistency level checkbox for Read and one for Write. However, HyperStore also supports an advanced option known as "[Dynamic Consistency Levels](#)" (page 80), whereby you can configure both a primary consistency level and a fallback consistency level to be used in instances when the primary consistency level cannot be achieved. In the CMC interface you can do this by selecting more than one consistency level checkbox for a given operation type.

6.10.8.6. Consistency Levels

To boost data durability and availability, HyperStore implements replication or erasure coding for object data and replication for object metadata. This entails distributing each object's data and metadata to multiple end-point nodes across the cluster. When you create storage policies, along with configuring a replication or erasure coding scheme you will also configure consistency levels for writes and reads. Consistency levels impose requirements as to **what portion of the endpoint writes or reads associated with each S3 request must be successfully completed before the system can return a success response** to the S3 client. If the consistency requirements cannot be met for a given S3 request at a given time -- for example, due to one or more endpoint nodes being inaccessible -- an HTTP 503 error response is returned to the client.

Below is the full list of consistency levels supported by the HyperStore system. Your consistency level options when configuring a particular storage policy will be limited by the data distribution scheme (replication or erasure coding, single DC or multi-DC) that you have selected for that policy.

- [ALL](#)
- [QUORUM](#)
- [EACH QUORUM](#)
- [LOCAL QUORUM](#)

- [ANY QUORUM](#)
- [ONE](#)

Note In the case of writes, if the consistency requirement is met by something less than completing writes of all replicas (or all erasure coded fragments), then after returning a success response to the client the system continues to try to complete the remaining writes. If any of these writes fail they will later be recreated by [automatic data repair](#).

Note As an advanced option you can also configure "dynamic" consistency levels, whereby the system will try to achieve a "fallback" consistency level if the primary consistency level cannot be achieved. For more information see ["Dynamic Consistency Levels"](#) (page 80).

6.10.8.7. Consistency Level "ALL"

The consistency level "ALL" is a supported option for every type of data distribution scheme, for both reads and writes. The table below shows the general requirements of ALL in the context of replication based storage policies and erasure coding based storage policies. Following the table are examples for each supported data distribution scheme.

	Replication	Erasure Coding
Read ALL	For an S3 GET to succeed, the system must succeed in reading all object data replicas and all object metadata replicas.	For an S3 GET to succeed, the system must succeed in reading "k" object data fragments and all object metadata replicas.
Write ALL	For an S3 PUT to succeed, the system must succeed in writing all object data replicas and all object metadata replicas.	For an S3 PUT to succeed, the system must succeed in writing all ("k"+"m") object data fragments and all object metadata replicas.

Example for a "Replication Within Single Data Center" policy

Suppose a "Replication Within Single Data Center" policy uses 3X replication. With this configuration, for each object the system will store 3 object data replicas and 3 object metadata replicas.

If the policy's consistency level for Read is set to ALL, then during an S3 GET operation all 3 [object data replicas](#) and all 3 object metadata replicas must be successfully read before the object is returned to the client.

If the consistency level for Write is set to ALL, then during an S3 PUT operation all 3 object data replicas and all 3 object metadata replicas must be successfully written before a success response is returned to the client.

Example for an "EC Within Single Data Center" policy

Suppose an "EC Within Single Data Center" policy uses 4+2 erasure coding. With this configuration, for each object the system will store 6 object data fragments and 3 object metadata replicas (2m-1, by default).

If the policy's consistency level for Read is set to ALL, then during an S3 GET operation 4 object data fragments and all 3 object metadata replicas must be successfully read before the object is returned to the client.

If the consistency level for Write is set to ALL, then during an S3 PUT operation all 6 object data fragments and all 3 object metadata replicas must be successfully written before a success response is returned to the client.

Example for a "Replication Across Data Centers" policy

Suppose a "Replication Across Data Centers" policy uses 3X replication in DC-East and 2X replication in DC-West. With this configuration, for each object the system will store 3 object data replicas and 3 object metadata replicas in DC-East, and 2 object data replicas and 2 object metadata replicas in DC-West.

If the policy's consistency level for Read is set to ALL, then during an S3 GET operation all 5 [object data replicas](#) and all 5 object metadata replicas must be successfully read before the object is returned to the client.

If the consistency level for Write is set to ALL, then during an S3 PUT operation all 5 object data replicas and all 5 object metadata replicas must be successfully written before a success response is returned to the client.

Example for a "Replicated EC" policy

Suppose a "Replicated EC" policy uses 4+2 erasure coding replicated in DC-East and DC-West. With this configuration, for each object the system will store 6 object data fragments and 3 object metadata replicas (2m-1, by default) in each of those DCs.

If the policy's consistency level for Read is set to ALL, then during an S3 GET operation a total of 4 unique object data fragments (from among all the object's fragments in the two DCs) as well as all 6 object metadata replicas must be successfully read before the object is returned to the client.

If the consistency level for Write is set to ALL, then during an S3 PUT operation all 12 object data fragments and all 6 object metadata replicas must be successfully written before a success response is returned to the client.

Example for an "EC Across Data Centers" policy

Suppose an "EC Across Data Centers" policy uses 7+5 erasure coding distributed across DC-East, DC-West, and DC-South. With this configuration, for each object the system will store 4 object data fragments in each of those DCs, as well as 9 object metadata replicas (2m-1, by default) spread approximately evenly across the DCs.

If the policy's consistency level for Read is set to ALL, then during an S3 GET operation a total of 7 unique object data fragments (from among all the object's fragments in the three DCs) as well as all 9 object metadata replicas must be successfully read before the object is returned to the client.

If the consistency level for Write is set to ALL, then during an S3 PUT operation all 12 object data fragments and all 9 object metadata replicas must be successfully written before a success response is returned to the client.

6.10.8.8. Consistency Level "ANY QUORUM"

The consistency level "ANY QUORUM" is supported only in "Replicated EC" data distribution schemes, for read and write operations. The table below shows the general requirements of ANY QUORUM, and following the table is an example.

	Replicated EC
Read ANY QUORUM	For an S3 GET to succeed the system must succeed in reading a total of "k" unique erasure coded fragments from among the storage policy's participating data centers, and also reading enough object metadata replicas to satisfy either a LOCAL QUORUM or QUORUM consistency level (implemented as dynamic consistency levels). The system automatically sets up this object metadata consistency configuration when you choose ANY QUORUM as the read consistency for the storage policy, because Cassandra (in which object metadata is stored) does not natively support an ANY QUORUM consistency level.

	Replicated EC
Write ANY QUORUM	For an S3 PUT to succeed the system must succeed in writing "k"+1 erasure coded fragments within any one of the storage policy's participating data centers and also writing enough object metadata replicas to satisfy either a LOCAL QUORUM or QUORUM consistency level

Example for a "Replicated EC" policy

Suppose a "Replicated EC" policy uses 4+2 erasure coding replicated in DC-East and DC-West. With this configuration, for each object the system will store 6 object data fragments and 3 object metadata replicas (2m-1, by default) in each of those DCs.

If the policy's consistency level for Read is set to ANY QUORUM, and an S3 GET request is received in DC-East (for example), then in order for that request to succeed the system must succeed in reading 4 unique object data fragments, and either 2 object metadata replicas in DC-East or a total of 4 object metadata replicas from the two DCs combined.

Note For the object data, reading 3 object data fragments in DC-East (for example) and 1 object data fragment in DC-West would satisfy the requirement as long as all those fragments are unique.

If the consistency level for Write is set to ANY QUORUM, and an S3 PUT request is received in DC-East, then in order for that request to succeed the system must succeed in writing 5 object data fragments either in DC-East or in DC-West, and writing either 2 object metadata replicas in DC-East or a total of 4 object metadata replicas in the two DCs combined.

Note For the object data, writing 3 object data fragments in DC-East (for example) and 2 object data fragments in DC-West would **not** satisfy the requirement, since the object data write quorum has to be achieved **within** one of the DCs.

6.10.8.9. Consistency Level "EACH QUORUM"

The consistency level "EACH QUORUM" is supported only in "Replication Across Data Centers" or "Replicated EC" data distribution schemes and only for write operations -- not reads. The table below shows the general requirements of EACH QUORUM in the context of replication based storage policies and erasure coding based storage policies. Following the table are examples for each supported data distribution scheme.

	Replication	Erasure Coding
Write EACH QUORUM	For an S3 PUT to succeed, in each of the storage policy's participating data centers the system must succeed in writing a majority of object data replicas and a majority of object metadata replicas.	For an S3 PUT to succeed, in each of the storage policy's participating data centers the system must succeed in writing "k"+1 object data fragments and a majority of object metadata replicas.

Note For replicated data or metadata, a quorum or majority of a set of replicas is defined mathematically as $(\#replicas/2) + 1$, rounded down to the nearest integer. For example with 3 replicas a quorum would be $(3/2 = 1.5) + 1 = 2.5$, rounded down = 2. With 4 replicas a quorum would be $(4/2 = 2)$

+1 = 3, rounded down = 3. With the EACH QUORUM consistency level the quorum must be achieved in each participating DC.

Example for a "Replication Across Data Centers" policy

Suppose a "Replication Across Data Centers" policy uses 3X replication in DC-East and 2X replication in DC-West. With this configuration, for each object the system will store 3 object data replicas and 3 object metadata replicas in DC-East, and 2 object data replicas and 2 object metadata replicas in DC-West.

If the policy's consistency level for Write is set to EACH QUORUM, then in order for an S3 PUT operation to succeed the system must succeed in writing 2 object data replicas and 2 object metadata replicas in DC-East, **and** 2 object data replicas and 2 object metadata replicas in DC-West.

Example for a "Replicated EC" policy

Suppose a "Replicated EC" policy uses 4+2 erasure coding replicated in DC-East and DC-West. With this configuration, for each object the system will store 6 object data fragments and 3 object metadata replicas (2m-1, by default) in each of those DCs.

If the policy's the consistency level for Write is set to EACH QUORUM, then in order for an S3 PUT operation to succeed the system must succeed in writing 5 object data fragments and 2 object metadata replicas in DC-East, **and** 5 object data fragments and 2 object metadata replicas in DC-West.

6.10.8.10. Consistency Level "LOCAL QUORUM"

The consistency level "LOCAL QUORUM" is supported only in "Replication Across Data Centers" or "Replicated EC" data distribution schemes, for write and read operations. With LOCAL QUORUM it's important to understand these two points:

- In the context of a storage policy that encompasses multiple data centers, for any given S3 request **the "local" data center is the data center in which the request is received from the S3 client**. For example in a storage policy that encompasses DC1 and DC2, for all S3 requests received in DC1 the local data center is DC1, and for all S3 requests received in DC2 the local data center is DC2.
- With a LOCAL QUORUM consistency level, whether an S3 request succeeds or not is based **solely on what happens in the local data center**. What happens in the remote data center(s) is irrelevant to achieving a LOCAL QUORUM consistency level.

With those points in mind, the table below shows the general requirements of LOCAL QUORUM in the context of replication based storage policies and erasure coding based storage policies. Following the table are examples for each supported data distribution scheme.

	Replication	Erasure Coding
Read LOCAL QUORUM	For an S3 GET to succeed, the system must succeed in reading a majority of the local DC's object data replicas and a majority of the local DC's object metadata replicas.	For an S3 GET to succeed, the system must succeed in reading "k" object data fragments within the local DC and a majority of the local DC's object metadata replicas.
Write LOCAL QUORUM	For an S3 PUT to succeed, the system must succeed in writing a majority of the local DC's object data replicas and a	For an S3 PUT to succeed, the system must succeed in writing "k"+1 object data fragments within the local DC and a

	Replication	Erasure Coding
	majority of the local DC's object metadata replicas.	majority of the local DC's object metadata replicas.

Note For replicated data or metadata, a quorum or majority of a set of replicas is defined mathematically as $(\#replicas/2) + 1$, rounded down to the nearest integer. For example with 3 replicas a quorum would be $(3/2 = 1.5) + 1 = 2.5$, rounded down = 2. With 4 replicas a quorum would be $(4/2 = 2) + 1 = 3$, rounded down = 3. With the LOCAL QUORUM consistency level the requirement is to successfully read or write a quorum among the replicas in the local DC. For example if the local DC is assigned 3 replicas, a local quorum is constituted by 2 of those replicas; or if the local DC is assigned 4 replicas, a local quorum is constituted by 3 of those replicas.

Example for a "Replication Across Data Centers" policy

Suppose a "Replication Across Data Centers" policy uses 3X replication in DC-East and 2X replication in DC-West. With this configuration, for each object the system will store 3 object data replicas and 3 object metadata replicas in DC-East, and 2 object data replicas and 2 object metadata replicas in DC-West.

If the policy's consistency level for Read is set to LOCAL QUORUM, and an S3 GET request is received in DC-East (for example), then in order for that request to succeed the system must succeed in reading 2 [object data replicas](#) and 2 object metadata replicas in DC-East.

If the consistency level for Write is set to LOCAL QUORUM, and an S3 PUT request is received in DC-East (for example), then in order for that request to succeed the system must succeed in writing 2 object data replicas and 2 object metadata replicas in DC-East.

Note that what happens in the non-local data center -- DC-West in the examples above -- is irrelevant to meeting the LOCAL QUORUM requirement for a given request. For an S3 request received in DC-East, DC-West could be unreachable or completely offline and LOCAL QUORUM could still be achieved so long as the read or write succeeds for 2 of the 3 replicas assigned to DC-East. Conversely, if the operation does not succeed for 2 of DC-East's 3 assigned replicas, then the S3 request fails regardless of whether or not some replicas can be read or written in DC-West.

Example for a "Replicated EC" policy

Suppose a "Replicated EC" policy uses 4+2 erasure coding replicated in DC-East and DC-West. With this configuration, for each object the system will store 6 object data fragments and 3 object metadata replicas (2m-1, by default) in each of those DCs.

If the policy's consistency level for Read is set to LOCAL QUORUM, and an S3 GET request is received in DC-East (for example), then in order for that request to succeed the system must succeed in reading 4 object data fragments and 2 object metadata replicas in DC-East.

If the consistency level for Write is set to LOCAL QUORUM, and an S3 PUT request is received in DC-East (for example), then in order for that request to succeed the system must succeed in writing 5 object data fragments and 2 object metadata replicas in DC-East.

Note that what happens in the non-local data center -- DC-West in the examples above -- is irrelevant to meeting the LOCAL QUORUM requirement for a given request. For the S3 PUT request received in DC-East, DC-West could be unreachable or completely offline and LOCAL QUORUM could still be achieved so long as the write succeeds for 5 object data fragments and 2 object metadata replicas in DC-East. Conversely, if the system cannot write the 5 object data fragments and 2 object metadata replicas in DC-East, then the S3 PUT

operation fails regardless of whether or not some data fragments and metadata replicas can be written in DC-West.

6.10.8.11. Consistency Level "ONE"

The consistency level "ONE" is supported only in "Replication Within Single Data Center" and "Replication Across Data Centers" data distribution schemes and only for read operations -- not writes.

With a read consistency of ONE, for an S3 GET to succeed the system must succeed in reading one object data replica and one object metadata replica.

Example for a "Replication Within Single Data Center" policy

Suppose a "Replication Within Single Data Center" policy uses 3X replication. With this configuration the system will for each object store 3 object data replicas and 3 object metadata replicas.

If the policy's consistency level for Read is set to ONE, then during an S3 GET operation just one object data replica and just one object metadata replica must be successfully read before the object is returned to the client.

Example for a "Replication Across Data Centers" policy

Suppose a "Replication Across Data Centers" policy uses 3X replication in DC-East and 2X replication in DC-West. With this configuration, for each object the system will store 3 object data replicas and 3 object metadata replicas in DC-East, and 2 object data replicas and 2 object metadata replica in DC-West.

If the policy's consistency level for Read is set to ONE, then during an S3 GET operation just one object data replica (from either DC) and just one object metadata replica (from either DC) must be successfully read before the object is returned to the client.

6.10.8.12. Consistency Level "QUORUM"

The consistency level "QUORUM" is a supported option for every type of data distribution scheme except "Replicated EC", for both reads and writes. The table below shows the general requirements of QUORUM in the context of replication based storage policies and erasure coding based storage policies. Following the table are examples for each supported data distribution scheme.

	Replication	Erasure Coding
Read QUORUM	For an S3 GET to succeed, the system must succeed in reading a majority of object data replicas and a majority of object metadata replicas.	For an S3 GET to succeed, the system must succeed in reading "k" object data fragments and a majority of object metadata replicas.
Write QUORUM	For an S3 PUT to succeed, the system must succeed in writing a majority of object data replicas and a majority of object metadata replicas.	For an S3 PUT to succeed, the system must succeed in writing "k"+1 object data fragments and a majority of object metadata replicas.

Note For replicated data or metadata, a quorum or majority of a set of replicas is defined mathematically as $(\#replicas/2) + 1$, rounded down to the nearest integer. For example with 3 replicas a

quorum would be $(3/2 = 1.5) + 1 = 2.5$, rounded down = 2. With 4 replicas a quorum would be $(4/2 = 2) + 1 = 3$, rounded down = 3.

Example for a "Replication Within Single Data Center" policy

Suppose a "Replication Within Single Data Center" policy uses 3X replication. With this configuration, for each object the system will store 3 object data replicas and 3 object metadata replicas.

If the policy's consistency level for Read is set to QUORUM, then during an S3 GET operation 2 [object data replicas](#) and 2 object metadata replicas must be successfully read before the object is returned to the client.

If the consistency level for Write is set to QUORUM, then during an S3 PUT operation 2 object data replicas and 2 object metadata replicas must be successfully written before a success response is returned to the client.

Example for an "EC Within Single Data Center" policy

Suppose an "EC Within Single Data Center" policy uses 4+2 erasure coding. With this configuration, for each object the system will store 6 object data fragments and 3 object metadata replicas (2m-1, by default).

If the policy's consistency level for Read is set to QUORUM, then during an S3 GET operation 4 object data fragments and 2 object metadata replicas must be successfully read before the object is returned to the client.

If the consistency level for Write is set to QUORUM, then during an S3 PUT operation 5 object data fragments and 2 object metadata replicas must be successfully written before a success response is returned to the client.

Example for a "Replication Across Data Centers" policy

Suppose a "Replication Across Data Centers" policy uses 3X replication in DC-East and 2X replication in DC-West. With this configuration, for each object the system will store 3 object data replicas and 3 object metadata replicas in DC-East, and 2 object data replicas and 2 object metadata replicas in DC-West.

If the policy's consistency level for Read is set to QUORUM, then during an S3 GET operation any 3 [object data replicas](#) and any 3 object metadata replicas must be successfully read before the object is returned to the client. For example the 3 readable object data replicas could be constituted as 1 in DC-East and 2 in DC-West, or 2 in DC-East and 1 in DC-West, or 3 in DC-East and 0 in DC-West.

If the consistency level for Write is set to QUORUM, then during an S3 PUT operation any 3 object data replicas and any 3 object metadata replicas must be successfully written before a success response is returned to the client. For example the 3 successfully written object data replicas could be constituted as 1 in DC-East and 2 in DC-West, or 2 in DC-East and 1 in DC-West, or 3 in DC-East and 0 in DC-West.

Example for an "EC Across Data Centers" policy

Suppose an "EC Across Data Centers" policy uses 7+5 erasure coding distributed across DC-East, DC-West, and DC-South. With this configuration, for each object the system will store 4 object data fragments and 3 object metadata replicas in each of those DCs (for the object metadata it's 9 replicas [2m-1, by default] divided evenly among the three DCs).

If the policy's consistency level for Read is set to QUORUM, then during an S3 GET operation any 7 object data fragments and any 5 object metadata replicas must be successfully read before the object is returned to the client.

If the consistency level for Write is set to QUORUM, then during an S3 PUT operation a total of 8 object data fragments and a total of 5 object metadata replicas must be successfully written before a success response is

returned to the client. The 8 successfully written object data fragments could be constituted as, for example, 2 in DC-East and 3 in DC-West and 3 in DC-South; or 4 in DC-East and 4 in DC-West and 0 in DC-South.

6.10.8.13. Create Storage Policy -- Group Visibility

In the "Group Visibility" section of the [storage policy creation interface](#) specify which user groups will be allowed to use the policy. Users from the groups that you select here will see the policy as an available storage policy when they create a storage bucket. Users are required to choose a storage policy when creating a bucket.

To make this storage policy available to **all** user groups, do nothing in the "Group Visibility" section of the interface. Making policies available to all user groups is the default behavior.

IMPORTANT ! If this storage policy is going to be the [default storage policy](#), do not specify any groups. The default storage policy must be available to all groups. If you do not yet have any storage policies in your system, the first policy that you create must be a default policy that is available to all groups.

If you want the policy to be used only by certain user groups, do the following for each group that you want the policy to be available to:

- a. Use the drop-down list to select a group.
- b. Click **Add**.

The groups that you select will then display in the "Group Visibility" section interface.

6.10.8.14. Create Storage Policy -- Compression

In the "Compression" field of the [storage policy creation interface](#) select the type of compression (if any) to use for objects stored in the HyperStore File System. This applies to replicated object data and erasure coded object data. It does not apply to metadata stored in the Metadata DB.

Supported options are:

- [Snappy](#)
- [Zlib](#)
- [LZ4](#)
- None (no compression)

When enabled, compression is applied to incoming objects at the "coordinator node" (the node that happens to receive the incoming S3 PUT request with that object), before the object data is transmitted to the endpoint nodes on which it will be stored. Any change that you make to the compression type setting — such as changing it from disabled to a particular compression type, or from one compression type to another — is applied only to new objects as they come into the system, not retroactively to objects that are already stored in the system.

If in a storage policy you enable both compression and server-side encryption, the compression is applied to the object data first and then the compressed data is encrypted.

Each object's compression type (if any) is stored in the object's metadata in the Metadata DB. Consequently, if for example you use Snappy for a while and then switch to LZ4, those objects that had been compressed with

Snappy will continue to have in their metadata an indicator that they were compressed with Snappy — and so the system will be able to de-compress the objects when they are downloaded by S3 client applications.

Note

* For S3 service usage tracking (for purposes of QoS enforcement and billing), the **uncompressed** size of objects is always used, even if you enable HyperStore compression.

* For a **PUT Object - Copy** operation, the copy of the object will be subject to whatever the **current** compression type setting is (which may be different than the setting that was in effect when the original object was uploaded).

6.10.8.15. Create Storage Policy -- Server-Side Encryption

IMPORTANT ! Before creating a storage policy that includes server-side encryption, review the **"Server-Side Encryption"** (page 186) and **complete the preparation steps** described in **"Using Regular SSE"** (page 188) or (if applicable to your use case) **"Using SSE-KMIP"** (page 191).

In the "Server-Side Encryption" field of the [storage policy creation interface](#) select the default method of server-side encryption that the system should apply to all objects in all buckets that use this storage policy.

Supported options are:

- **None** -- No server-side encryption
- **SSE** -- Regular server-side encryption using encryption keys managed by the HyperStore system
- **KMIP** -- Server-side encryption using encryption keys managed in part by an external Key Management System (KMS) that supports the Key Management Interoperability Protocol (KMIP). Examples of KMIP compliant KMS vendors include Thales, Fortinix, and Hashicorp. If you select KMIP then an additional field will display, in which you can select among the [KMIP profiles](#) that you have pre-configured. (A KMIP profile specifies the information that HyperStore needs to connect to a particular KMIP-supporting KMS. You must create at least one KMIP profile before you can configure a storage policy to use KMIP based encryption.)

Note that:

- HyperStore supports setting server-side encryption at three different levels: storage policy default; bucket default; and per-object. For a summary of the options at different levels and the interplay between different levels, see **"Server-Side Encryption"** (page 186).
- If you edit the Server-Side Encryption setting for an existing storage policy, your change applies only to objects that are uploaded from that time forward (into buckets that use that storage policy). The change does not apply to objects that are already in storage.

6.10.8.16. Create Storage Policy -- Workload Type

In the "Workload Type" section of the [storage policy creation interface](#) from the drop-down list choose the type of storage workload for which this bucket will be used:

- Leave the workload type at "Default" if either of these use cases applies:
 - The bucket will be used for a purpose other than storage for one of the backup applications listed below.

- The bucket will be used for multiple purposes.
- Choose one of the listed backup applications if the bucket will be used exclusively or primarily as a storage target for that backup application:
 - ACTIFIO
 - COHESITY
 - COMMVAULT
 - NETBACKUP
 - RUBRIK
 - VEEAM (Veeam 11 or earlier)
 - VEEAM12 (Veeam 12 or newer)
 - VEEAMO365 (Veeam for Microsoft Office 365)
 - NASUNI

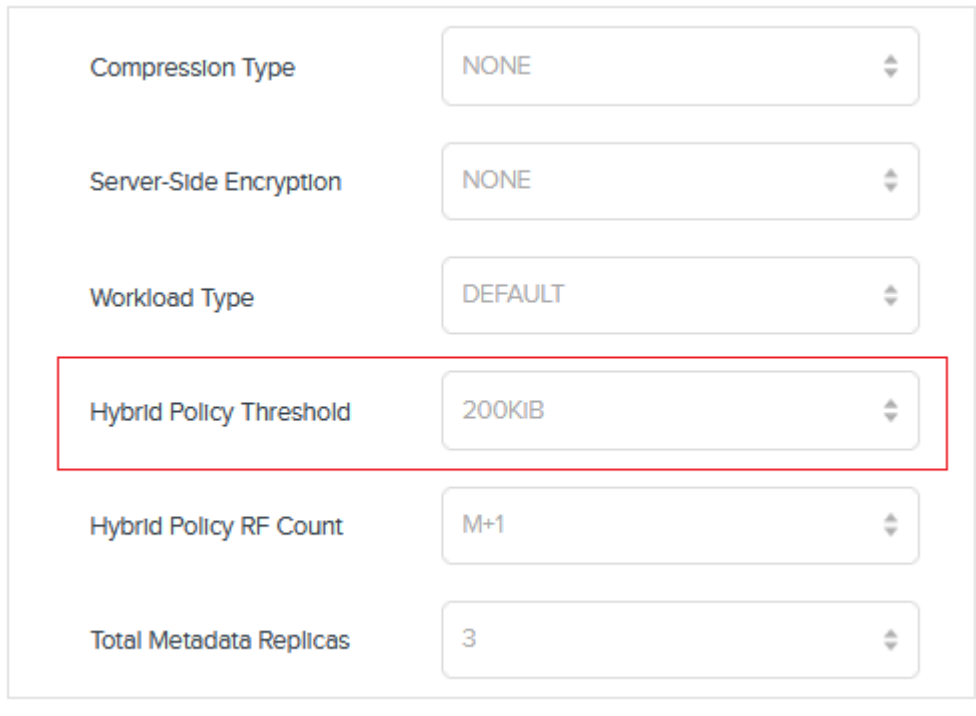
Based on your choice of workload type, the metadata for the bucket will be stored in the Metadata DB in a manner that optimizes storage efficiency and bucket listing performance (performance for the S3 *ListObjects* operation). For buckets used for one of the above-listed backup application workloads, the metadata storage scheme will be optimized based on how that application is known to write its data -- in particular, the directory structure that the application is known to employ when writing backup data to a storage target. For a bucket that uses the "Default" workload type, the system automatically recognizes if one of the backup applications listed above is writing a portion of the data in the bucket, and for that portion of the bucket data the metadata storage scheme is tailored to that application's data-writing pattern. For all other data in a "Default" workload type bucket, a metadata data storage scheme suitable to general purpose workloads is used.

Note that:

- A storage policy's Workload Type setting cannot be changed after the storage policy is created.
- If you have a workload other than the backup applications listed above, and you are experiencing poor performance for the S3 *ListObjects* operation, contact Cloudian Support. Cloudian Support can work with you to create a custom workload type that will optimize bucket metadata storage for your particular workload.
- For new buckets created in HyperStore version 7.5.1 and later, all Workload Types are capable of detecting if some or all of a bucket's content is associated with a different one of the Workload Types, and if so the metadata storage scheme will automatically be adapted to that detected workload. In buckets created prior to HyperStore 7.5.1, only buckets using the "DEFAULT" Workload Type had the ability to dynamically adapt to detected bucket content in this way.
- If you have upgraded to HyperStore 7.5.1 from an earlier version, and in your existing HyperStore system you have buckets that use a storage policy that's configured for the "VEEAM" Workload Type (for VEEAM11), those existing VEEAM buckets will not automatically adjust to the VEEAM12 data structure if you upgrade your VEEAM application to VEEAM12. If you have existing VEEAM buckets that you want to convert to VEEAM12, contact Cloudian Support for assistance. Note further that:
 - If you have existing buckets that use a storage policy that's configured for the "DEFAULT" Workload Type, those bucket will automatically detect and adjust to a VEEAM12 workload, if you use those buckets as targets for a VEEAM12 application.
 - From HyperStore 7.5.1 and forward, if you create **new** buckets that use storage policy that's configured for the "VEEAM" Workload Type (for VEEAM11), and then you later upgrade your VEEAM application to VEEAM12, those buckets will automatically adapt to the VEEAM12 workload.

IMPORTANT ! Once you finish creating a storage policy, you cannot change the storage policy's Workload Type.

6.10.8.17. Create Storage Policy -- Hybrid Policy Threshold



The screenshot shows a configuration form for creating a storage policy. It includes several settings, each with a label and a dropdown menu:

- Compression Type:** NONE
- Server-Side Encryption:** NONE
- Workload Type:** DEFAULT
- Hybrid Policy Threshold:** 200KiB (This row is highlighted with a red rectangular border)
- Hybrid Policy RF Count:** M+1
- Total Metadata Replicas:** 3

The "Hybrid Policy Threshold" setting displays in the [storage policy creation interface](#) only if you've selected EC Within Single Data Center, Replicated EC, or EC Across Data Centers as the [data distribution scheme](#) for the storage policy that you are creating.

For large objects erasure coding is a more storage space-efficient means for ensuring data availability and durability than replication. But for smaller objects, when all things are considered -- capacity consumption for object data and metadata as well as object read/write performance -- replication is the superior data protection method. In an EC storage policy, the "Hybrid Policy Threshold" setting defines an object size threshold such that:

- Objects larger than this size threshold will be protected by erasure coding (using whichever EC scheme you selected for the storage policy -- EC Within Single Data Center, Replicated EC, or EC Across Data Centers)
- Objects at or smaller than this size threshold will be protected by replication rather than erasure coding.

For example, with the default threshold of 200KiB, a 225KiB or 10MiB or 1GiB object uploaded to a bucket that uses this storage policy will be erasure coded and a 150KiB or 200KiB object uploaded to the bucket will be replicated (not erasure coded).

The 200KiB default value is based on testing to determine the approximate object size threshold above which erasure coding tends to deliver a better overall combination of resource utilization efficiency and performance, and below which replication is better. But the "Hybrid Policy Threshold" drop-down menu offers you the option of setting the threshold at 8KiB, 32KiB, 128KiB, 200KiB (the default), 512KiB, 1MiB, or 2MiB. The drop-down

menu also offers a "Disabled" option: if you set the Hybrid Policy Threshold to "Disabled" then this storage policy will apply erasure coding to all objects no matter how small.

In most environments and with most types of workloads, it's beneficial to use the Hybrid Policy Threshold feature. If you are considering disabling the threshold for a storage policy, you should consult first with Cloudian Support or your Cloudian sales representative.

The 200KiB default threshold is appropriate for most use cases. If you are considering using a different threshold than the 200KiB default, Cloudian Support or your Cloudian sales representative can provide advice based on your environment and use case.

IMPORTANT ! Once you finish creating a storage policy, you cannot change the storage policy's Hybrid Policy Threshold.

Note

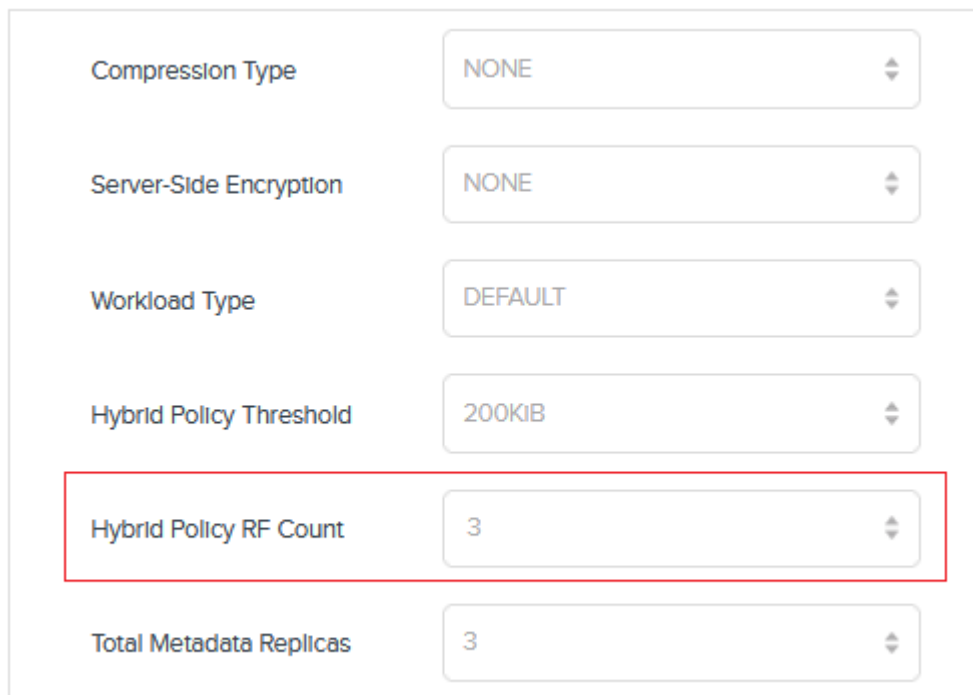
* The Hybrid Policy feature is supported only for new storage policies created in HyperStore 7.5.1 or later.

* The option to disable the Hybrid Policy Threshold feature may be removed in a future HyperStore release.

Replication Factor Used for Small Objects

As you're creating a new storage policy you can configure the replication factor for objects at or smaller than your Hybrid Policy Threshold size. To configure the replication factor use the [Hybrid Policy RF Count](#) setting in the storage policy creation interface.

6.10.8.18. Create Storage Policy -- Hybrid Policy RF Count



Compression Type	NONE
Server-Side Encryption	NONE
Workload Type	DEFAULT
Hybrid Policy Threshold	200KIB
Hybrid Policy RF Count	3
Total Metadata Replicas	3

The "Hybrid Policy RF Count" setting displays in the [storage policy creation interface](#) only if you've selected an erasure coding [data distribution scheme](#) (EC Within Single Data Center, Replicated EC, or EC Across Data Centers) for the storage policy that you are creating.

If you have a [Hybrid Policy Threshold](#) for an erasure coding storage policy that you're creating, objects at or smaller than the Hybrid Policy Threshold size will be protected by replication rather than erasure coding, and the replication factor will be as you configure with the Hybrid Policy RF Count setting. The RF Count options depend on what the "m" value is in the erasure coding data distribution scheme that you are using:

"m" Value	Supported RF Count Options (to be applied to small objects)	Default
2	3, 4, or 5	3
3	3, 4, 5, 6, or 7	4
4	3, 4, 5, 6, 7, 8, or 9	5

In each case the default value is equal to $m+1$. For example for a 4+2 erasure coding policy, the RF Count default is 3.

The table below shows how the RF Count options are applied to small objects (objects at or below the threshold size) in different types of EC data distribution schemes, using the examples of RF Count of 3, 4, and 5.

RF Count Option	EC Within Single Data Center	Replicated EC	EC Across Data Centers
3	3 replicas within the one DC	3 replicas within each DC in the storage policy	A total of 3 replicas distributed among the DCs in the storage policy
4	4 replicas within the one DC	4 replicas within each DC in the storage policy	A total of 4 replicas distributed among the DCs in the storage policy
5	5 replicas within the one DC	5 replicas within each DC in the storage policy	A total of 5 replicas distributed among the DCs in the storage policy

In all cases, the write consistency level and read consistency level applied to replicated, smaller-than-threshold objects is [QUORUM](#). These consistency levels are not configurable.

When configuring the Hybrid Policy RF Count setting, be aware that **choosing a higher RF count prioritizes read availability** (since a higher number of replicas allows for more replicas to be unreachable while still achieving quorum) and **choosing a lower RF count prioritizes performance** (fewer replicas to write and read) **and storage capacity savings** (fewer replicas to store). If you don't have a clear preference based on your use case, Cloudian recommends using the default setting ($m+1$).

IMPORTANT ! Once you finish creating a storage policy, you cannot change the storage policy's Hybrid Policy RF Count. You can only configure a Hybrid Policy RF Count as you are creating a new storage policy.

6.10.8.19. Create Storage Policy -- Total Metadata Replicas

The screenshot shows a configuration interface for creating a storage policy. It includes several settings, each with a label and a dropdown menu:

- Compression Type:** NONE
- Server-Side Encryption:** NONE
- Workload Type:** DEFAULT
- Hybrid Policy Threshold:** 200KIB
- Hybrid Policy RF Count:** M+1
- Total Metadata Replicas:** 3 (This setting is highlighted with a red rectangular box)

The "Total Metadata Replicas" setting displays in the [storage policy creation interface](#) only if you've selected an erasure coding [data distribution scheme](#) (EC Within Single Data Center, Replicated EC, or EC Across Data Centers) for the storage policy that you are creating.

Even in storage policies that erasure code object data, object metadata is protected by replication rather than erasure coding. This is because an object's metadata is very small in size, and erasure coding is inappropriate for very small amounts of data.

With the "Total Metadata Replicas" setting you can configure the object metadata replication factor for the erasure coding storage policy that you are creating. Your options are limited to those in the drop-down list, and the list will be determined by the data distribution scheme and $k+m$ values that you've selected as you've been configuring the new storage policy.

	EC Within Single Data Center	Replicated EC	EC Across Data Centers
Maximum	$2m+1$	$2m+1$ per DC	$2m-1$ total
Default	$2m-1$	$2m-1$ per DC	$2m-1$ total
Minimum	3	3 per DC	3 total
Other supported values	Odd numbers between $2m-1$ and minimum, if	Odd numbers between $2m-1$ and minimum, if any	Odd numbers between $2m-1$ and minimum, if any

	EC Within Single Data Center	Replicated EC	EC Across Data Centers
	any		
Examples	4+2 EC within single DC -> options are 5, 3 (default)	4+2 Replicated EC in 2 DCs -> 10, 6 (default) (these options reflect 5 or 3 replicas in each DC, respectively)	5+4 EC across 3 DCs -> options are 7 (default), 5, 3
	9+3 EC within single DC -> options are 7, 5 (default), 3	12+4 Replicated EC in 3 DCs -> 27, 21 (default), 15, 9 (these options reflect 9, 7, 5, or 3 replicas in each DC, respectively)	7+5 EC across 3 DCs -> options are 9 (default), 7, 5, 3

Note For "EC Across Data Centers" policies the metadata replicas will be distributed evenly among the DCs if mathematically possible (for example for 9 total replicas and 3 DCs it would be 3 replicas in each DC). Otherwise the replicas will be distributed as close to evenly as possible (for example for 5 total replicas and 3 DCs it would be 2 replicas in each of two DCs and 1 replica in the third DC).

When configuring the Total Metadata Replicas setting, be aware that **choosing a higher replica count prioritizes read availability** (since a higher number of replicas allows for more replicas to be unreachable while still achieving your [configured consistency level](#)) and **choosing a lower replica count prioritizes performance** (fewer replicas to write and read) **and storage capacity savings** (fewer replicas to store).

For example in a 4+2 EC storage policy within a single DC, and a QUORUM consistency requirement for reads, with $2m+1$ as the metadata replication factor there will be 5 metadata replicas per object and an object's metadata can still be read if 2 replicas are unavailable. With $2m-1$ as the metadata replication factor there will be 3 metadata replicas and an object's metadata can be read if 1 replica is unavailable but not if 2 replicas are unavailable. On the other hand, write and read performance would be marginally better with the lower number of replicas, and -- when applied across a potentially very large number of objects stored under the storage policy -- the Cassandra storage load would be less with the lower number of metadata replicas per object.

If you don't have a clear preference based on your use case, Clouidian recommends using the default setting ($2m-1$).

IMPORTANT ! Once you finish creating a storage policy, you cannot change the storage policy's Total Metadata Replicas value.

Note that:

- In the case of [Hybrid erasure coding](#) policies your configured Total Metadata Replicas value applies to the metadata of all objects regardless of size.
- The Total Metadata Replicas is configurable only for new Hybrid storage policies created in HyperStore 7.5.3 or later. For existing Hybrid storage policies created in earlier HyperStore versions the metadata replication factor for erasure coding policies was $2m-1$ (unless your Clouidian representative used a hidden setting to change your default to $2m+1$).

6.10.8.20. Edit a Storage Policy

To edit an existing storage policy, in the policy list on the CMC's [Storage Policies](#) page click **View/Edit** for the policy that you want to edit. You can then edit policy attributes such as metadata search support, data

consistency requirements, group visibility, compression, or server-side encryption. For guidance on working with these policy characteristics, see ["Add a Storage Policy" \(page 513\)](#).

You cannot change a storage policy's Data Distribution Scheme or Workload Type, and -- specific to erasure coding policies -- you cannot change a policy's Hybrid Policy Threshold, Hybrid Policy RF Count, or Total Metadata Replicas. By clicking **View/Edit** for a policy you can see how these attributes are currently configured for a policy, but you cannot change these attributes.

Any changes you make to how stored objects are handled, such as compression or server-side encryption, will apply only to objects uploaded **after** you make the storage policy edit -- not to objects that are already in storage. For example if you enable server-side encryption on an existing storage policy, then from that time forward objects that get uploaded to buckets that use that storage policy will be encrypted -- but objects that were already uploaded prior to the configuration change will not be encrypted. Conversely, if an existing storage policy has been configured for encryption and then later you disable encryption for that policy, then from that time forward newly uploaded objects will not be encrypted -- but objects that had already been uploaded (and encrypted) prior to the configuration change will remain encrypted.

IMPORTANT ! Erasure coding storage policies created with HyperStore versions 5.x or 6.x included a functionality that has been deprecated. The functionality, which identified more than $k+m$ endpoints as possible write destinations for an erasure coded object's $k+m$ fragments, ultimately was found to pose potential problems that outweighed the benefits (and so erasure coding storage policies created in HyperStore 7.0.1 and later returned to using the more conventional approach of identifying only $k+m$ endpoints for erasure coded writes). Support for legacy storage policies with the deprecated functionality will end in a future HyperStore release (the exact timing is still to be determined). Starting with HyperStore 7.5.2, the system automatically detects if you have any of these deprecated storage policies in your system. If you do have any of these deprecated storage policies, then:

- * A warning will display in the CMC **Dashboard**. The warning will indicate that you have one or more deprecated storage policies in your system and that you will need to contact your Cloudbian Sales Representative to plan for migrating data from those deprecated storage policies to supported storage policies.
- * In the CMC's **Storage Policies** page -- which lists all your storage policies -- a warning symbol will display next to the names of the deprecated storage policy or policies. Hover text shows the warning description.
- * Users creating new buckets will not be able to use a deprecated storage policy (such policies will not display in the list of available storage policies that users see when creating a new bucket).

6.10.8.21. Designate a Default Storage Policy

At all times you must have one and only one **default storage policy** defined in each of your HyperStore service regions. The default policy is the one that will be applied when users create new buckets without specifying a policy.

In your storage policy list (in the CMC's [Storage Policies](#) page), the current default policy is listed first, with its Region name highlighted in green. In the example below, the policy named "HSFS-1" is the default storage policy.

STORAGE POLICIES								+ CREATE STORAGE POLICY
☐	REGION	STATUS	NAME	DESCRIPTION	DATA DISTRIBUTION POLICY	NO OF REPLICAS	LOCAL EC	
☐	Us-norcal	ACTIVE	HSFS-1	HSFS - DC1:2 - DC2:1	Multi DC	3	N/A	View/Edit
☐	Us-norcal	ACTIVE	EC-1	EC: K2-M1	Multi DC	2	2 + 1	Set As Default View/Edit
☐	Us-norcal	ACTIVE	Test_Policy		Single DC	3	N/A	Set As Default View/Edit
☐	Us-norcal	ACTIVE	Test_Policy_2		Single DC	1	2 + 1	Set As Default View/Edit
								DISABLE DELETE

To be eligible for being the default storage policy, a policy must be:

- Active (Status = ACTIVE)
- Visible to all user groups. If you're unsure whether a particular policy is visible to all groups, click **View/Edit** for the policy, then check in the **Group Visibility** panel. If individual group names appear in this panel, that means the policy is currently configured to be visible to only those groups. You can change this by deleting all the specific groups that are displayed and saving the edited storage policy.

The first storage policy that you create will automatically become the default storage policy. To subsequently designate a different policy as the default, click **Set as Default** for that policy. The interface will ask you to confirm that you want to take this action.

Note that **changing the default storage policy has no effect on which policy is used by currently existing buckets**. Buckets that have been using the old default storage policy will continue to do so. The impact is that when new buckets are created without specifying a storage policy, they will be assigned the new default storage policy.

6.10.8.22. Disable a Storage Policy

Disabling a storage policy prevents users from choosing the policy when they create new buckets. However, buckets that are already using the policy will continue to do so even after the policy is disabled.

Note You cannot disable the default storage policy. If you want to disable that policy you must first promote a different policy to be the new default.

Note Disabled policies still count toward the configurable limit on the number of policies that can exist in the system.

To disable a policy, in the CMC's [Storage Policies](#) page select the checkbox to the left of the policy name and then click **Disable**. The interface will ask you to confirm that you want to take this action.

If subsequently you want to re-enable a disabled storage policy, select the policy and click **Enable**. When you re-enable a storage policy, the policy once again becomes available for service users to assign to newly created buckets.

6.10.8.23. Delete a Storage Policy

Deleting a storage policy completely removes the policy from the system. There are restrictions on which policies you can delete. You cannot delete:

- The default storage policy. If you want to delete that policy you must first [promote a different policy to be the new default](#).
- The system-generated policy named "DEFAULT_<regionName>" (applicable only to HyperStore systems that have upgraded from versions older than 5.2).
- Storage policies that are currently being used by one or more buckets. If a policy is being used by buckets and you want those buckets to continue using it but you no longer want the policy to be available to new buckets, you can disable the policy rather than deleting it. See ["Disable a Storage Policy"](#) (page 544). If you really want to delete such a policy, see the required preliminary steps below.

To delete a storage policy that has never been used (that is, the policy has never been associated with any buckets), in the CMC's [Storage Policies](#) page select the checkbox to the left of the policy name and then click **Delete**. The interface will ask you to confirm that you want to take this action, and when you confirm the policy will be deleted.

To delete a policy that has been or is currently being used by one or more buckets, do the following:

1. If any buckets are currently using the policy, first delete all objects from those buckets, then delete the buckets themselves.

Note It is not possible to reassign a different storage policy to a bucket. If you want to delete a storage policy that is currently being used by one or more buckets, you must delete those buckets.

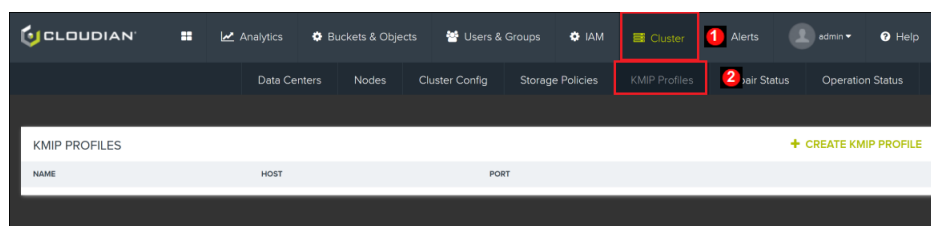
If you are not certain which buckets are currently using a given storage policy, you can use the Admin API method [GET /bppolicy/bucketsperpolicy](#) to retrieve this information.

2. Delete the policy: In the [Storage Policies](#) page select the checkbox to the left of the policy name and then click **Delete**. The interface will ask you to confirm that you want to take this action, and when you confirm the policy will be deleted.

Note As a best practice, after deleting a storage policy wait 24 hours and then run [hssstool cleanup](#) on each node in your cluster, using the `-a` option and the `-policy` option (`hssstool cleanup -h <host> -a -policy`). This will clean up any garbage data associated with the deleted storage policy -- such as replicas or erasure coded fragments that the system failed to remove when you manually deleted all the objects from buckets that had been using the storage policy (Step 1 above). Normally you should only run `hssstool cleanup` on one node at a time, per data center. If you have circumstances where you need to clean up multiple nodes within a data center, you can try running it on two or three nodes concurrently, but pay close attention to cluster performance and also contact Cloudian Support for further guidance.

6.10.9. KMIP Profiles

Path: **Cluster** → **KMIP Profiles**



Supported tasks:

- ["Add a KMIP Profile"](#) (page 546)
- ["Edit a KMIP Profile"](#) (page 549)
- ["Delete a KMIP Profile"](#) (page 549)

6.10.9.1. Add a KMIP Profile

This section describes how to create a Key Management Interoperability Protocol (KMIP) profile. A KMIP profile identifies a specific Key Management System (KMS) that supports KMIP for purposes of server-side encryption of stored object data ("data at rest"), and encapsulates the information needed for HyperStore to connect to that KMS. If you wish you can create multiple KMIP profiles, each for a different KMS. For general information about server-side encryption with KMIP, see ["Using SSE-KMIP"](#) (page 191).

After you create a profile it will be available for use in two ways:

- When you [create a storage policy](#) or [edit a storage policy](#), you can optionally enable KMIP based encryption as part of the storage policy. When doing so you will choose from among the KMIP profiles that you have created. Subsequently, whenever users create buckets that use that storage policy, KMIP encryption will automatically be applied to all objects that get uploaded to the bucket (using the KMS identified by the KMIP profile that you selected when configuring the storage policy).
- If you create some storage policies for which you do not enable KMIP based encryption, then when a user creates a bucket that uses one of those storage policies they will have the option to enable KMIP encryption specifically for that bucket. If they do so, they will choose from among the KMIP profiles that you have created (as well as any KMIP profiles created by the user or their group administrator, if you've [configured the system to allow users and/or group admins to create KMIP profiles](#)).

During this procedure for creating a KMIP profile you will need to upload a client keystore file that HyperStore will use for TLS-authenticated access to the KMS that you identify in the KMIP profile. A client keystore is a single file containing a private key and a certificate (or certificate chain). Typically the private key and certificate(s) will have been provided to you by the KMS. If your TLS client private key and certificate(s) are not yet in the form of a keystore file, you must convert them to a keystore file before you can create a KMIP profile. The following client keystore file formats are supported:

- PKCS12 (.p12 or .pfx extension)
- JKS (.jks extension)
- BCFKS (.bcfks extension) -- Use this format if (and only if) you've set ["common.fips.enabled"](#) (page 743) to *true* in your HyperStore system (by default it's *false*). If you have configured HyperStore to allow group admins and/or regular users to create KMIP profiles, then you should communicate to them that they too should use this keystore file format.

For tools and instructions for creating a keystore file, check any reputable online resource.

Once you've converted your private key and certificate(s) into a keystore file, put the keystore file in a working directory on your local device (the device on which you're running your browser) so that you can upload it during the procedure below.

Note If the KMS uses a **private** Certificate Authority (CA) as its root CA for TLS, you will also need to upload a copy of the KMS's TLS server certificate file from your local device. The supported server certificate formats are JKS and BCFKS. As with the keystore format, use the BCFKS format for the server certificate file if and only if you've set ["common.fips.enabled"](#) (page 743) to *true* in your HyperStore

system (and communicate this as a requirement to group admins and/or regular users if you've configured the system to allow them to create KMIP profiles). Uploading the KMS's server certificate is not necessary if the KMS uses a public CA as its root CA.

Note If you had been using KMIP while running HyperStore versions older than 8.2 -- back when only one KMS was allowed -- that existing KMS connection information is automatically encapsulated into a KMIP "profile" by the HyperStore 8.2 upgrade process, and is given the KMIP profile name "default". If this applies to you, you should see the "default" profile listed in the **KMIP Profiles** page.

1. In the **KMIP Profiles** page, click **Create KMIP Profile**. This opens the **Create New KMIP Profile** panel.

2. In the **Create New KMIP Profile** panel, complete the KMIP profile information:

Profile Name (mandatory)

Your choice of a name for the profile. When you finish creating the profile, this name is how the profile will be identified in CMC interfaces (including the storage policy configuration interface and the bucket properties interface). The name must be unique within your whole HyperStore system (including names of any KMIP profiles that have been created by group administrators or regular users).

The vertical bar character "|" is not allowed in profile names.

Host (mandatory)

IP address or the DNS-resolvable FQDN or hostname of the KMS. Do not include the transfer protocol prefix (i.e. do not include `https://`).

Port (mandatory)

KMIP listening port used by the KMS. The default is 5696 which is the port typically used for KMIP.

User Name and Password (if required by KMS)

The user name and password for accessing your KMS account, if the KMS requires these for KMIP

connections.

Client Keystore (required)

In this field, upload a TL client keystore file from your local device. For more information about this file see the [introduction to this "Add a KMIP Profile" procedure](#).

Server Certificate (required if KMS uses a private CA as its root CA)

In this field, upload a copy of the KMS's TLS server certificate file from your local device. This is required only if the KMS uses a **private** Certificate Authority (CA) as its root CA, rather than using a public CA as its root CA. HyperStore's TLS truststore includes all the major public root CAs by default.

TLS Protocols (if required by KMS)

The TLS protocol(s) supported by the KMS, for incoming TLS connections. Default is TLSv1.2 and TLSv1.3.

This setting is mandatory if required by the particular KMS that you are using; otherwise this setting is optional. Note that this setting is required for connecting to a HashiCorp Vault KMS or a PyKMIP KMS.

TLS Cipher Suites (if required by KMS)

The TLS cipher suite(s) supported by the KMS, for incoming TLS connections. To add a cipher suite to the list, type the suite name in to the "TLS Cipher Suites" text input box and then click **Add TLS Cipher Suite**. If there are multiple supported suites for the KMS, add just one suite to the list at a time (typing in the suite name and clicking **Add TLS Cipher Suite**) until all suites have been added. Be careful with your typing since a mistake in a cipher suite name will cause the KMIP profile creation to fail (when you click **Save** in Step 3 below).

This setting is mandatory if required by the particular KMS that you are using; otherwise this setting is optional. Note that this setting is required for connecting to a HashiCorp Vault KMS or a PyKMIP KMS.

3. Click **Save**.

When you click Save the CMC will verify the information you've provided by initiating a test connection to the KMS. Upon a successful test connection, the KMIP profile will be saved in the system and it will appear in the profile list in the **KMIP Profiles** page.

KMIP PROFILES			+ CREATE KMIP PROFILE
NAME	HOST	PORT	
test_admin1	10.20.2.119	5696	View/Edit Delete

Note If the test connection fails the CMC will display an error message, and you can check the profile configuration again to see if you've entered any inaccurate information.

Note that in the KMIP profiles list you will see only the profiles created by system administrators. KMIP profiles created by group administrators or regular users -- if any -- will not be visible to system administrators. (Whether or not group admins and regular users are allowed to create KMIP profiles is configured by the **"s3.k-mip.manageProfile.grantees"** (page 822) setting. By default they are not allowed to do so.)

Note also that HyperStore does not impose a limit on the number of KMIP profiles created.

6.10.9.2. Edit a KMIP Profile

IMPORTANT ! If the connection details change for a KMS that you're using, edit the KMIP profile for the KMS. Do not create a new KMIP profile for the KMS, since buckets cannot switch from one KMIP profile to a different KMIP profile.

To edit an existing KMIP profile, in the **KMIP Profiles** list click **View/Edit** for the profile that you want to edit. This opens the **Edit KMIP Profile** panel, which shows all the profile attributes that you set for the profile when you created it. Edit the profile attributes as desired and then click **Save**.

When you click **Save** the CMC will verify the updated information you've provided by initiating a test connection to the KMS. Upon a successful test connection, the updated KMIP profile will be saved in the system. If a connection cannot be made with the new information you've provided, an error message will display and your profile changes will not be saved.

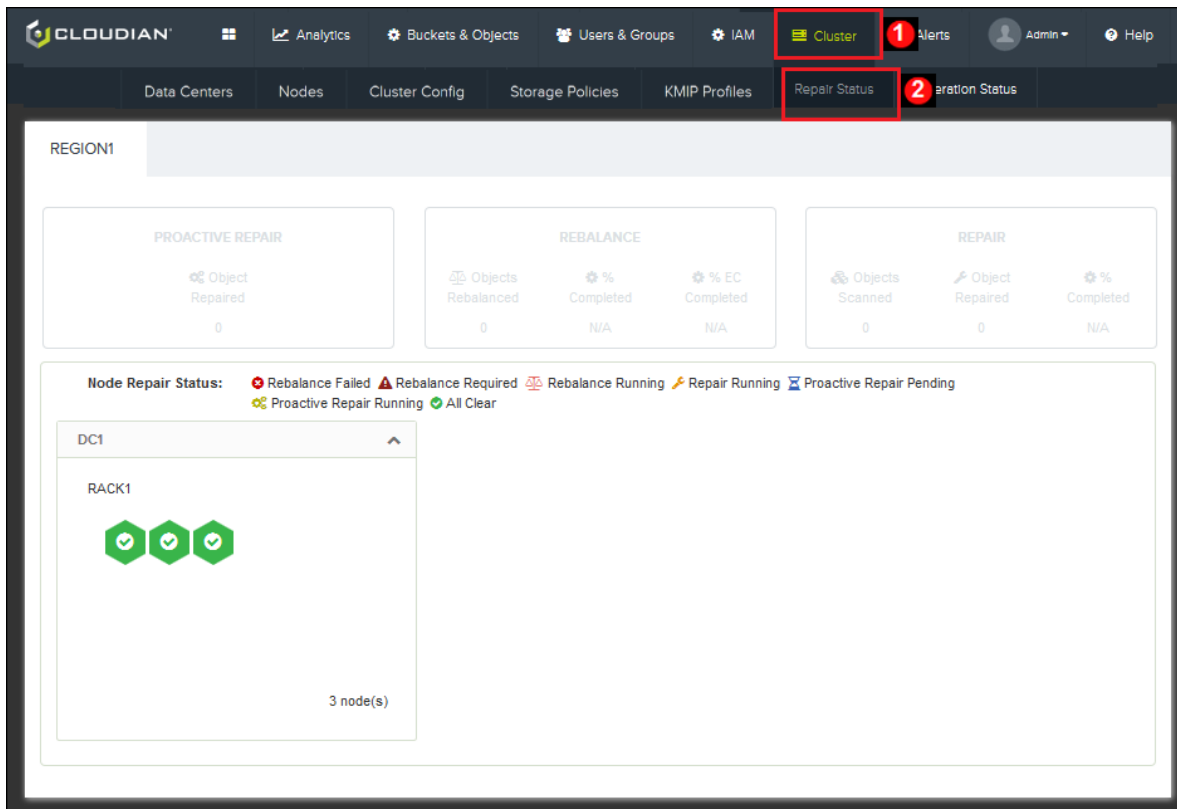
6.10.9.3. Delete a KMIP Profile

To delete a KMIP profile, in the **KMIP Profiles** list click **Delete** for the profile that you want to delete. In the confirmation box that pops up, confirm that you want to delete the profile.

Note You cannot delete a KMIP profile that is being used by any storage policy configuration or by any user's bucket properties configuration.

6.10.10. Repair Status

Path: **Cluster** → **Repair Status**



Supported tasks:

- View repair status for cluster
- View repair status for a node

6.10.10.1. View Repair Status for Cluster

The upper section of the **Repair Status** page shows aggregate data repair status for your cluster as a whole. The information displays in three panels:

Proactive Repair

This panel is activated if [proactive repair](#) is either pending or in progress on any node in your cluster.

For the cluster as a whole, this panel shows:

- Objects Repaired -- The number of objects that have been repaired by in-progress proactive repair operations

Note that an "object" in this context can be either an object replica or an erasure coded object fragment. So if for example the in-progress proactive repair has so far repaired 12 object replicas and 8 erasure coded object fragments, the Objects Repaired value would be 20.

Rebalance

This panel is activated if any of the following operations is in progress on any node:

- A *rebalance* operation (which you should be initiating toward the end of the procedure for ["Adding Nodes"](#) (page 579))

- A *decommission* operation (which the system would automatically initiate as part of the process of ["Removing a Node"](#) (page 603), if the node was accessible within the Cassandra ring)

These operations have in common that object replicas (or erasure coded fragments) are being re-located within the cluster in response to the cluster having been re-sized.

For the cluster as a whole, this panel shows:

- Objects Rebalanced -- How many object replicas or erasure coded fragments have been relocated so far, by the currently in-progress *rebalance* or *decommission* operation.
- % Completed -- For object replicas, the token ranges for which rebalancing or decommissioning has been completed as a percentage of the total token ranges impacted by the cluster resizing.
- % EC Completed -- For object erasure coded fragments, the token ranges for which rebalancing or decommissioning has been completed as a percentage of the total token ranges impacted by the cluster resizing.

Repair

This panel is activated if an [hsstool repair](#) or [hsstool repairc](#) operation is in progress on any node in your cluster. It indicates the number of objects that have been scanned and (from among the scanned objects) the number of objects that needed and received repair. It also shows the completion percentage for the operation, on the node on which it is being run.

For the cluster as a whole, this panel shows:

- Objects Scanned -- The number of objects that have been scanned by in-progress *repair* and/or *repairc* operations to determine whether or not the objects are in need of repair.
- Objects Repaired -- The number of objects that have been repaired by in-progress *repair* and/or *repairc* operations.
- % Completed -- For replica repair only, the completion percentage so far. This is calculated as the number of token ranges for which repair has been completed divided by the total number of token ranges that will be repaired by this *repair* operation. The % Completed metric does not apply to *repairc* operations -- if only a *repairc* operation is running (and not a *repair* operation), then the % Completed field displays "N/A" for not applicable.

6.10.10.2. View Repair Status for a Node

The lower section of the CMC's **Repair Status** page displays a color-coded node icon (cube) for each node in the cluster. The icon indicates the node's current repair status:



Rebalance Failed — The node has been added to the cluster and *rebalance* was run on the node, but the *rebalance* operation failed. Try running *rebalance* on the node again. See ["Adding Nodes"](#) (page 579).



Rebalance Required — The node has been added to the cluster, but *rebalance* has not yet been run on the node. The *rebalance* operation is required in order to shift some data to the node from other nodes in the cluster. See ["Adding Nodes"](#) (page 579).



Rebalance Running — Either a *rebalance* operation is in progress for the node (if the node has been recently [added to the cluster](#)); or else a *decommission* operation is in progress for the node (if the node is live and is being [removed from the cluster](#)).



Repair Running — An [hsstool repair](#) or [hsstool repairec](#) operation is running on the node. This may be as a result of the [scheduled auto-repair](#) feature, or because a system administrator initiated the operation.



Proactive Repair Pending — The system has detected that the node is in need of [proactive repair](#). The repair will occur at the next proactive repair interval (by default every 60 minutes).



Proactive Repair Running — [Proactive repair](#) is in progress on the node.



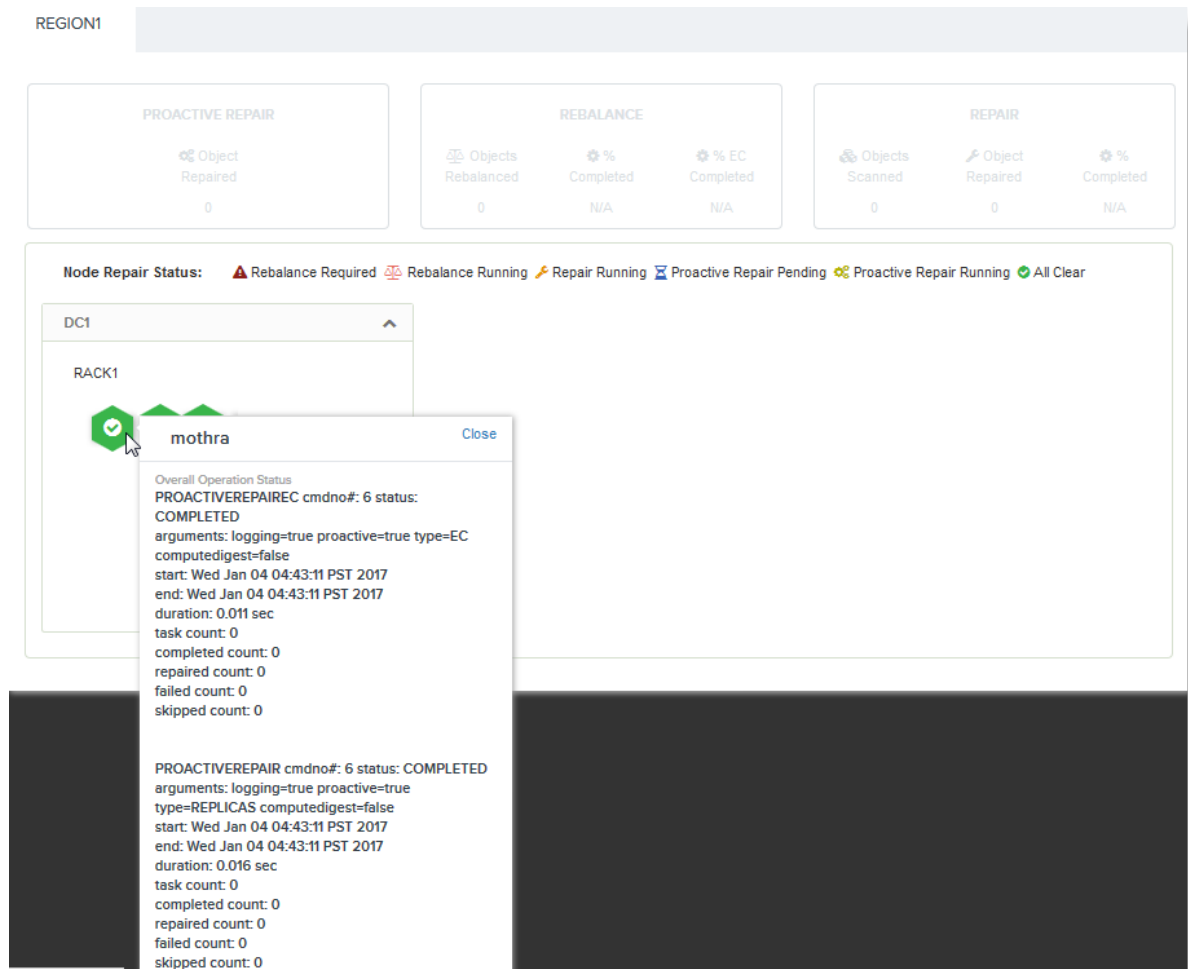
All Clear — No repair of any type is in progress on the node, and the node is not currently in need of rebalance or proactive repair.



Unavailable —The repair information for the node is unavailable, such as if the node is down or inaccessible or the HyperStore Service is down on the node.

Note In the event that multiple repair types are simultaneously running on the node, the color-coding reflects whichever in-progress repair type started first on the node.

If you click a node icon, detailed information about in-progress and recently completed repairs on that node will display.



This is the same information that is retrieved when you run the `hssstool opstatus` command on a node. For description of the available information items see the command response section of "[hssstool opstatus](#)" (page 882).

In the case of proactive repair, the information items will also include the **proactive repair queue length**. This indicates the number of objects for which data -- either a replica or an erasure coded fragment -- currently needs to be written to the node by the proactive repair feature. As the proactive repair of the node proceeds and fewer objects remain to repair, the proactive repair queue shrinks. (Conversely, at times when the node is down, unreachable, or otherwise unable to support writes, the node's proactive repair queue grows.)

If after clicking a node icon, you click the host name at the top of a node's status detail display you will jump to the [Node Status](#) page for that node.

Note If proactive repair is pending on a node, and if for some reason you want to trigger the proactive repair on that node immediately rather than waiting for the automatic hourly run, you can do so by using the [hssstool proactiverepair](#) command with the "-start" option.

Note For information about stopping an in-progress repair see "[Disabling or Stopping Data Repairs](#)" (page 134).

6.10.11. Operation Status

Path: **Cluster** → **Operation Status**

OPERATION LIST

Show entries

Search:

OPERATION NAME	TARGET	STATUS	PROGRESS	START TIME	LAST UPDATE	
addNode	region1:dc1:failhost	❌ failed	failed	Feb-23-2018 07:27	Feb-23-2018 07:28	View Delete
cleanupec	region1:dc1:store1	✅ completed	100%	Feb-23-2018 07:25	Feb-23-2018 07:25	View Delete
repair	region1:dc1:store1	✅ completed	100%	Feb-23-2018 07:25	Feb-23-2018 07:25	View Delete
cleanup	region1:dc1:store1	✅ completed	100%	Feb-23-2018 07:24	Feb-23-2018 07:24	View Delete

Showing 1 to 4 of 4 entries

[Previous](#) [Next](#)

Supported task:

- Check status of operations launched from the CMC

In the **Operation Status** page you can view the status of long-running operations that you have launched from the CMC. Status reporting in this page is supported for these operation types:

- Add Node
- Add Data Center
- Add Region
- Uninstall Node
- hssstool rebalance
- hssstool repair or repairec
- hssstool cleanup or cleanupec

Note For *hssstool* operations, the CMC's **Operation Status** page reports only on *hssstool* operations that you initiate through the CMC's [Node Advanced](#) page. It does not report on *hssstool* operations that you initiate manually through the command line. For viewing status of operations that you initiate on the command line use [hssstool opstatus](#).

For each operation the **Operation Status** page displays the Operation Name, Target node (identified as `<region-name>::<datacenter-name>::<hostname>`), and current Status, as well as the Progress (as an approximate percentage of completion), operation Start Time, and Last Update time (the last time that the CMC obtained status information for the operation).

The operation Status will be one of In-Progress, Completed, Failed, or Terminated. A Terminated status is applicable only for *hssstool repair*, *hssstool repairec*, *hssstool cleanup*, or *hssstool cleanupec* operations that an operator has terminated by the "stop" command option that's supported for those operation types.

- **To refresh** the display, click the refresh icon above the **Search** field.



- To view status detail for an operation, click the **View** button to the right of the operation status summary.

OPERATION NAME	TARGET	STATUS	START TIME	LAST UPDATE
cleanupec	region1-dct1-store1	completed	Feb-23-2018 07:25	Feb-23-2018 07:25

100%

```

CLEANUPEC cmdno#: 1 status: COMPLETED
arguments: deleteobject-without-bucketinfo=false check-protection=true deletedata=true deleteobject-without-policy=false
logging=true delete-only-outofrange-objects=false
start: Fri Feb 23 04:25:54 PST 2018
end: Fri Feb 23 04:25:54 PST 2018
duration: 0.014 sec
progress percentage: 100%
time remaining: 0 ms
task count: 0
completed count: 0
deleted count: 0
failed count: 0
skipped count: 0

Operation succeeded.

```

In the case of *hsstool* operations these are the same operation details as are provided by [hsstool opstatus](#), including the estimated time remaining to complete the operation (if the operation is not yet complete). To refresh the status detail, close the detail view and then reopen it.

- To filter the operation list, in the **Search** field enter an operation name, region name, data center name, or hostname.
- To delete an operation status line from the page click **Delete** to the right of the line. In the case of *hsstool* operations the status detail will still be available through the [hsstool opstatus](#) command.

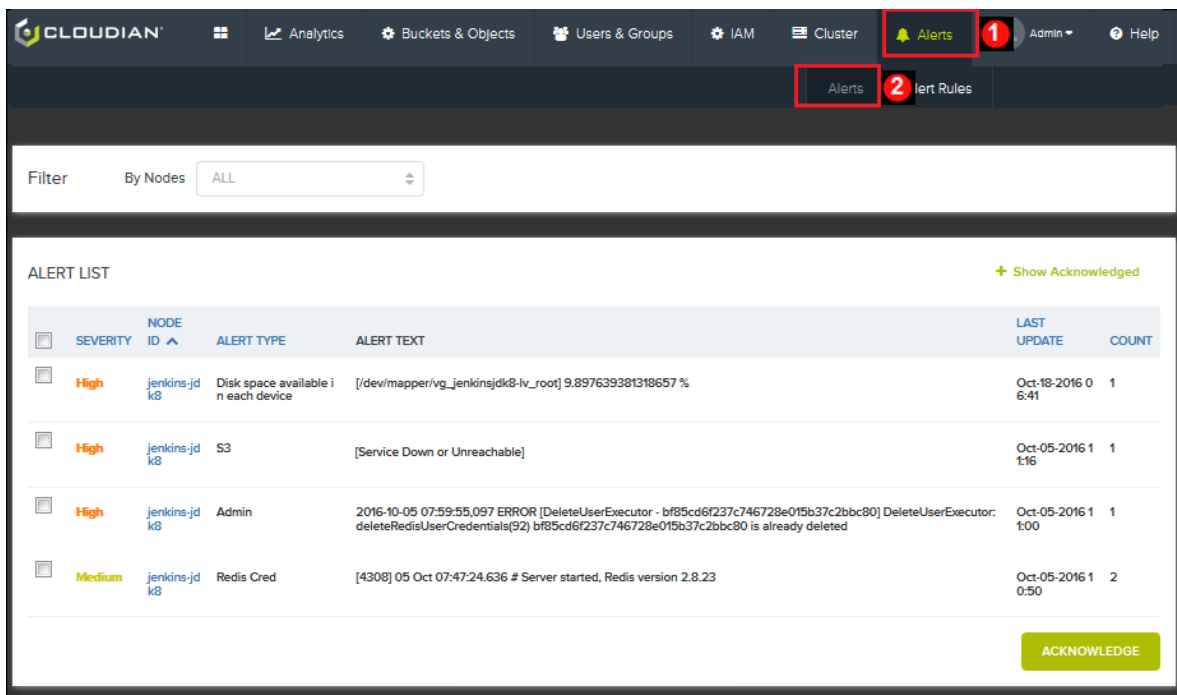
6.11. Alerts

The **Alerts** tab contains the following functions:

- ["Alerts"](#) (page 555)
- ["Alert Rules"](#) (page 560)

6.11.1. Alerts

Path: **Alerts** → **Alerts**



Supported tasks:

- ["Review Alerts" \(page 556\)](#)
- ["Acknowledge Alerts" \(page 559\)](#)

6.11.1.1. Review Alerts

In the **Alerts** page you can review active alerts that have been generated by your HyperStore nodes. Alerts are triggered by the occurrence of node events or conditions for which alert rules have been configured. Your HyperStore system comes with a set of pre-configured alert rules which are listed in the CMC's [Alert Rules](#) page. In that page you can also edit the pre-configured rules if you wish, or create additional alert rules.

The **Alerts** page auto-refreshes once per minute. For each alert the following information is displayed:

Severity

Each alert is assigned a severity level of Critical, High, Medium, or Low. The severity level assigned to each alert is configurable in the [Alert Rules](#) page.

Node ID

The HyperStore node on which the alert occurred. Note that at the top of the **Alerts** page you can filter the **Alert List** by node. By default, "ALL" is selected, to show results for all nodes in your system.

You can click on the Node ID to jump to the ["Node Status" \(page 471\)](#) page for that node.

Alert Type

- **Statistic threshold alerts** — Alerts indicating that a monitored performance statistic has crossed a threshold. For this category of alert, the "Alert Type" field indicates the statistic for which a configured threshold was crossed (such as CPU utilization, Disk space available on node, Disk space available on device, Number of S3 GET transactions per second, and so on).

- **Service down/unreachable alerts** — Alerts indicating that a service is either down or unreachable by the HyperStore monitoring system. For this category of alert, the "Alert Type" field indicates the service type that is down or unreachable -- Admin, Cassandra (Metadata DB), HyperStore, Redis Credentials (Credentials DB), Redis QoS (QoS DB), or S3. (The system also supports alerts for when such a service is restored or reachable again -- these are not pre-configured in the system but you can add alert rules for these if you wish).

Note To determine whether a service is actually down on a node (as opposed to being up but unreachable by the monitoring system), log into the node and run:

```
systemctl is-active <servicename>
```

The <servicename> string can be one of {cloudian-s3, cloudian-cassandra, cloudian-hyperstore, cloudian-redis-credentials, cloudian-redis-qos, cloudian-redismon, cloudian-cmc, cloudian-agent, cloudian-coreDNS}.

- **Log message alerts** — Alerts indicating that a WARN or ERROR level message has been written to a service application log. For this category of alert, the "Alert Type" field indicates the service for which the WARN or ERROR occurred -- for example the Admin Service or the S3 Service.

Note For the Admin, HyperStore, Redis Monitor, S3, and Phone Home services, alerts based on log messages are filtered as described in ["Filtering of Log Message Based Alerts"](#) (page 569).

- **Repair completion alerts** — Alerts indicating that a data repair operation has finished its run. For this category of alert, the "Alert Type" field displays "Repair Completion Status". Note that the completion of routine "proactive" repairs does not trigger these alerts -- only completion of scheduled auto-repairs or repairs operations that you execute yourself (from the CMC or the command line) will trigger these alerts. For more information on repair types see .
- **Disk error alerts** -- Alerts indicating that a HyperStore data disk read/write error has been detected and/or the disk has been disabled. For this category of alert, the "Alert Type" field displays "Disk Error". To learn more about the status of a data disk for which a Disk Error alert has been generated, go to the ["Node Status"](#) (page 471) page and select the node on which the disk resides, then view the [Disk Detail Info](#) section of the page.

Note For HyperStore Appliances only, an alert is also triggered if an SSD (storing the OS and metadata) fails.

For more details about the types of alerts that HyperStore monitors see ["Alert Rules"](#) (page 560).

Alert Text

- For **statistic threshold alerts**, this field will indicate the statistic's value which caused an alert rule to be triggered.
- For **service down/unreachable alerts**, this field will be a text string "[Service Down or Unreachable]"
- For **disk error alerts**, this field indicates the mount point for the disk that's had an error.
- For **log message alerts**, this field will be the full text of the log entry (truncated if the log message is larger than 256 characters).

- For general information about HyperStore log entry formatting, see ["HyperStore Logs"](#) (page 667).
- Most log entries of level WARN or ERROR include an alphanumeric **message code** that uniquely identifies the log message (either "HSxxxxxx" or "DCxxxxxx" or "RMxxxxxx" or "ISxxxxxx"). Hold your cursor over the message code in the **Alerts** interface to display "tool tip" text with the log level and recommended corrective action for the message code.)

Note For log message based alerts that have occurred multiple times without being acknowledged, the "Alert Text" -- including the log entry timestamp -- will be from the most recent instance of the log message.

Last Update

This field shows the local date and time at which the alert was detected by the HyperStore monitoring system.

For alerts that have occurred multiple times without being acknowledged (as indicated by the "Count" value), the "Last Update" field indicates when the **most recent instance** of the alert was detected.

Count

The number of unacknowledged times that the same alert has occurred.

- For **statistic threshold alerts**, a Count value greater than "1" means that the monitoring system has detected the statistic to be across its configured threshold multiple times, without being acknowledged.

Note Each node's CPU utilization, disk space availability, and total network throughput are checked each minute. For each of these statistics, each time that the once-per-minute checks finds that the statistic is across its notification threshold, the Count for the alert is incremented. For example, if you have an alert rule for "CPU utilization > 50%", and if the monitoring system's once-per-minute checks find three different instances where the node's CPU utilization was in excess of 50%, then the **Alert List** will show one line for the "CPU Utilization" alert type, with a Count of 3.

For S3 statistics — GET/PUT transactions per second, GET/PUT throughput, and GET/PUT average latency — the statistics are calculated at the node every **five** minutes. However, the alert monitoring system retrieves the values each minute, just like for other statistics. So, the monitoring system retrieves the same calculated S3 statistics five times, until the next set of S3 statistics is calculated. If an S3 statistic value is across a notification threshold, the monitoring system retrieves that same statistic value five times, and consequently shows a Count of 5. So for alerts based on S3 statistic thresholds, the Count will overstate how often the alert has actually happened, by as much as a factor of 5.

- For **service down/unreachable alerts** the Count increments each 60 seconds for as long as the service is down or unreachable by the HyperStore monitoring system.
- For **log message alerts**, a Count value greater than "1" means that the exact same log message has occurred multiple times without being acknowledged.

Note The monitoring system checks the logs every 30 seconds for new warning or error messages. If a log is rotated during the 30 second interval between one check and the next, it's

possible that an instance of a log message may be missed by the monitoring system. In this alert the Count value may not be exactly correct, particularly in circumstances where many error messages are occurring and logs are being rotated more frequently than usual.

- For **repair completion alerts**, a Count value greater than "1" means that multiple repair completion alerts of the specified repair type (REPAIR, EC:REPAIR, or CASSANDRA:REPAIR) have occurred on the same node without being acknowledged. Note that if the Count is greater than "1", the Alert Text will show only the status of the first-finished repair operation. To view status detail on all recently run repairs use *hssstool opstatus* or check the CMC's **"Repair Status"** (page 549) page.

Note The alert list is sorted primarily by alert Severity (high to low) and secondarily by reverse chronological order (based on the time of the most recent instance of each listed alert, as indicated in the Last Update column). You can re-sort the alert list by any column except for Alert Text by clicking the column heading. Click once for ascending order, click a second time for descending order.

6.11.1.2. Acknowledge Alerts

In the **Alerts** page you can acknowledge that you have seen and reviewed the node alert notifications that display in the **Alert List**. Once you acknowledge an alert, it will no longer display in the **Alert List** (unless you choose to display acknowledged alerts as described further below).

- To acknowledge one or multiple alerts in the **Alert List**, click the checkbox to the left of the alert(s) and then click **Acknowledge** at the bottom of the list. Note that as you are selecting alerts to acknowledge (by clicking checkboxes), the **Alerts** page's auto-refresh feature will temporarily pause, so as not to clear your checkbox selections.
- To acknowledge **all** alerts, click the checkbox at the left side of the **Alert List** column heading row and then click **Acknowledge** at the bottom of the list.
- To show previously acknowledged alerts in the list as well as unacknowledged alerts, click **Show Acknowledged** at the top of the alert list. Previously acknowledged alerts then display within the list, and are distinguished from unacknowledged alerts by the absence of a checkbox to the left of the alert. You can click **Hide Acknowledged** to hide the acknowledged alerts again.

The **Alert List** can display a maximum of 50 unacknowledged alerts and acknowledged alerts combined. Note that some of these 50 alerts may have occurred multiple times as indicated by the alert's "Count" value. Even if an alert has a Count showing that it has occurred multiple times, this counts as only one alert toward the **Alert List** display maximum of 50 alerts.

Note Acknowledged alerts are automatically deleted from the system after a time period configured by the **"monitoring.event.acknowledgedTTI"** (page 807) setting. By default this period is 86400 seconds (one day). After they are deleted acknowledged alerts will not display in the Alert List even if you click **Show Acknowledged**. If you wish you can reduce the configurable time-to-live for acknowledged alerts to as little as 1 second (so that they are deleted from your system right after acknowledgment). Note that regardless of your configured time-to-live for acknowledged alerts, a record of your system's alert history will persist in the **Smart Support** logs that by default are uploaded to Cloudian Support each day.

Note If you acknowledge an alert for which the underlying notification triggering condition still exists, a new alert may be generated quickly. For example, if a service is down, and you acknowledge the service down alert but do not restart the service, a new service down alert will be triggered as soon as the monitoring system polls the service status again (which occurs each minute). If you want to temporarily disable an alert rule you can do so in the [Alert Rules](#) page, but be sure to remember to re-enable the rule at the appropriate time.

6.11.2. Alert Rules

Path: **Alerts** → **Alert Rules**

ALERT RULES

CREATE ALERT RULE

Alert Type: Throughput for GET operations

Condition: > 1 BYTE PER SECOND

Target Email: [Empty field]

Severity Level: Medium

☒ USE DEFAULT EMAIL ADDRESS

☐ SEND SNMP TRAP

CREATE

ACTIVE ALERT RULES

GENERAL STATUS				
	send email to	send snmp trap	severity level	actions
☐ CPU Utilization greater than 90.0 %	default		Medium	Edit Delete
☐ Disk Error	default		Critical	Edit Delete
☐ Disk space available in node less than 10.0 %	default		High	Edit Delete
☐ Disk space available in each device less than 15.0 %	default		High	Edit Delete
☐ Node Unreachable	default		Critical	Edit Delete
☐ Repair Completion Status	default		Low	Edit Delete

SERVICE STATUS				
	send email to	send snmp trap	severity level	actions
☐ Admin service error	default		High	Edit Delete
☐ Admin service down	default		High	Edit Delete

Supported tasks:

- ["Review Supported Rule Types and Pre-Configured Rules"](#) (page 561)
- ["Add an Alert Rule"](#) (page 567)
- ["HyperStore SNMP Trap Content"](#) (page 568)
- ["Edit an Alert Rule"](#) (page 569)
- ["Disable an Alert Rule"](#) (page 569)
- ["Delete an Alert Rule"](#) (page 569)

- ["Filtering of Log Message Based Alerts"](#) (page 569)
- ["Putting a Node or DC into Maintenance"](#) (page 614)

6.11.2.1. Review Supported Rule Types and Pre-Configured Rules

Alert rules specify the system conditions that will trigger HyperStore alerts. HyperStore supports the alert rule types described in the tables below. Alert codes (indicated in the "Alert Code" column of the tables) are alphanumeric codes that are included in the alerts to aid in alert identification. As indicated by the "Pre-Configured?" column, for some alert rule types HyperStore comes with pre-configured rules that have already been created for you and are active in the system. These pre-configured rules are listed in the main part of the **Alert Rules** page when you first access the CMC. All pre-configured rules include sending a notification email to the default system administrator email address. If you want to activate an alert rule that is not pre-configured in the system you can do so as described in ["Add an Alert Rule"](#) (page 567).

For all active rules, an alert is generated if the specified condition occurs **on any node in the system**. All rules are based on node conditions (as opposed to aggregate, system-wide conditions).

Network Status Rules

Rule Type	Description	Alert Code	Pre-Configured?
Number of GET transactions per second	For each node, every five minutes the system calculates the average number of S3 GETs processed per second, based on the last approximately 1000 S3 GET transactions. Alert rules can be based on this average exceeding or falling below a user-defined threshold.	NS000001	No
Number of PUT transactions per second	Same as above, except for S3 PUTs.	NS000002	No
Throughput for GET operations	For each node, every five minutes the system calculates the average throughput for S3 GETs in bytes per second, based on the last approximately 1000 S3 GET transactions. Alert rules can be based on this average exceeding or falling below a user-defined threshold.	NS000003	No
Throughput for PUT operations	Same as above, except for S3 PUTs.	NS000004	No
Latency for GET operations	For each node, every five minutes the system calculates the 95th percentile for transaction latencies of S3 GETs, in milliseconds, based on the last approximately 1000 S3 GET transactions. The 95th percentile latency value indicates that of the last 1000 S3 GET transactions, 95% completed in that many milliseconds or less. Alert rules can be based on this value exceeding or falling below a user-defined threshold.	NS000005	No
Latency for PUT operations	Same as above, except for S3 PUTs.	NS000006	No
Network throughput	For each node, each minute the system calculates the	NS000007	No

Rule Type	Description	Alert Code	Pre-Configured?
(incoming)	average number of bytes of incoming network throughput per second, based on the last minute of activity. Alert rules can be based on this value exceeding or falling below a user-defined threshold. Note Network throughput statistics are based on all data moving into or out of a node — not just S3 request data. For example, data transmission associated with cluster maintenance operations would count toward these statistics.		
Network throughput (outgoing)	Same as above, except for outgoing throughput.	NS000008	No

General Status Rules

Rule Type	Description	Alert Code	Pre-Configured?
Disk space available in node	For each node, each minute the system calculates the aggregate available disk space for the node's data disks, as a percentage of the total capacity of the data disks. Alert rules can be based on this value falling below a user-defined threshold. Note In calculating this statistic the system does not count "reserved" disk capacity as being available. Instead, available capacity is what's left after deducting both used capacity and reserved capacity from the total capacity. For information about "reserved" capacity see "Capacity Managed" (page 285).	GS000003	Yes -- a High severity alert if disk space available on a node is less than 10%
Disk space available in each device	Same as above, except for each individual data disk on each node.	GS000004	Yes -- a High severity alert if space available on a disk is less than 15%
Disk error	With this alert rule, an alert is triggered whenever an "HSDISKERROR" or "HSDISKDISABLED" message appears in the HyperStore Service application log (<i>cloudian-hyperstore.log</i>) on any node.	GS000005	Yes -- a Critical severity alert

Rule Type	Description	Alert Code	Pre-Configured?
	Note In the case of a multipath (MPIO) disk, a disk error will also be triggered if one of the paths to the disk goes down.		
Node unreachable	With this alert rule, an alert is triggered if a node cannot be reached by the HyperStore monitoring system.	GS000001	Yes -- a Critical severity alert
Load average (5 minutes)	For each node, every five minutes the system calculates the Linux load average (average number of processes in execution or queued for execution) for the node as a whole during the past five minutes. Alert rules can be based on this average exceeding a user-defined threshold. Note Be sure to take into account the number of processing cores on your HyperStore hosts when setting a threshold for alerts based on load average.	GS000007	No
CPU utilization	For each node, each minute the system checks the node's average CPU utilization level during the past minute. Alert rules can be based on this value exceeding a user-defined threshold.	GS000006	Yes -- a Medium severity alert if CPU utilization is greater than 90%
Repair completion status	With this alert rule, an alert is triggered whenever a data repair operation finishes its run. The alert will indicate the operation's finishing status: COMPLETED, FAILED, or TERMINATED. Note The completion of routine "proactive" repairs does not trigger these alerts. For more information on repair types see " Automated Data Repair Feature Overview " (page 130).	GS000002	Yes -- a Low severity alert

Service Status Rules

Rule Type	Description	Alert Code	Pre-Configured?
Admin Service status	For the Admin Service, separate alert rules can be set for any of these statuses: Service is down or unreachable	SS000001 for down or unreachable	Yes -- a High severity alert if the

Rule Type	Description	Alert Code	Pre-Con-figured?
	• Service is back up after being down or unreachable • Service logs an error • Service logs a warning	HSxxxxxx for log errors or warnings (codes are specific to log event)	service is down, and a High sever- ity alert if the service logs an error Note For the Adm- in, Hyp- erSt- ore, Redi- s Mon- itor, S3, Cron and Mon- itor- ing, and Pho- ne Hom- e ser- vice- s, alert- s base- d on log mes- sage- s are filter- ed as desc- ribed in

Rule Type	Description	Alert Code	Pre-Configured?
			"Filtering of Log Message Based Alerts" (page 569).
Cassandra Service status	For the Cassandra Service (Metadata DB), separate alert rules can be set for any of these statuses: • Service is down or unreachable • Service is back up after being down or unreachable • Service logs an error • Service logs a warning	SS000002 for down or unreachable	Yes -- a High severity alert if the service is down, and a High severity alert if the service logs an error
HyperStore Service status	For the HyperStore Service , separate alert rules can be set for any of these statuses: • Service is down or unreachable • Service is back up after being down or unreachable • Service logs an error • Service logs a warning	SS000003 for down or unreachable HSxxxxxx for log errors or warnings (codes are specific to log event)	Yes -- a High severity alert if the service is down, and a High severity alert if the service logs an error
Redis QoS Service status	For the Redis QoS Service (QoS DB), separate alert rules can be set for any of these statuses: • Service is down or unreachable • Service is back up after being down or unreachable • Service logs a warning	SS000004 for down or unreachable	Yes -- a High severity alert if the service is down, and a Medium severity alert if the service logs a warning
Redis Credentials Service status	For the Redis Credentials Service (Credentials DB), separate alert rules can be set for any of these statuses:	SS000005 for down or unreachable	Yes -- a High severity alert if the service is down, and a Medium severity alert if the service logs a warning

Rule Type	Description	Alert Code	Pre-Configured?
	- Service is down or unreachable - Service is back up after being down or unreachable - Service logs a warning		ity alert if the service is down, and a Medium severity alert if the service logs a warning
Redis Monitor Service status	For the Redis Monitor Service, separate alert rules can be set for any of these statuses: - Service is down or unreachable - Service is back up after being down or unreachable	SS000006 for down or unreachable RMxxxxxx for log errors or warnings (codes are specific to log event)	Yes -- a High severity alert if the service is down
S3 Service status	For the S3 Service, separate alert rules can be set for any of these statuses: - Service is down or unreachable - Service is back up after being down or unreachable - Service logs an error - Service logs a warning Note Along with alerts pertaining to providing S3 service to clients, the S3 service alerts category includes alerts pertaining to auto-tiering and cross-region replication.	SS000007 for down or unreachable HSxxxxxx for log errors or warnings (codes are specific to log event)	Yes -- a High severity alert if the service is down, and a High severity alert if the service logs an error
Cron Mon Service status	For the service that monitors <i>cron.d</i> (which triggers the various HyperStore maintenance cron jobs), separate alert rules can be set for either of these statuses: - Service logs an error - Service logs a warning	SS000008 for down or unreachable	Yes -- a High severity alert if the service logs an error
Phone Home Service status	For the Phone Home (Smart Support) service, separate alert rules can be set for either of these statuses: - Service logs an error - Service logs a warning	SS000009 for down or unreachable	Yes -- a High severity alert if the service logs an error
K3S Service	- Service is down or unreachable - Service is back up after being down or	SS000010 for down or	No

Rule Type	Description	Alert Code	Pre-Configured?
(Kubernetes on File Service nodes and Search Service nodes)	unreachable	unreachable	
File Service	- Service is down or unreachable - Service is back up after being down or unreachable	SS000011 for down or unreachable	No
Search Service	- Service is down or unreachable - Service is back up after being down or unreachable	SS000012 for down or unreachable	No

6.11.2.2. Add an Alert Rule

To add a new alert rule, in the **Alert Rules** page do the following:

1. From the "Alert Type" drop-down list, select an alert type.

The screenshot shows the Cloudian Alert Rules configuration page. The 'Alert Type' dropdown is open, displaying a list of alert types including 'Network Status', 'Number of Get transactions per second', 'Throughput for GET operations', 'Throughput for PUT operations', 'Latency for GET operations', 'Latency for PUT operations', 'Network throughput (incoming)', 'Network throughput (outgoing)', 'General Status', 'Disk space available in node', 'Disk space available in each device', 'Disk Error', 'Node Unreachable', 'Load Average (5 Min)', 'CPU Utilization', 'Repair Completion Status', 'Service Status', and 'Admin service status'. The 'Severity Level' is set to 'Medium'. The 'CREATE' button is visible. Below the form, a table lists existing alert rules with columns for 'send email to', 'send snmp trap', 'severity level', and 'actions'.

send email to	send snmp trap	severity level	actions
default		Medium	Edit Delete
default		Critical	Edit Delete
default		High	Edit Delete
default		High	Edit Delete
default		Critical	Edit Delete

2. Configure a rule for that alert type. A rule defines the conditions that will trigger the alert. The options (as presented in the rule-configuring interface) will vary according to the alert type. For descriptions of alert types see ["Review Supported Rule Types and Pre-Configured Rules"](#) (page 561).
3. If you want the alert to include sending an **email notification** to the default system administrator email address(es), leave the "Use Default Email Address" option checked. If you want to customize the target email addresses for this particular alert, uncheck the "Use Default Email Address" option and enter the address(es) in the "Target Email" field (for multiple addresses, use comma separation). If you do not want email notification for this alert, uncheck the "Use Default Email Address" option and leave the "Target Email" field empty.

Note The default system administrator email address is configured in the ["SMTP/Email Settings for Alerts/Notifications"](#) (page 498) section of the CMC's **Configuration Settings** page. Alert emails are sent by the [HyperStore System Monitoring / Cron Job host](#), using your specified SMTP server.

4. If you want the alert to include sending an **SNMP trap** to your SNMP management system, select the "Send SNMP Trap" checkbox. For more information about HyperStore SNMP traps see ["HyperStore SNMP Trap Content"](#) (page 568).
5. Select a **Severity level** to assign to the alert. You can choose from Critical, High, Medium, or Low.

Note Subsequently, if the alert occurs and is displayed in the [Alerts](#) page, the display includes the severity level that you assigned to the alert. Also, in the [Alerts](#) page you will be able to sort the displayed alert list by severity level.

6. Click **Create**.

Your new alert rule then displays in the **Active Alert Rules** section of the page.

IMPORTANT ! If your HyperStore system has multiple service regions, a drop-down list displays at the top of the **Alert Rules** page so you can select a region for which to manage alert rules. Any alert rule actions that you take in one region — such as adding, editing, or disabling a rule — will **not** carry over to the other region(s). If you want an alert rule change to apply to all of your regions, you must make that change **for each region one at a time**, using the **Alert Rules** page.

6.11.2.3. HyperStore SNMP Trap Content

- Trap destination settings are configured in the ["SNMP Trap Destination Settings"](#) (page 500) section of the CMC's **Configuration Settings** page. The destination that you configure there will be used for all traps that you enable in your alert rules.
- HyperStore uses SNMP version **2c**.
- Traps are sent by the [HyperStore System Monitoring / Cron Job host](#).
- Traps sent by HyperStore contain the following variable bindings (valbinds):
 - sysUpTime (hardcoded to 50 seconds; ignore this field)
 - snmpTrapOID (always 1.3.6.1.4.1.16458.4.1.1.1)
 - sysDescr (always Cloudian HyperStore)
 - hsTrapSeverity (an integer: 0 for Low, 1 for Medium, 2 for High, 3 for Critical)
 - hsTrapHost (HyperStore host on which the condition occurred)
 - hsTrapType
 - hsTrapMessage
 - hsTrapCondition
 - hsTrapValue (may or may not be present depending on the type of trap)
 - hsTrapDevice (present only if the trap is in regard to a disk problem)

6.11.2.4. Edit an Alert Rule

In the **Alert Rules** page you can edit existing alert rules — for example to change a threshold value, change the severity level assigned to a particular rule, or change notification settings for a particular rule.

Note If you want to change the default email address(es) that alert emails are sent to, go to the CMC's [Configuration Settings](#) page and edit the "Default Email Address to Receive Notifications" setting. You do not need to make any change on the **Alert Rules** page to do this. The same is true for changing the destination for sending SNMP traps: do it on the **Configuration Settings** page.

To edit a rule:

1. Click **Edit** next to the rule in the **Rules** list.
2. Modify the rule as desired.

Note To change from using the default target email address to using a different email address specifically for this rule, uncheck the "Use Default" option to the right of the rule statement and then enter the custom email address in the text field. To disable email notification for just this rule, leave the text field empty.

3. Click **Done** to apply your changes.

Your modified rule will then display in the **Rules** list.

6.11.2.5. Disable an Alert Rule

To temporarily disable an alert rule, in the **Alert Rules** page select the checkbox to the left of the rule and then click **Disable** on the lower right of the page. If you want you can disable multiple rules at the same time by selecting multiple rules' checkboxes and then clicking **Disable**.

A disabled rule will remain in your **Rules** list display, but the rule will no longer be applied by the HyperStore system. In the rules list a disabled rule is distinguished by its name being grayed out.

To re-enable a rule that has been disabled, select the checkbox to the left of the rule and then click **Enable** on the lower right of the page. The HyperStore system will resume applying the alert rule.

6.11.2.6. Delete an Alert Rule

To delete an alert rule, in the **Alert Rules** page click **Delete** to the right of the rule in the **Rules** list. The rule will no longer display in the **Rules** list and will no longer be applied by the HyperStore system.

If you want you can delete multiple rules at the same time by selecting the checkboxes to the left of those rules and then clicking **Delete** on the lower right of the page.

6.11.2.7. Filtering of Log Message Based Alerts

For the Admin Service, HyperStore Service, Redis Monitor Service, S3 Service, and Phone Home Service, application log error messages and warning messages will trigger alerts only if **both** of the following are true:

- Alerts for that log message severity, for that service, are enabled in the CMC **Alert Rules** page. (By default, alerts are enabled for Error messages for each of these services, but not for Warning messages. You can modify this configuration in the **Alert Rules** page if you wish -- for example you could enable alerts for Warning messages from the S3 Service.)
- The message code for the particular log message matches against one of the message codes specified by the configuration setting "**monitoring.alerts.active**" (page 805). This setting is a filtering mechanism to reduce the number of log messages that can result in alerts.

A second filtering mechanism is supported if you have [Cloudian HyperIQ](#): You have the option of suppressing CMC alerting for alerts that appear in HyperIQ (to avoid duplication of alerting across the CMC and HyperIQ). By default this duplicate suppression is disabled. You can enable it in the CMC's "**Configuration Settings**" (page 498) page, in the "Alert Settings" section (or you prefer to use *hsctl config* on the command line instead, the setting is "**monitoring.alerts.suppressHIQDuplicates**" (page 806).)

See Also:

- "**How HyperStore Implements Alerts**" (page 570)

6.11.3. How HyperStore Implements Alerts

In the CMC's [Alert Rules](#) page you can create rules for having the HyperStore system generate alerts when specified events occur. When an event covered by an alert rule occurs, the HyperStore system:

- Sends an SNMP trap, if the alert rule for the event includes sending an SNMP trap. The trap is sent by the [HyperStore System Monitoring / Cron Job host](#), to the SNMP destination configured in the "**SNMP Trap Destination Settings**" (page 500) section of the CMC's **Configuration Settings** page.
- Displays an alert in the CMC's [Node Status](#) page (in the **Alert List** for the node on which the event occurred) and also in the [Alerts](#) page.
- Emails an alert notification to specified system administrator addresses (unless you disable email notification, which you can do on a per-rule basis as described in "**Alert Rules**" (page 560)). The alert email is sent by the [HyperStore System Monitoring / Cron Job host](#), using your specified SMTP server.

Alerting through the CMC and through email and SNMP is implemented as follows:

- When the event that is specified by the alert rule occurs, an alert email is sent to the configured email address(es), and a trap is sent if you've enabled SNMP as part of the rule. An alert notice is also displayed on the [Node Status](#) page and on the [Alerts](#) page.
- The sending of the alert email sets off a 24 hour hold on any further alert emails in association with the same alert rule on the same node. The same hold period applies to SNMP trap sending. The 24 hour hold works like this:
 - So long as the original instance of the alert remains unacknowledged by administrators, any additional events that trigger the same alert rule on the same node will not generate additional email notifications (or SNMP traps). Such additional event instances will only generate additional alerts in the CMC interface, so that the alert's "Count" value increments.
 - As soon as an administrator acknowledges the alert in the CMC's **Node Status** page or in the **Alerts** page, the 24 hour hold is lifted and any new instances of the event will trigger a new email notification (and SNMP trap if enabled).

- If 24 hours pass without the original alert being acknowledged by an administrator, then the next subsequent instance of the event will trigger the sending of a new alert email (and SNMP trap if enabled). This then initiates a new 24 hour hold period.

6.11.3.1. Handling of Alerts for ERROR or WARN Log Messages

For alert rules based on ERROR messages in service logs, an alert is written to the CMC's [Node Status](#) page and [Alerts](#) page for **each individual ERROR message** that occurs in the service logs. For alert rules based on WARN messages in service logs, the alerts are written for each individual WARN message that occurs in the service logs and also for each individual ERROR message.

However, for purposes of email notification, each additional log message from the same service is considered the same type of "event", and consequently the 24 hour hold described above applies. For example, if you have an alert rule in place for ERROR messages in the S3 service logs:

- If ERROR message "X" appears in the S3 service log on node1, an alert is written to the CMC's **Node Status** and **Alerts** pages, and an email notification is sent.
- If 10 minutes later ERROR message "X" appears again in the S3 service log on node1, another alert is written to the CMC's **Node Status** and **Alerts** pages, but no additional email notification is sent.
- If 10 minutes later ERROR message "Y" appears in the S3 service log on node1, another alert is written to the CMC's **Node Status** and **Alerts** pages, but no additional email notification is sent. Although ERROR message "Y" is different from ERROR message "X", it's still considered to be part of the same alert rule and consequently it does not trigger an additional email notification.

Note In describing alerting behavior for the second and third messages, the scenario above presumes that no administrator has yet acknowledged the earlier ERROR message alert. Recall that acknowledging an alert releases the 24-hour hold on additional emails for that alert type, in which case a new instance of the underlying event triggers a new email notification.

6.12. My Account

In the upper right of the CMC interface, holding your cursor over your user name displays a drop-down menu from which you can choose these options relating to your user account:

- [Profile](#)
- [Security Credentials](#)

6.12.1. Profile

Path: Drop-down menu under your user name (at top right) → **Profile**

ACCOUNT PROFILE

USER ID: admin	GROUP NAME: 0	USER TYPE: SystemAdmin
FULL NAME: System Admin	EMAIL ADDRESS: 	PHONE NUMBER:
ADDRESS LINE 1: 	ADDRESS LINE 2: 	
CITY: 	STATE, PROVINCE, REGION: 	ZIP OR POSTAL CODE:
COUNTRY: 	WEB SITE URL: 	

SAVE

Supported task:

- Change your contact information

Change Your Contact Information

1. Update your contact information.
2. Click **Save**.

Note Changing your system administrator email address in this page does not impact the sending of system notification emails. To change the email address to which notification emails are sent, use the [Default Email Address to Receive Notifications](#) setting on the **Configuration Settings** page.

6.12.2. Security Credentials

[CMC Interface]

Path: Drop-down menu under your user name (at top right) → **Security Credentials**

Supported tasks:

- ["Change Your Console Password"](#) (page 573)
- ["Enable or Disable Multi-Factor Authentication \(MFA\)"](#) (page 574)
- ["Manage Your S3 Access Keys"](#) (page 577)

6.12.2.1. Change Your Console Password

Note If the system is configured to use LDAP authentication for your user group, the **Change Password** function will not be available and the instructions below are not applicable to you. Instead, your password should be controlled through your organization's LDAP system.

1. In the "Current Password" field, enter your current password.
2. In the "New Password" field, enter a new password that you will use to log into the CMC . Then in the "Confirm Password" field, enter the new password again.

Passwords must meet the following conditions by default:

- Minimum of nine characters, maximum of 64 characters
- Must contain:
 - At least one lower case letter
 - At least one upper case letter
 - At least one number
 - At least one special character such as !, @, #, \$, %, ^, etc.

Note You can optionally configure HyperStore to require a higher minimum password length. You can also optionally configure additional password restrictions such as a password expiration period, a restriction against a user's new password being too similar to their previous

password, a restriction on password reuse, and a restriction against too-frequent password changes. In the [common configuration setting group](#), these restrictions are configurable by the `common.user.password.*` settings.

3. Click **Change Password**.

6.12.2.2. Enable or Disable Multi-Factor Authentication (MFA)

If you wish you can add another security layer to your CMC access by enabling multi-factor authentication (MFA). If you enable MFA for your account, then during the CMC login process you will be required to enter an MFA code (after you enter your user name and password).

Before enabling MFA for your account you must install a third party virtual MFA device -- a software application that generates MFA codes -- for the browser from which you will be accessing the CMC. This can be any MFA application that supports the open TOTP (time-based one-time password) standard, also known as RFC 6238. Examples of free MFA applications include Google Authenticator and Microsoft Authenticator.

Note

* If "[cmc.mfa.enforced](#)" (page 723) is set to `true` in your system configuration (by default it is `false`), then when you try to log into the CMC you will be presented with the **Set up Virtual MFA Device** screen described in Step 2 below if you have not already set up MFA.

* MFA is supported for CMC users whose logins are authenticated by way of an LDAP system.

* SMS based MFA (MFA in which a text message is sent to your cell phone) is not supported.

After installing a third party MFA application (virtual MFA device) for your browser, do the following to enable MFA for your HyperStore CMC account:

1. In the "Sign-In Credentials" section of the **Security Credentials** page, under "Multi-Factor Authentication", click where it currently says **Disabled**.

The screenshot shows the 'SIGN-IN CREDENTIALS' form. It includes a 'USER ID' field with the value 'PubsUser1'. Below it are 'NEW PASSWORD' and 'CONFIRM PASSWORD' fields. To the right, there is a 'CURRENT PASSWORD' field. A red rectangular box highlights the 'MULTI-FACTOR AUTHENTICATION (MFA)' section, which contains a red circle icon and the text 'Disabled'. At the bottom right of the form is a green button labeled 'CHANGE PASSWORD'.

2. The **Set up Virtual MFA Device** screen displays. Using your virtual MFA device, scan the QR code in the **Set up Virtual MFA Device** screen. The scanning method will depend on what type of MFA application you have; for example if you have a PC browser-based MFA application you'll do this by clicking and dragging around the QR code. (Note: The **Set up Virtual MFA Device** screen also displays a Secret Key -- if you prefer you can type the Secret Key into your MFA application rather than scanning the QR code.)

Set up virtual MFA device

Set up your MFA device by either scanning the QR Code or entering the Secret Key into your device

QR CODE



SECRET KEY

OMLL5PACRTI2BHJ65MEVXAYN6T4XQUMZ5HZ2BHKQUKUOX4PBAVNX7I6TNGGYDUWO

After setting up your device, enter two consecutive MFA codes here:

MFA Code 1

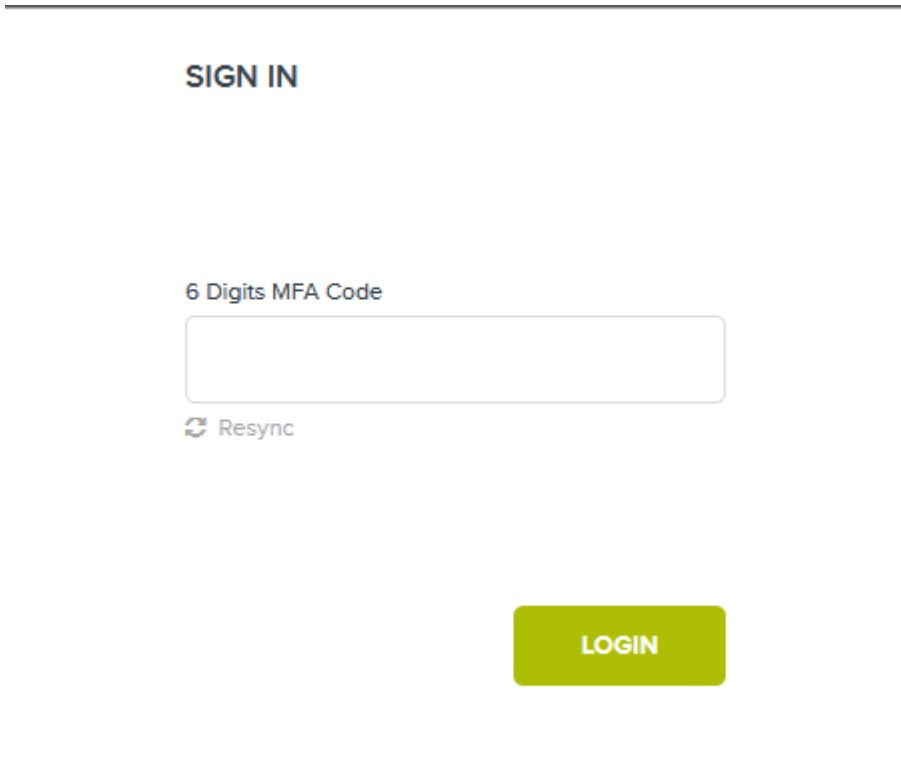
MFA Code 2

Cancel

Submit

- After you've scanned the QR code or typed the Secret Key into your MFA application, get an MFA code from the application (it will be a six-digit code), promptly enter it into the **Set up Virtual MFA Device** screen's first MFA code field, then get the next consecutive MFA code from your MFA application -- typically applications generate a new code every 30 seconds -- and promptly enter it into the **Set up Virtual MFA Device** screen's second MFA code field. Then promptly click **Submit**.

You should then see a message that MFA has been enabled for your account and that you will be automatically logged out. After the system logs you out, log back in to the CMC . During login, after you enter your user name and the correct password you will be prompted to enter the current MFA code from your MFA application:



Enter the code and you will be logged in to the CMC .

Note that:

- You can **disable MFA** for your account, in the **Security Credentials** page: Use the slider to change from **Enabled** to **Disabled**.
- As a system administrator you can [disable MFA on a user's account](#). Note that if "[cmc.mfa.enforced](#)" ([page 723](#)) is set to *true* in your system configuration (by default it is *false*), and if you disable MFA on a user's account, the next time that user logs into the CMC they will need to set up MFA again.
- On occasion a virtual MFA device can become out of time sync with the system's internal representation of that device, and as a result you may experience failures to validate an MFA code even if you enter the code correctly. If you have three successive MFA code validation failures, the system will require you to "resync" your virtual MFA device with the system. This requires you to enter two consecutive MFA codes from your virtual MFA device (similar to what you did when you first enabled the device).

MFA validation failed 3 times. Resync is required.

SIGN IN

Type Two Consecutive MFA Codes.

MFA Code 1

MFA Code 2

RESYNC

If the "resync" fails you can try again, but note that:

- If you try and fail "resync" three times, you will need to start the login process over at the beginning, by entering your user name and password. After entering the correct user name and password you will again be prompted to enter an MFA code.
- If you go through the whole failure sequence of "user name and password -> 3 failed attempts at entering a valid MFA code -> 3 failed resync attempts" **three times**, you will be locked out of the CMC and will need to ask another system administrator to unlock your account.

Rather than getting locked out, after going through the whole failure sequence once or twice you can ask another system administrator to disable the MFA requirement on your account. Most likely there is a problem with your virtual MFA device (MFA application on your computer or mobile device). After MFA is disabled on your account you will be able to log in to the CMC without entering an MFA code. If you wish you can subsequently reinstall your virtual MFA device (or uninstall your MFA device and install a different one) and then re-enable MFA on your account, following the same procedure (in the **Security Credentials** page) that you did when you enabled MFA in the first place.

6.12.2.3. Manage Your S3 Access Keys

As a system administrator you do not have an S3 storage account but you will need S3 access credentials if you want to use the HyperStore IAM Service to perform certain read-only administrative tasks. (For more

information about this feature see [Role-Based Access to Admin API Operations](#).) Note that even with S3 credentials, system administrators cannot create buckets or perform other S3 operations.

- To create a new S3 data access key, in the **S3 Access Credentials** dialog click **Create New Key**. A new access key ID then displays in the access key list.
- To view the secret access key that corresponds to an access key ID, to the right of the access key ID click **View Secret Key**. A secret key display box appears which enables you to view your secret key and to copy it using `<Ctrl>-c` if you want to. Note that the **OK** and **Cancel** buttons have no effect other than to close the secret key display box.

Note If you view your secret key(s) multiple times during one console login session, your browser may display a "Prevent this page from creating additional dialogs" checkbox that appears as if it's part of the console UI. Do not select this checkbox. If you select this checkbox and then click **OK** you will no longer be able to view your secret access keys during your current console login session.

- To activate or deactivate an access key, to the right of the displayed access key ID click **Activate** or **Inactivate**.
- To delete an access key, to the far right of the displayed access key ID click **Delete**. You will be asked to confirm that you want to delete the key.

Applying Your New S3 Access Key to Your File Shares

If you own one or more [file shares](#) (for access by SMB or NFS client applications), then after updating your S3 access key you must do the following so that the File Service can use your new S3 access key to continue to archive each file share's data to the S3 bucket that you've associated with the share:

1. For each of your file shares, in the S3 tab of the [file share's custom properties configuration](#) update the S3 access key setting to indicate the key that the File Service should use when archiving data to the S3 bucket. The S3 tab will show a drop-down list of all of your S3 access keys.
2. For each of your file shares that is currently open, [close the share and then reopen it](#).

Chapter 7. Cluster and Node Operations

7.1. Adding Nodes

This procedure is for **adding one or more nodes to an existing HyperStore data center**. During this procedure you will:

- Prep the new node(s)
- Verify that your existing cluster is in a proper condition to add nodes
- Add the new node(s) to the cluster
- Rebalance data within the cluster

IMPORTANT !

* Within a data center, if your HyperStore system is configured with multiple rack names and you are adding nodes, make sure to **position the new nodes so that when you are all done adding nodes, each rack has the same number of nodes**. Having an imbalance in the number of nodes per rack will result in data load imbalance, such that on racks with fewer nodes there will be more stored data per node than on racks with more nodes.

* Each node that you add must have the **same networking configuration as the existing nodes** in your HyperStore system -- for example, if the existing nodes each have two NICs, one for front-end network and one for back-end network, then the new nodes must have the same configuration.

* Once you have added a node to the cluster **you cannot simply "undo" the process**. If after adding a new node to the cluster you were to change your mind about keeping the node in the cluster, you would still be required to rebalance data within the cluster and then afterwards you would need to decommission the node.

* If you want to **re-use a [removed node's](#) host machine** by adding it back to your HyperStore cluster as a new node, then before using the host machine as a new HyperStore node you must remove all data from the host and re-install the operating system.

7.1.1. Special Requirements if an Existing Node is Down

The CMC's **Add Node** function **will not work if an existing node in your cluster is down or unreachable**. If you have more nodes in your cluster than are required by your configured storage policies and one of those nodes is permanently down, follow the procedure for **"[Removing a Node](#)" (page 603)** to remove the dead node from the cluster. After you complete that procedure you can then follow this procedure for Adding Nodes. If your current cluster has only the minimum number of nodes required by your storage policies and one of those nodes is dead, contact Clouidian Support for guidance.

7.1.2. Preparing to Add Nodes

Before you add a node or nodes to your cluster, prepare by taking the actions below.

1. Make sure the new host(s) **meet requirements** for:
 - Hardware specs and operating system: See **"[Host Hardware and Operating System](#)" (page 46)**.

- Open listening ports: See "[HyperStore Listening Ports](#)" (page 57).
- 2. Make sure you have the **information you will need** to complete this procedure: each new node's host-name, IPv4 address, internal interface name (optional), and *root* user password (or *sa_admin* user password for a "Secure Appliance" with its HyperStore Shell enabled and root password disabled at the factory).
- 3. **Start the new host(s)**, if not already running.
- 4. From your Config Controller node use the *system_setup.sh* tool's **Prep New Node to Add to Cluster** function to complete network interface configuration, time zone set-up, prerequisites installation, and disk formatting for each new node.

More detail:

- a. On your Config Controller node change into the installation staging directory (*/opt/cloudian-staging/8.2*) and then launch the *system_setup.sh* tool.

```
# ./system_setup.sh
```

If you are using the HyperStore Shell

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Config Controller node you can launch the system setup tool with this command:

```
$ hspkg setup
```

Once launched, the setup tool's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

- b. In the tool's main menu select "**8**" for **Prep New Node to Add to Cluster**. When prompted provide the IP address of a new node, and then the password for logging into the node. A menu of node preparation tasks will then display.
- c. Use the node preparation task menu to prepare the node:
 - Complete the configuration of network interfaces for the node, if you haven't already.
 - Set the timezone for the node.
 - Install and configure HyperStore prerequisites on the node.
 - Set up data disks on the node with *ext4* file systems, if you haven't already. Make sure to format and mount **all** available data disks on the node. The tool also supports setting the node's Metadata DB (Cassandra) drives into a RAID1 configuration.
 - After completing the setup tasks for the node choose the "Return to Master Node" option, which returns you to the tool's main menu.
- d. Repeat steps "b" and "c" above for each new node that you're adding. When you're done, exit the *system_setup.sh* tool.

IMPORTANT ! If the new node(s) are not HyperStore Appliances and **if you do not use *system_setup.sh* to format the data disks on the new node(s)**, then in the installation staging directory on your Config Controller node you must for each new node create a text file named *<hostname>_fslist.txt* that specifies the new node's data mount points, in this format:

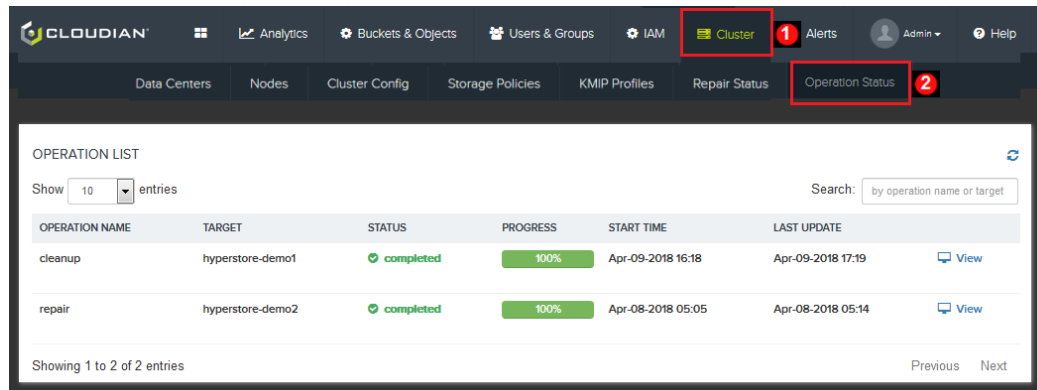
```
<devicename> <mountpoint>
<devicename> <mountpoint>
etc...
```


5. **Make sure the cluster is in proper condition** to add nodes:

- a. If there are any operations in-progress in the CMC's **Operation Status** page, wait for them to finish (or otherwise, the **Add Node** operation will automatically stop them).

More detail:

When you initiate the **Add Node** operation in the CMC (as described in "Adding Nodes" below), the system will automatically stop any in-progress *repair*, *repairrec*, *repaircassandra*, *cleanup*, or *cleanuppec* operations in the service region. If there is an in-progress operation that you do not want the system to stop, wait until the operation completes before you add nodes to your cluster.

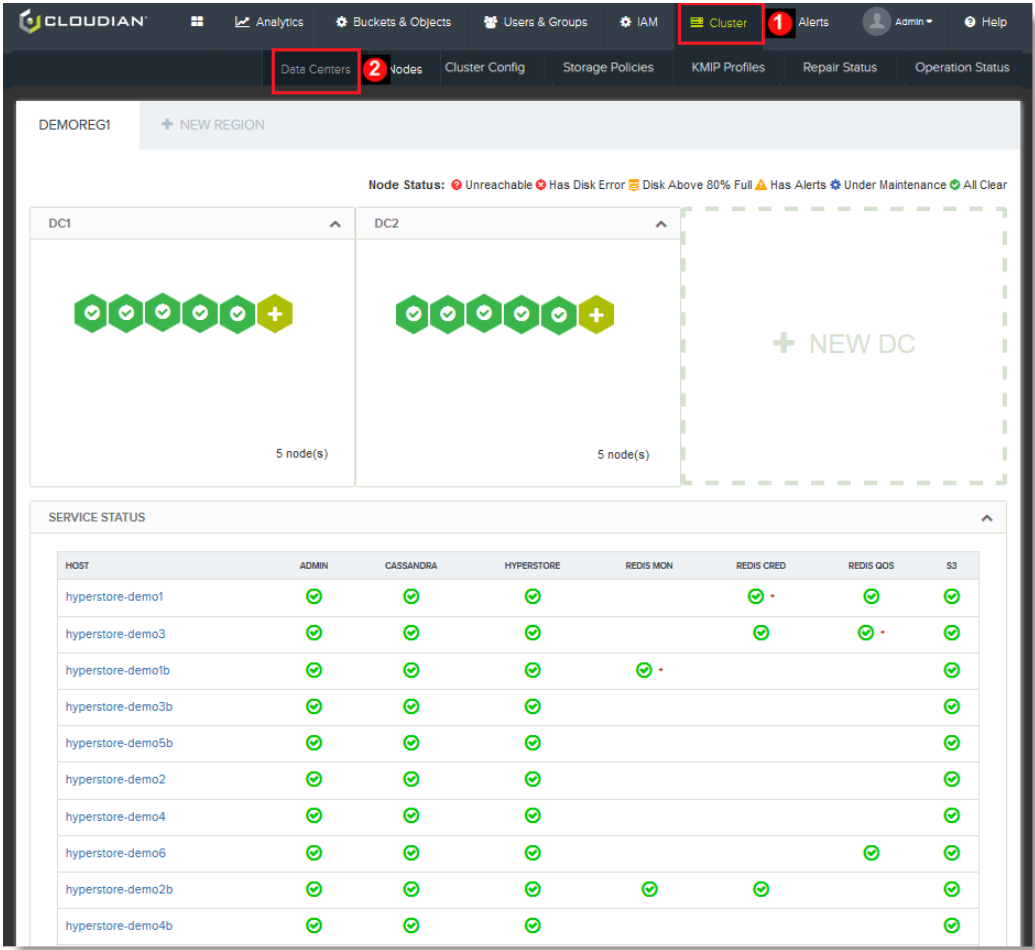


- If a repair is running on a node because you recently initiated a ["replacedisk"](#) operation on that node, Cloudian recommends **waiting until the repair completes** before you add a node to your cluster.
- If you proceed with adding a node at a time when a repair is in progress, and the system automatically stops the repair, do not run the *"-resume"* option on that repair after you've added the node. Adding a node affects the token range distribution within the cluster, so resuming an interrupted repair operation afterward is not supported. If you want to repair a node on which repair was interrupted, wait until after you've completed the rebalance operation (as part of the Adding Nodes procedure), and then run a fresh repair operation on the node for which repair was interrupted.

Note When you initiate the **Add Node** operation in the CMC, the system will also automatically disable the auto-repair feature -- so that no new auto-repairs kick off while you're expanding your cluster -- and then after you've completed the rebalance operations for all new nodes the system will automatically re-enable auto-repair.

- b. In the **Data Centers** page, make sure that all the existing nodes and services are up and running in the service region.

More detail:



The **Add Node** function will not let you add nodes if any existing nodes or services in the region are down.

7.1.3. Adding Nodes

1. In the CMC's **Data Centers** page, in the display for the data center and rack in which you want to add a node or nodes, click the light green cube icon that has a plus sign on it.



More detail:

Node Status: ⬮ Unreachable ⬮ Has Disk Error ⬮ Stop Write ⬮ Disk Above 80% Full ⬮ Has Alerts ⬮ Under Maintenance ⬮ Add Node in progress ⬮ All Clear

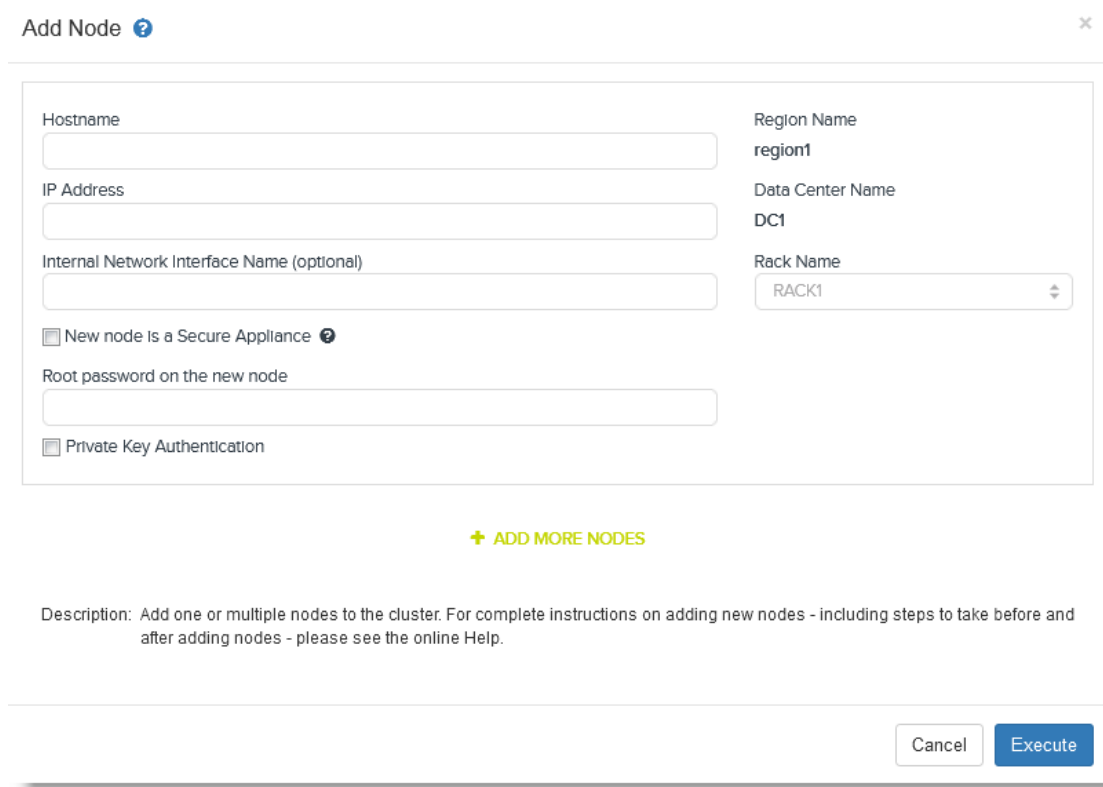
HOST	ADMIN	CASSANDRA	HYPERSTORE	REDIS MON	REDIS CRED	REDIS QOS	S3
hyperstore-demo1	✓	✓	✓		✓	✓	✓
hyperstore-demo3	✓	✓	✓		✓	✓	✓
hyperstore-demo1b	✓	✓	✓	✓			✓
hyperstore-demo3b	✓	✓	✓				✓
hyperstore-demo5b	✓	✓	✓				✓
hyperstore-demo2	✓	✓	✓				✓
hyperstore-demo4	✓	✓	✓				✓
hyperstore-demo6	✓	✓	✓			✓	✓
hyperstore-demo2b	✓	✓	✓	✓	✓		✓
hyperstore-demo4b	✓	✓	✓				✓

Clicking the light green cube opens the **Add Node** interface.

Note If the data center has multiple racks for HyperStore -- which is possible only if your original HyperStore version installed was earlier than version 7.2 -- and if you want the new node(s) to be assigned to a new rack rather than one of the existing racks, just click on the light green cube icon for any rack in the display for the correct data center. If (and only if) the data center has multiple racks for your existing HyperStore nodes, you will have an opportunity to specify a new rack name in the next step below.

- In the **Add Node** dialog that displays, complete the node information fields for the new node(s). After completing all the fields for one new node, click **Add More Nodes** and complete the fields for another new node; and repeat this process until you've completed the fields for all the new nodes that you want to add to the data center.

More detail:



Add Node ?

Hostname

IP Address

Internal Network Interface Name (optional)

☐ New node is a Secure Appliance ?

Root password on the new node

☐ Private Key Authentication

Region Name
region1

Data Center Name
DC1

Rack Name
RACK1

+ ADD MORE NODES

Description: Add one or multiple nodes to the cluster. For complete instructions on adding new nodes - including steps to take before and after adding nodes - please see the online Help.

Cancel Execute

Hostname (required)

Hostname of the new node. This must be just a hostname, not an FQDN. Do not use the same hostname for more than one node in your entire HyperStore system. Each node must have a unique hostname, even if the nodes are in different domains.

IP Address (required)

Service network IP (v4) address that the hostname resolves to. Do not use IPv6.

Internal Network Interface Name (optional)

If the new node will use a different dedicated interface for internal cluster traffic than other nodes in your cluster use — for example if the new node uses "eth2" for internal traffic while other nodes in your cluster are using "eth1" for internal traffic — enter the interface name in this field. If the new node will use the same internal network interface as your existing nodes you can leave this field empty.

Rack Name (required)

The behavior of this field depends on your particular HyperStore system and existing set-up:

- If your original HyperStore system is version 7.2 or later, or if your original HyperStore system was earlier than 7.2 and you have only been using one rack name for your existing nodes in the data center, this field value is fixed to the rack name that you have been using for your existing nodes. You cannot edit this field.
- If your original HyperStore system was earlier than 7.2 and you have been using multiple rack names for your existing nodes in the data center, you can select any of those rack names from the drop-down list or create a new rack name.

IMPORTANT ! If your system is configured with multiple rack names and you are adding nodes, make sure to position the new nodes so that when you are all done adding nodes, each rack has the same number of nodes (for example, three nodes on each rack). Having an imbalance in the number of nodes per rack will result in data load imbalance, such that on racks with fewer nodes there will be more stored data per node than on racks with more nodes.

Authentication fields (based on new node type)

During the **Add Node** operation, the HyperStore installer on your Config Controller node needs to securely connect to the new node. The authentication options for doing so depend on the new node type:

- **If the new node is a "Secure Appliance"** (a HyperStore Appliance for which the HyperStore Shell [HSH] was enabled and the root password disabled at the factory), select the "New node is a Secure Appliance" checkbox. Then enter the `sa_admin` user's password for the new node.

☒ New node is a Secure Appliance ?

`sa_admin` user password on the new node

Note Before the new node is added to your HyperStore cluster, the `sa_admin` user's password on the new node may be different than the `sa_admin` user's password in your existing cluster. If so, after the new node is added to the cluster, the `sa_admin` user's password on the new node will be automatically changed to match the `sa_admin` user's password in the cluster.

- **If the new node is not a "Secure Appliance"** -- that is, if the new node is a standard Appliance or a software-only node on commodity hardware -- then you can use either one of these authentication methods (use one method or the other -- not both):
 - Enter the `root` user's password for the new node.
 - OR
 - Select the "Private Key Authentication" checkbox. In this case the installer will use the same private key as was used to install the existing cluster. Distribution of the corresponding public key to the new node depends on how you handled SSH key set-up during installation of the existing HyperStore cluster:
 - If during installation of the cluster you let the installer generate an SSH key pair for you, or you used your own existing SSH key pair and you copied both the private and the public key into the installation staging directory on the Config Controller, then distribution of the public key to the new node will be taken care of automatically by the installer.
 - If during installation of the cluster you used your own existing SSH key pair and you copied only the private key into the installation directory -- and you

copied the public key to the target installation nodes manually -- **then you must also copy the public key to the new node manually**, before executing the **Add Node** operation.

3. Click **Execute**. The system will first run a pre-check to confirm that each of your specified new nodes can be reached through the network. The system then performs a **simulation that shows how token ranges and data will be distributed across your cluster** if you proceed with adding the new nodes. The simulation results are displayed in an overlay over the **Add Node** dialog, including summary assessments of whether the data associated with each of your existing storage policies will be well-balanced across the cluster if you add the new nodes. Scroll through the simulation results and carefully review the summary assessments. Typically the assessments will indicate a good data distribution balance, and you can then proceed with adding the new nodes by clicking **Install**.

IMPORTANT ! If the assessment projects an **imbalance** in storage capacity utilization if you proceed with adding the new nodes using the system's default method of updating the token range map, the Add Node operation will **abort**. If this occurs contact your Cloudian Account Team or Partner for assistance in implementing your cluster expansion in a way that provides better capacity utilization balance.

4. When you click **Install** in Step 3 the system then adds the new node(s) to your cluster, one node at a time. As part of adding a new node to the cluster, the system automatically streams to the new node its proper share of system and object metadata. Depending on how many nodes you are adding and how much data is in your system, this process can take anywhere from several hours up to several days or more. When this streaming process completes successfully for all the new nodes, for each new node an orange node icon with a gear inside of it should display in the **Data Centers** page. 

More detail:

The **Add Node** operation entails verifying that the new host meets HyperStore requirements, installing software, updating system configuration, starting services, joining the new node into the cluster, and streaming system and object metadata to the new node (in the Cassandra database).

There are two CMC locations where you can monitor the **Add Node** operation progress:

- The **Data Centers** page. As each node is added to the cluster a new node icon appears, with color-coding to indicate the node's status. You can hold your cursor over the node icon for a text description of the status.



Grey node with gear -- The node has been added to the cluster and Cassandra repair is in progress (which streams system and object metadata to the new node).



Orange node with gear -- Cassandra repair has completed successfully, and the node is ready for you to run a "rebalance" operation to stream object data to it (as described later in this procedure). **If all the new nodes show this status** you can proceed to Steps 5 and 6.



Red node with gear -- Cassandra repair has failed for the node.

- The **Operation Status** page. In this page you will see a one-line status summary for the **addNode** operation (even if multiple nodes are being added there is just one summary status line for the operation as a whole). If you click **View** to the right of the summary status line you can display detailed progress information.

Operation Status				
OPERATION NAME	TARGET	STATUS	START TIME	LAST UPDATE
addNode	region1:DC1	● Inprogress	Dec-07-2020 17:21	Dec-07-2020 17:21
0%				
The operation is initialized and the session will be managed by the node: cld163 addNode is starting				

Handling Failures:

- *Failure in adding the node to the cluster:* If in the **Data Centers** page the grey node icon (indicating that the node has been added and Cassandra repair is underway) never appears for one or more nodes, go the **Operation Status** page and view the status detail. Try to resolve the problem reported in the status detail -- in some cases a reboot of the new node may suffice. Then go back to the CMC's **Data Centers** page and again initiate the Add Node operation.  This time

enter the node information **only for the node(s) that failed to be added** to the cluster, then Execute the operation. Be sure to include only the node(s) that failed to be added (do not include the successfully added nodes, and do not add entirely new nodes that you haven't tried to add yet -- this will cause the operation to fail.) Note that you will not be shown a data distribution simulation again on this second attempt at adding the nodes.

Note Do not change the disk configuration of the new nodes before trying again to add the new nodes (since this could result in sub-optimal token distribution, and consequently a sub-optimal data distribution). If for some reason you must change the disk configuration before trying again to add the new nodes, contact Clodian Support.

- *Failure in Cassandra repair* (as indicated by red node icon with gear): If this status occurs, first check the **Data Center** page's Service Status section to make sure that Cassandra is up and running on all nodes (if it's down on any node, go to the **Node Status** page for that node and start Cassandra). After making sure Cassandra is up on all nodes go to the **Node Advanced** page, and from the Maintenance command type menu run *repaircassandra* on the new node. Periodically check the progress. In the **Data Centers** page the new node's icon should turn orange when the repair completes successfully.

If these corrective measures fail, contact Clodian Support.

5. After the **Add Node** operation has completed successfully for all of the new nodes -- as indicated by there being an orange node icon for each new node in the **Data Centers** page -- **update your DNS and/or load balancer** configurations to include the new nodes, so that the new nodes can participate in servicing S3 user request traffic. For more information see ["DNS Set-Up" \(page 53\)](#) and ["Load Balancing" \(page 56\)](#).

6. In the **Node Advanced** page, from the Maintenance command type group, **execute *hsstool rebalance* on each new node**. Initiate these rebalance operations from the CMC **Node Advanced** page, not from the command line. When launching *rebalance* on each new node, use the **cleanupfile** option. Rebalance is a long-running background operation that you can run concurrently on multiple new nodes that you've added (but check the details below for important limitations on concurrency). When all new nodes have completed successful rebalancing, for each node a green check-marked cube icon will display in the **Data Centers** page.



More detail:

The screenshot shows the Cloudbian management interface. The top navigation bar includes 'Cluster' (highlighted with a red box and a red '1'), 'Alerts', 'Admin', and 'Help'. Below this, the 'Nodes' tab is selected (highlighted with a red box and a red '2'). The 'Advanced' sub-tab is also selected (highlighted with a red box and a red '3'). In the 'Advanced' section, the 'Command Type' dropdown is set to 'Maintenance' (circled in red). The 'hsstool Command' dropdown is set to 'rebalance' (circled in red). The 'Options' section has the 'cleanupfile' checkbox checked (circled in red), with 'stop', 'resume', and 'newrun' options unchecked. The 'Target Node' dropdown is set to 'hyperstore-demo6'. A description at the bottom reads: 'Rebalance data after adding new nodes to a data center'. An 'EXECUTE' button is located at the bottom right.

The rebalance operation populates the new node(s) with their appropriate share of S3 object data. The rebalance is a background operation that may take up to several days or more for each new node to complete, depending on factors such as data volume and network bandwidth. When rebalance is performed with the **cleanupfile** option, as soon as a replica or fragment is successfully copied to the new node, it is deleted from the older node on which it no longer belongs -- thereby freeing up storage space on the older nodes as the rebalance operation progresses. For more information see "[hsstool rebalance](#)" (page 904).

You can run *rebalance* on multiple new nodes concurrently. However, **do not run *rebalance* concurrently on a number of new nodes that exceeds one-third the number of your already existing nodes** in that same data center. For example, if you have 12 existing nodes in a DC and you are adding 6 new nodes to that DC, do not run *rebalance* on more than 4 of the new nodes concurrently (one-third of 12). In this scenario you would run *rebalance* on 4 of the new nodes concurrently, and then when the *rebalance* operation completes on at least 2 of those nodes, you can start *rebalance* on the other 2 new nodes. In this way *rebalance* is never running on more than 4 nodes at a time (never running on more than one-third the number of the pre-existing nodes in the DC).

Note During in-progress rebalance operations the affected data remains readable by S3 client applications. Meanwhile the new nodes are immediately available to support writes of new data, even before any rebalancing occurs.

Note When you have rebalance running on a new node or multiple new nodes, you cannot add any additional nodes to the service region (using the CMC's **Add Node** feature) until rebalance has completed successfully on all of the current new nodes.

Use the **Data Centers** page to periodically **check the progress** of the rebalance operation on each of the new nodes. The icon for each of the new nodes will be color-coded, and you can hold your cursor over the icon for a text description of the status.



Grey node with gear -- The rebalance operation is in progress. If you have added multiple nodes and have started rebalance operations on the nodes, each node's status icon will remain in this status until rebalance completes on all of the nodes in the batch.

Red node with gear -- The rebalance operation has failed for one or more token ranges. If this status displays, first go to the **Data Centers** page and make sure that all the existing nodes and services are up and running in the service region. Start any nodes or services that are down. Then go to the **Operation Status** page to check the status information for the rebalance operation, and click **View** for operation status details:



- If the rebalance operation status shows COMPLETED but also shows that some stream jobs failed (the "failed count" metric is non-zero), then go to the **Nodes Advanced** page and run rebalance on the node again with the **resume** option (select the resume checkbox when you run rebalance on the node). This will try the rebalance again just for the failed token range(s). You do not need to specify the **cleanupfile** option since resuming the rebalance will use whichever options you specified for the original rebalance run.
- If the rebalance operation status shows FAILED, then go to the **Nodes Advanced** page and run rebalance on the node again with the **newrun** option (select the newrun checkbox when you run rebalance on the node). Also select the **cleanupfile** option if you want data removed from the existing nodes after it's successfully copied to the new node. By using the newrun option you initiate a fresh rebalance operation on the node. Note that this fresh rebalance run will not re-stream to the new node any data that had already been streamed there by the prior rebalance operation (there will not be a duplication of data on the new node). Instead the new rebalance run will detect that this data is already in place on the new node and will not re-stream it.

If after these steps the rebalance operation still does not succeed, contact Cloudbian Support.



Green node with check mark -- The rebalance operation has completed successfully.

If for some reason you need to stop -- and then later resume -- a rebalance operation on a node, you can do so as described in "[hsstool rebalance](#)" (page 904).

This completes the procedure for adding nodes to your cluster.

Note: If you removed a dead node prior to performing the **Adding Nodes procedure** and deferred the node repair operations that are necessary after you remove a dead node, perform those repairs now.

More detail:

- a. From the **Node Advanced** page, run [hsstool repair](#) on each node in the service region except for the new node(s), just one node at a time. When repairing each node, use the **allkeyspaces** option and also the **-pr** option. Leave the **-l** and **-m** options selected, as they are by default. Use the **Operation Status** page to track the progress of each repair. After repair of a node is complete, repair another node -- until all nodes except for the new node(s) have been successfully repaired.
- b. If you have erasure coded object data in your system, from the **Node Advanced** page run [hsstool repairec](#) on one node in each HyperStore data center in the region. It doesn't matter which node you run it on, as long as you do it for one node in each DC in the region. Use the **Operation Status** page to track repair progress.

7.2. Adding a Data Center

This procedure is for adding a [new data center](#) to an existing HyperStore service region. During this procedure you will:

- Prep the nodes in the new data center
- Verify that your existing cluster is in a proper condition to successfully add a data center
- Add the new data center's nodes to the cluster
- Take initial steps to start utilizing the new data center

Note Each node that you add must have the same networking configuration as the existing nodes in your HyperStore system -- for example, if the existing nodes each have two NICs, one for front-end network and one for back-end network, then the new nodes must have the same configuration.

7.2.1. Special Requirements if an Existing Node is Down or Unreachable

The CMC's **Add DC** function **will not work if an existing node in your cluster is down or unreachable**. If you have more nodes in your cluster than are required by your configured storage policies and one of those nodes is permanently down, follow the procedure for ["Removing a Node"](#) (page 603) to remove the dead node from the cluster. After you complete that procedure you can then follow this procedure for Adding a Data Center. If your current cluster has **only the minimum number of nodes required by your storage policies and one of those nodes is dead**, contact Cloudian Support for guidance.

7.2.2. Preparing to Add a Data Center

Before you add a new data center to a region, prepare by taking the actions below.

1. The HyperStore nodes in each data center will need to be able to communicate with the HyperStore nodes in the other data center(s). This includes HyperStore services that listen on the internal interface. Therefore, if you haven't already done so you must **configure your inter-DC networking so that the DCs' internal networks are connected to each other** (for example, by using a VPN).
2. Make sure the new host(s) **meet requirements** for:
 - Hardware specs and operating system: See ["Host Hardware and Operating System"](#) (page 46).
 - Open listening ports: See ["HyperStore Listening Ports"](#) (page 57).

3. Make sure you have the **information you will need** to complete this procedure: the new data center's name, and each new node's hostname, IPv4 address, internal interface name (optional), and root password (or `sa_admin` user password for a "Secure Appliance" with its HyperStore Shell enabled and root password disabled at the factory). For data center names only alphanumeric characters and dashes are supported.
4. **Start the new host(s)**, if not already running.
5. From your Config Controller node use the `system_setup.sh` tool's **Prep New Node to Add to Cluster** function to complete network interface configuration, time zone set-up, prerequisites installation, and disk formatting for each new node.

More detail:

- a. In your existing cluster, on your Config Controller node change into the installation staging directory (`/opt/cloudian-staging/8.2`) and then launch the `system_setup.sh` tool.

```
# ./system_setup.sh
```

If you are using the HyperStore Shell

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Config Controller node you can launch the system setup tool with this command:

```
$ hspkg setup
```

Once launched, the setup tool's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

- b. In the tool's main menu select **"8"** for **Prep New Node to Add to Cluster**. When prompted provide the IP address of a new node, and then the password for logging into the node. A menu of node preparation tasks will then display.
- c. Use the node preparation task menu to prepare the node:
 - Complete the configuration of network interfaces for the node, if you haven't already.
 - Set the timezone for the node.
 - Install and configure HyperStore prerequisites on the node.
 - Set up data disks on the node with `ext4` file systems, if you haven't already. Make sure to format and mount **all** available data disks on the node. The tool also supports setting the node's Metadata DB (Cassandra) drives into a RAID1 configuration.
 - After completing the setup tasks for the node choose the "Return to Master Node" option, which returns you to the tool's main menu.
- d. Repeat steps "b" and "c" above for each new node that you're adding. When you're done, exit the `system_setup.sh` tool.

IMPORTANT ! If the new node(s) are not HyperStore Appliances and **if you do not use `system_setup.sh` to format the data disks on the new node(s)**, then in the installation staging directory on your Config Controller node you must for each new node create a text file named `<hostname>_fslist.txt` that specifies the new node's data mount points, in this format:

```
<devicename> <mountpoint>
<devicename> <mountpoint>
etc...
```

- In the CMC's **Data Centers** page, make sure that all the existing nodes and services are up and running in the region in which you are adding a data center. The **Add DC** function will not let you add nodes if any existing nodes or services in the region are down.

Node Status: ● Unreachable ● Has Disk Error ■ Disk Above 80% Full ▲ Has Alerts ● Under Maintenance ● All Clear

HOST	ADMIN	CASSANDRA	HYPERSTORE	REDIS MON	REDIS CRED	REDIS QOS	S3
hyperstore-demo1	✓	✓	✓		✓	✓	✓
hyperstore-demo3	✓	✓	✓		✓	✓	✓
hyperstore-demo1b	✓	✓	✓	✓			✓
hyperstore-demo3b	✓	✓	✓				✓
hyperstore-demo5b	✓	✓	✓				✓
hyperstore-demo2	✓	✓	✓				✓
hyperstore-demo4	✓	✓	✓				✓
hyperstore-demo6	✓	✓	✓			✓	✓
hyperstore-demo2b	✓	✓	✓	✓	✓		✓
hyperstore-demo4b	✓	✓	✓				✓

Note When you initiate the **Add DC** function (as described below) the system automatically disables Cassandra auto-repair throughout the service region, and then when the **Add DC** operation has completed the system automatically re-enables Cassandra auto-repair throughout the service region.

7.2.3. Adding a Data Center

- In the CMC's **Data Centers** page, with the correct service region tab selected, click the large box that says **+NEW DC**.

Node Status: ⬮ Unreachable ⬮ Has Disk Error ⬮ Stop Write ⬮ Disk Above 80% Full ⬮ Has Alerts
⬮ Under Maintenance ⬮ Add Node in progress ⬮ All Clear

DC1 **DC2**

5 node(s) 5 node(s)

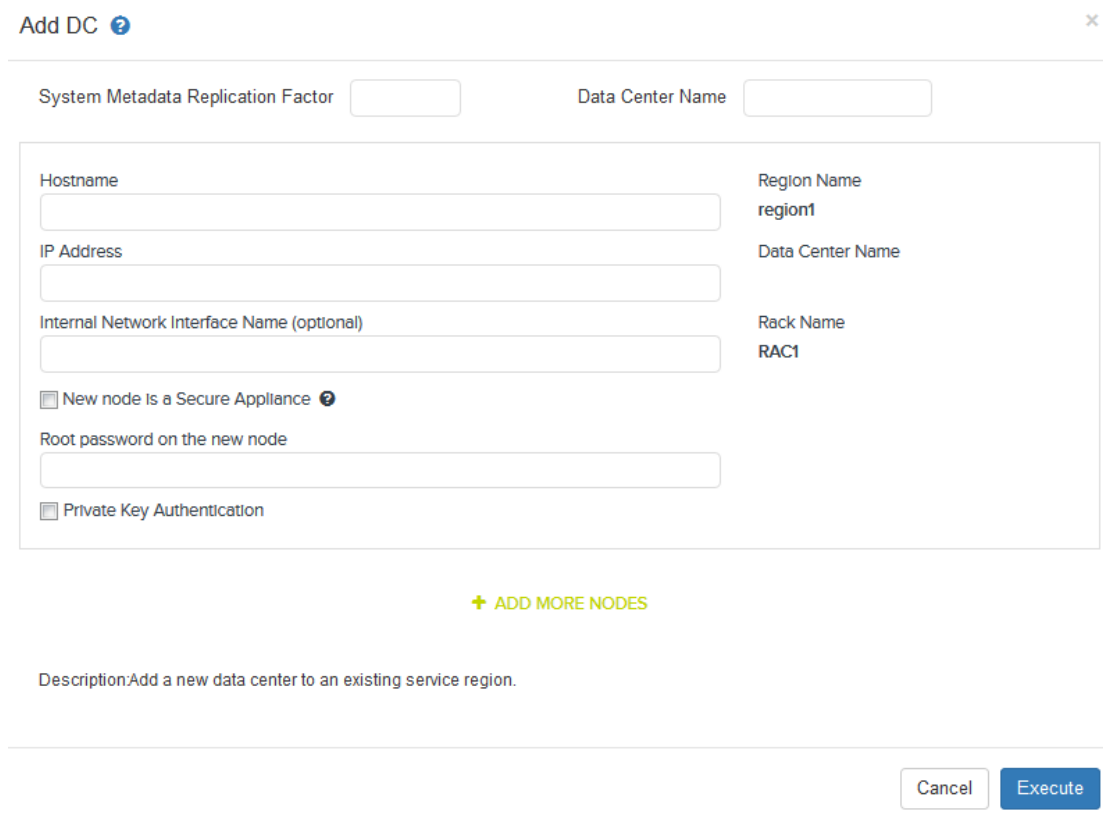
+ NEW DC

SERVICE STATUS

HOST	ADMIN	CASSANDRA	HYPERSTORE	REDIS MON	REDIS CRED	REDIS QOS	S3
hyperstore-demo1	✓	✓	✓		✓	✓	✓
hyperstore-demo3	✓	✓	✓		✓	✓	✓
hyperstore-demo1b	✓	✓	✓	✓			✓
hyperstore-demo3b	✓	✓	✓				✓
hyperstore-demo5b	✓	✓	✓				✓
hyperstore-demo2	✓	✓	✓				✓
hyperstore-demo4	✓	✓	✓				✓
hyperstore-demo6	✓	✓	✓			✓	✓
hyperstore-demo2b	✓	✓	✓	✓	✓		✓
hyperstore-demo4b	✓	✓	✓				✓

- In the **Add DC** interface that displays, complete the top two fields for the new data center as a whole, then complete the remaining fields for each node in the data center, clicking **Add More Nodes** to display fields for additional nodes as needed.

More detail:



Add DC ?

System Metadata Replication Factor Data Center Name

Hostname Region Name

IP Address Data Center Name

Internal Network Interface Name (optional) Rack Name

☐ New node is a Secure Appliance ?

Root password on the new node

☐ Private Key Authentication

+ ADD MORE NODES

Description: Add a new data center to an existing service region.

Cancel Execute

System Metadata Replication Factor (required)

In this field indicate how many replicas of system metadata (such as usage reporting data, user account information, and system monitoring data) you want to store in the new DC. For each metadata item, each replica will be stored on a different node, to protect this metadata against loss or corruption. It's recommended that you set this replication factor to 3 if you are going to have three or more nodes in the new DC. If there will be only two nodes in the new DC, set this to 2; if there will be only one node, set this to 1. You cannot set a metadata replication factor higher than the number of nodes in the new DC.

Data Center Name (required)

Name of the new data center. Maximum 256 characters. Only ASCII alphanumeric characters and dashes are allowed.

Hostname (required; must be unique within system)

Hostname of the new node.

Note

- * This must be just a hostname, not an FQDN.
- * Do not use the same hostname for more than one node in your entire HyperStore system. Each node must have a unique hostname within your entire HyperStore system, even in the case of nodes that are in different domains.

IP Address (required)

Service network IP (v4) address that the hostname resolves to. Do not use IPv6.

Internal Network Interface Name (optional)

If the new node will use a different dedicated interface for internal cluster traffic than other nodes in your cluster use — for example if the new node uses "eth2" for internal traffic while other nodes in your cluster are using "eth1" for internal traffic — enter the interface name in this field. If the new node will use the same internal network interface as your existing nodes you can leave this field empty.

Rack Name (fixed value "RAC1")

The "Rack Name" value is automatically fixed to "RAC1" for all nodes in the new data center. You cannot edit this value.

Note This is an internal value used by HyperStore. It does not need to correspond to any actual rack name in your data center.

Authentication fields (based on new node type)

During the **Add DC** operation, the HyperStore installer on your Config Controller node needs to securely connect to each new node. The authentication options for doing so depend on the new node type:

- **If the new node is a "Secure Appliance"** (a HyperStore Appliance for which the HyperStore Shell [HSH] was enabled and the root password disabled at the factory), select the "New node is a Secure Appliance" checkbox. Then enter the `sa_admin` user's password for the new node.

☒ New node is a Secure Appliance ?

sa_admin user password on the new node

Note Before the new node is added to your HyperStore cluster, the `sa_admin` user's password on the new node may be different than the `sa_admin` user's password in your existing cluster. If so, after the new node is added to the cluster, the `sa_admin` user's password on the new node will be automatically changed to match the `sa_admin` user's password in the cluster.

- **If the new node is not a "Secure Appliance"** -- that is, if the new node is a standard Appliance or a software-only node on commodity hardware -- then you can use either one of these authentication methods (use one method or the other -- not both):
 - Enter the `root` user's password for the new node.
 - OR
 - Select the "Private Key Authentication" checkbox. In this case the installer will use the same private key as was used to install the existing cluster. Distribution of the corresponding public key to the new node depends on how you handled SSH key set-up during installation of the existing HyperStore cluster:
 - If during installation of the cluster you let the installer generate an SSH key pair for

you, or you used your own existing SSH key pair and you copied both the private and the public key into the installation staging directory on the Config Controller, then distribution of the public key to the new node will be taken care of automatically by the installer.

- If during installation of the cluster you used your own existing SSH key pair and you copied only the private key into the installation directory -- and you copied the public key to the target installation nodes manually -- **then you must also copy the public key to the new node manually**, before executing the **Add DC** operation.
3. Click **Execute**. This initiates a background operation that will take anywhere from several minutes to several hours to complete depending on your environment. When it completes successfully the **Data Centers** page will display an additional block representing the newly added DC, with a green, check-marked cube icon for each of the new DC's nodes.



More detail:

The **Add DC** operation entails verifying that the new hosts meet HyperStore requirements, installing software, updating system configuration, starting services, joining the new nodes into the cluster, and streaming system metadata to the new nodes (in Cassandra).

You can use the **Operation Status** page to monitor progress of the operation.

OPERATION NAME	TARGET	STATUS	PROGRESS	START TIME	LAST UPDATE
addDC	DC3	in progress		Apr-30-2018 11:25	Apr-30-2018 11:26

For status detail click **View** to the right of the summary status line.

When the operation is complete, to see the new nodes in the **Data Centers** page you may need to refresh the page in your browser.

If you hold your cursor over each cube in the new data center the node host names will display.

Note If the **Operation Status** page indicates that the **Add DC** operation has failed, click "View" for detail. Then for more information to support troubleshooting efforts, *grep* for "ERROR" level messages in the *cloudbian-installation.log* file under the installation staging directory on your Config Controller node.

4. **Update your DNS and load balancing configurations** to include the new data center and its nodes, if you have not already done so. For more information see ["DNS Set-Up"](#) (page 53) and ["Load Balancing"](#) (page 56).
5. Go to the CMC's [Storage Policies](#) page and **create one or more storage policies that utilize the new DC**. Until you create storage policies that use the new DC and users subsequently create buckets that

use those storage policies, no S3 object data will be stored in the new DC.

Note Because your previously existing storage policies do not include the new DC, none of the data already stored in your system in association with those storage policies will be migrated into the new DC. Accordingly, there is no need for you to run a *rebalance* operation after adding a new DC.

This completes the procedure for adding a data center to your cluster.

NOTE: If you removed a dead node prior to performing the Adding a Data Center procedure and deferred the node repair operations that are necessary after you remove a dead node, perform those repairs now.

More detail:

- a. From the **Node Advanced** page, run [hsstool repair](#) on each node in the service region except for the new nodes, just one node at a time. When repairing each node, use the **allkeyspaces** option and also the **-pr** option. Leave the **-l** and **-m** options selected, as they are by default. Use the **Operation Status** page to track the progress of each repair. After repair of a node is complete, repair another node -- until all nodes except for the nodes in the new data center have been successfully repaired.
- b. If you have erasure coded object data in your system, from the **Node Advanced** page run [hsstool repair](#) on one node in each HyperStore data center in the region except for the new data center. It doesn't matter which node you run it on, as long as you do it for one node in each DC in the region, except for the new DC. Use the **Operation Status** page to track repair progress.

7.3. Adding a Region

This procedure is for adding a new [service region](#) to your HyperStore system. During this procedure you will:

- Prep the nodes in the new service region
- Add the new region's nodes to the system
- Take initial steps to start utilizing the new region

Note Each node that you add must have the same networking configuration as the existing nodes in your HyperStore system -- for example, if the existing nodes each have two NICs, one for front-end network and one for back-end network, then the new nodes must have the same configuration.

7.3.1. Preparing to Add a Region

Before you add a new region to your system, prepare by taking the actions below.

1. The HyperStore nodes in each region will need to be able to communicate with the HyperStore nodes in the other region(s). This includes HyperStore services that listen on the internal interface. Therefore, if you haven't already done so you must **configure your networking so that the internal networks of all of your data centers in all of your regions are connected to each other** (for example, by using a VPN).

2. Make sure the new host(s) **meet requirements** for:
 - Hardware specs and operating system: See ["Host Hardware and Operating System"](#) (page 46).
 - Open listening ports: See ["HyperStore Listening Ports"](#) (page 57).
3. Make sure you have the **information you will need** to complete this procedure: the name of the new region, the name(s) of the data center(s) that will comprise the region, and each new node's hostname, IPv4 address, internal interface name (optional), rack name, and root password (or `sa_admin` user password for a "Secure Appliance" with its HyperStore Shell enabled and root password disabled at the factory). For region, DC, and rack names only alphanumeric characters and dashes are supported.
4. **Start the new host(s)**, if not already running.
5. From your Config Controller node use the `system_setup.sh` tool's **Prep New Node to Add to Cluster** function to complete network interface configuration, time zone set-up, prerequisites installation, and disk formatting for each new node.

More detail:

- a. In your existing cluster, on your Config Controller node change into the installation staging directory (`/opt/cloudian-staging/8.2`) and then launch the `system_setup.sh` tool.

```
# ./system_setup.sh
```

If you are using the HyperStore Shell

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Config Controller node you can launch the system setup tool with this command:

```
$ hspkg setup
```

Once launched, the setup tool's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

- b. In the tool's main menu select **"8" for Prep New Node to Add to Cluster**. When prompted provide the IP address of a new node, and then the password for logging into the node. A menu of node preparation tasks will then display.
- c. Use the node preparation task menu to prepare the node:
 - Complete the configuration of network interfaces for the node, if you haven't already.
 - Set the timezone for the node.
 - Install and configure HyperStore prerequisites on the node.
 - Set up data disks on the node with `ext4` file systems, if you haven't already. Make sure to format and mount **all** available data disks on the node. The tool also supports setting the node's Metadata DB (Cassandra) drives into a RAID1 configuration.
 - After completing the setup tasks for the node choose the "Return to Master Node" option, which returns you to the tool's main menu.
- d. Repeat steps "b" and "c" above for each new node that you're adding. When you're done, exit the `system_setup.sh` tool.

IMPORTANT ! If the new node(s) are not HyperStore Appliances and **if you do not use `system_setup.sh` to format the data disks on the new node(s)**, then in the installation staging directory on your Config Controller node you must for each new node create a text file named

<hostname>_fslst.txt that specifies the new node's data mount points, in this format:

<devicename> <mountpoint>

<devicename> <mountpoint>

etc...

7.3.2. Adding a Region

1. In the CMC's **Data Centers** page, click the tab that says **+NEW REGION**.

Node Status: ⛔ Unreachable ⛔ Has Disk Error ⛔ Stop Write ⚠️ Disk Above 80% Full ⚠️ Has Alerts
⚙️ Under Maintenance ⚙️ Add Node in progress ✅ All Clear

HOST	ADMIN	CASSANDRA	HYPERSTORE	REDIS MON	REDIS CRED	REDIS QOS	S3
hyperstore-demo1	✅	✅	✅		✅	✅	✅
hyperstore-demo3	✅	✅	✅		✅	✅	✅
hyperstore-demo1b	✅	✅	✅	✅			✅
hyperstore-demo3b	✅	✅	✅				✅
hyperstore-demo5b	✅	✅	✅				✅
hyperstore-demo2	✅	✅	✅				✅
hyperstore-demo4	✅	✅	✅				✅
hyperstore-demo6	✅	✅	✅			✅	✅
hyperstore-demo2b	✅	✅	✅	✅	✅		✅
hyperstore-demo4b	✅	✅	✅				✅

2. In the **Add Region** interface that displays, enter the name of the new region then complete the fields for each node in the data center, clicking **Add More Nodes** to display fields for additional nodes as needed.

More detail:

Add Region ?

Region Name

Hostname

IP Address

Internal Network Interface Name (optional)

☐ New node is a Secure Appliance ?

Root password on the new node

☐ Private Key Authentication

Region Name

Data Center Name

Rack Name **RAC1**

+ ADD MORE NODES

Description: Add a new service region to an existing system.

Region Name (required)

Name of the new region. Maximum 52 characters. Only ASCII alphanumeric characters and dashes are allowed. Letters must be lower case.

Hostname (required; must be unique within system)

Hostname of the new node.

Note

- * This must be just a hostname, not an FQDN.
- * Do not use the same hostname for more than one node in your entire HyperStore system. Each node must have a unique hostname within your entire HyperStore system, even in the case of nodes that are in different domains.

IP Address (required)

Service network IP (v4) address that the hostname resolves to. Do not use IPv6.

Data Center Name (required)

Name of the data center in which the new node is located. Maximum 256 characters. Only ASCII alphanumeric characters and dashes are allowed.

Internal Network Interface Name (optional)

If the new node will use a different dedicated interface for internal cluster traffic than other nodes in your cluster use — for example if the new node uses "eth2" for internal traffic while other nodes in your

cluster are using "eth1" for internal traffic — enter the interface name in this field. If the new node will use the same internal network interface as your existing nodes you can leave this field empty.

Rack Name (fixed value "RAC1")

The "Rack Name" value is automatically fixed to "RAC1" for all nodes in the new region. You cannot edit this value.

Note This is an internal value used by HyperStore. It does not need to correspond to any actual rack name in your data center(s).

Authentication fields (based on new node type)

During the **Add Region** operation, the HyperStore installer on your Config Controller node needs to securely connect to each new node. The authentication options for doing so depend on the new node type:

- **If the new node is a "Secure Appliance"** (a HyperStore Appliance for which the HyperStore Shell [HSH] was enabled and the root password disabled at the factory), select the "New node is a Secure Appliance" checkbox. Then enter the `sa_admin` user's password for the new node.

☒ New node is a Secure Appliance ?

sa_admin user password on the new node

Note Before the new node is added to your HyperStore cluster, the `sa_admin` user's password on the new node may be different than the `sa_admin` user's password in your existing cluster. If so, after the new node is added to the cluster, the `sa_admin` user's password on the new node will be automatically changed to match the `sa_admin` user's password in the cluster.

- **If the new node is not a "Secure Appliance"** -- that is, if the new node is a standard Appliance or a software-only node on commodity hardware -- then you can use either one of these authentication methods (use one method or the other -- not both):
 - Enter the `root` user's password for the new node.
 - OR
 - Select the "Private Key Authentication" checkbox. In this case the installer will use the same private key as was used to install the existing cluster. Distribution of the corresponding public key to the new node depends on how you handled SSH key set-up during installation of the existing HyperStore cluster:
 - If during installation of the cluster you let the installer generate an SSH key pair for you, or you used your own existing SSH key pair and you copied both the private and the public key into the installation staging directory on the Config Controller, then distribution of the public key to the new node will be taken care of

automatically by the installer.

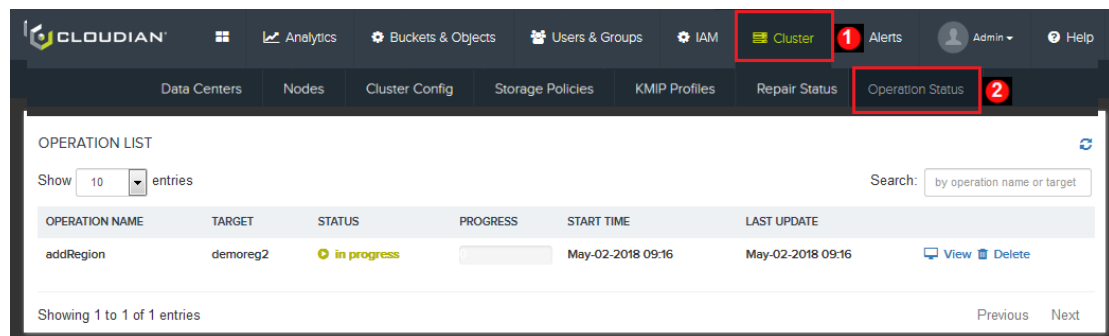
- If during installation of the cluster you used your own existing SSH key pair and you copied only the private key into the installation directory -- and you copied the public key to the target installation nodes manually -- **then you must also copy the public key to the new node manually**, before executing the **Add Region** operation.
3. Click **Execute**. This initiates a background operation that will take anywhere from several minutes up to an hour to complete depending on your environment. When it completes successfully the **Data Centers** page will display an additional tab representing the newly added region, with a green, check-marked cube icon for each of the new region's nodes.



More detail:

The **Add Region** operation entails verifying that the new hosts meet HyperStore requirements, installing software, updating system configuration, starting services, and joining the new nodes into a new Cassandra ring.

Use the **Operation Status** page to monitor progress of the operation.



For status detail click **View** to the right of the summary status line. (The page may at one point indicate that it cannot retrieve status information -- this is due to an S3 Server / Admin Server restart which is an automatic part of the **Add Region** operation. Wait a few minutes then refresh the page.)

When the operation is complete, to see the new nodes in the **Data Centers** page you may need to refresh the page in your browser.

If you hold your cursor over each cube in the new region the node host names will display.

Note If the **Operation Status** page indicates that the **Add Region** operation has failed, click "View" for detail. Then for more information to support troubleshooting efforts, *grep* for "ERROR" level messages in the *cloudian-installation.log* file under the installation staging directory on your Config Controller node.

4. **Update your DNS and load balancing** configurations to include the new service region and its nodes, if you have not already done so. Note that name server configurations in each of your existing region(s) and the new region must have entries for **all** your regions' S3 service endpoints, as well as for the global Admin service and CMC service endpoints. For more information see ["DNS Set-Up" \(page 53\)](#) and ["Load Balancing" \(page 56\)](#).

5. Go to the [Storage Policies](#) page and select the new region. Then **create a storage policy for the new region**. This first storage policy will become the default storage policy for the region. Later if you've created more than one storage policy in the region you can change which policy is the default policy if you wish. Until you create one or more storage policies in the new region and users subsequently create buckets that use those storage policies, no S3 object data will be stored in the new region.
6. Optionally, set Quality of Service (QoS) limits for users' and groups' activity in the new service region. Each service region has its own QoS configuration. By default, in each region no QoS limits are enforced. For more information see ["Set Quality of Service \(QoS\) Limits" \(page 430\)](#). When using the QoS configuration interfaces be sure to select the new service region from the interfaces' drop-down list of regions.

This completes the procedure for adding a service region to your system.

7.4. Removing a Node

This procedure is for **permanently removing a node from your cluster** so that the data storage responsibilities of that node are redistributed to the remaining nodes in the cluster. During this procedure you will:

- Verify that your existing system is in a proper condition to successfully support the removal of a node
- Remove the node
- If the node you removed was "dead" -- the Cassandra Service on the node is down or unreachable when you remove the node -- repair the remaining nodes in your cluster (this is not necessary if the node you removed was live)

Note If you are removing a node after having added a new node to your cluster, you must complete the rebalance operation for the new node before removing a node. For more information on rebalance see ["Adding Nodes" \(page 579\)](#).

IMPORTANT ! Removing a node should be something that you do **only if absolutely necessary**. When you remove a live node from your cluster, during the decommissioning process the data replicas and erasure coded fragments on that node are unavailable to help meet your configured read consistency requirements, when servicing read requests from client applications. Once data has been streamed from the decommissioning node to the other nodes that data is again available to contribute to meeting read consistency requirements. Depending on the data volume and network bandwidth, it may take several days or more until the decommissioning process has completed for all of the node's data.

Note For instructions on **uninstalling an entire HyperStore system**, from all nodes, including deleting all data, see "cloudianInstall.sh Command Line Options" in the *Cloudian HyperStore Installation Guide*.

7.4.1. Preparing to Remove a Node

Take these actions to prepare to remove a node from your cluster:

1. **Make sure that removing the node won't leave your cluster with fewer nodes than your configured storage policies require.**

More detail:

For example if 4+2 erasure coding is being used in your system you cannot reduce your cluster size to fewer than 6 nodes, even temporarily. Or for another example if you have a storage policy that for each object places 3 replicas in DC1 and 3 replicas in DC2, do not reduce the number of nodes in either data center to fewer than 3.

If you're not certain what storage policies currently exist in your system, check the CMC's [Storage Policies](#) page.

The CMC's **Uninstall Node** function checks and enforces this requirement and will not let you remove a node if doing so would leave fewer than the required number of nodes in your cluster. If your cluster is currently at the minimum size required by your storage policies and you want to remove a node:

- If the node you want to remove is **live** you can first add a new node to your cluster by following the complete procedure for **"Adding Nodes"** (page 579) (including rebalancing); and then after that you will be able to remove the node that you want to remove.
 - If the node you want to remove is **dead** contact Cloudbian Support for guidance.
2. **Make sure that you have sufficient available storage capacity on the other nodes in your cluster.**
The data from the removed node will be redistributed to the remaining nodes that are encompassed by the same storage policies as the removed node.

More detail:

Each remaining node must have available storage capacity to take on some of the data from the removed node. You can review your cluster and per-node storage space availability in the CMC's [Capacity Explorer](#) page.

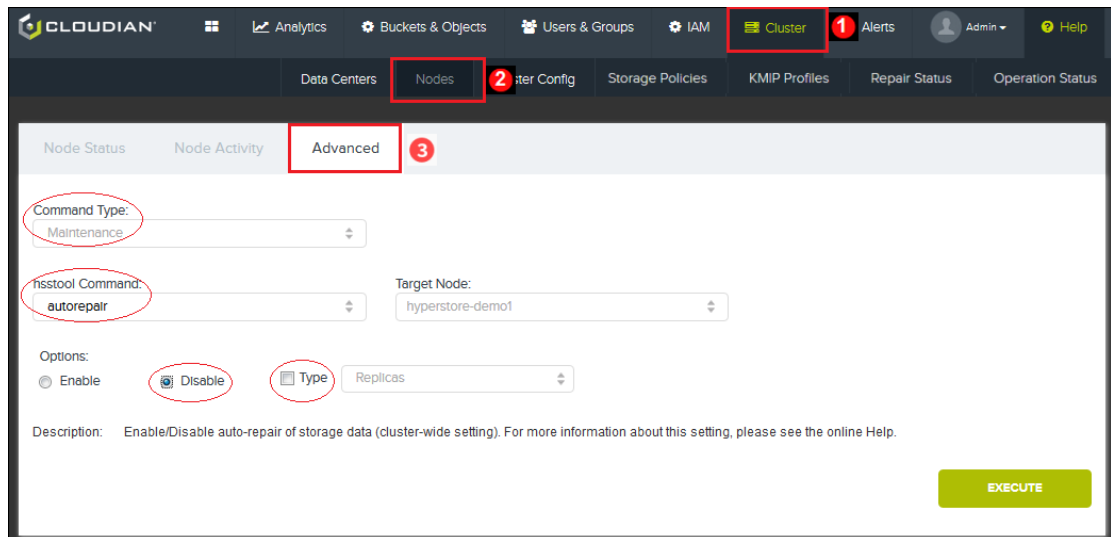
Note If any of the other nodes are in the **"stop-write" condition** -- or if any disks on a node are in stop-write condition -- at a time when you decommission a different node from your system, the decommissioning process overrides the stop-write restriction on the node(s) or disk(s) where it exists in order to stream to **all** nodes and disks in the cluster their share of data from the decommissioned node. Consequently, disks that were nearly full before a decommissioning operation may become completely full during the decommission operation, resulting in stream job failures once the disks can accept no more data.

To avoid this, before removing a node from your cluster make sure there is plenty of space on all the remaining nodes and disks to absorb the data from the node you intend to remove. **If you want to remove a node from your cluster at a time when some disks on the other nodes are nearly full, consult with Cloudbian Support.**

3. In the **Node Advanced** page, from the Maintenance command type group, execute the *autorepair* command with the Disable option to **temporarily disable the automated repair feature** in the service region in which you are removing a node.

More detail:

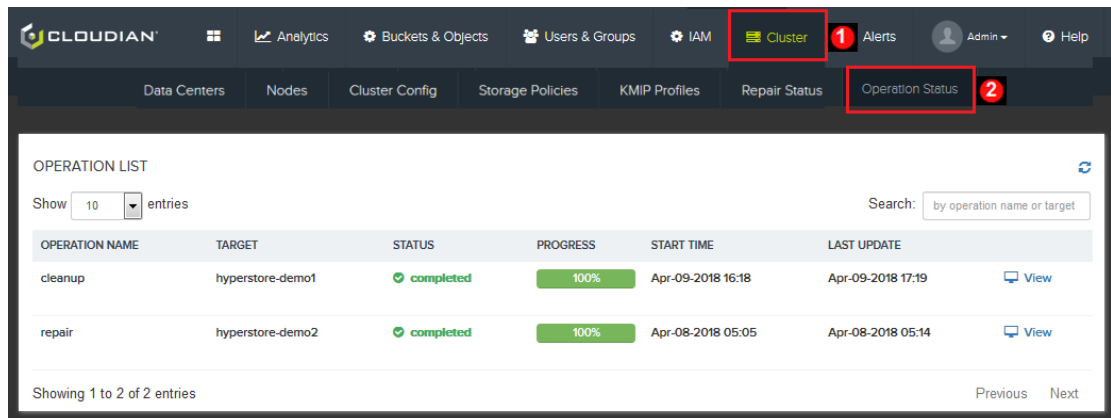
The target node for the command can be any node in the region. Leave the "Type" option unselected so that all automated repair types are disabled. This will prevent any new schedule-based auto-repairs or proactive repairs from launching during the node removal process.



4. In the **Operation Status** page, make sure there are no rebalance operations or other operations currently in progress in the service region.

More detail:

If any operations are in progress, wait until the operations complete before removing a node from your cluster. The CMC's **Uninstall Node** function will not let you remove a node if a major operation such as rebalance, repair, or cleanup is running in the service region.



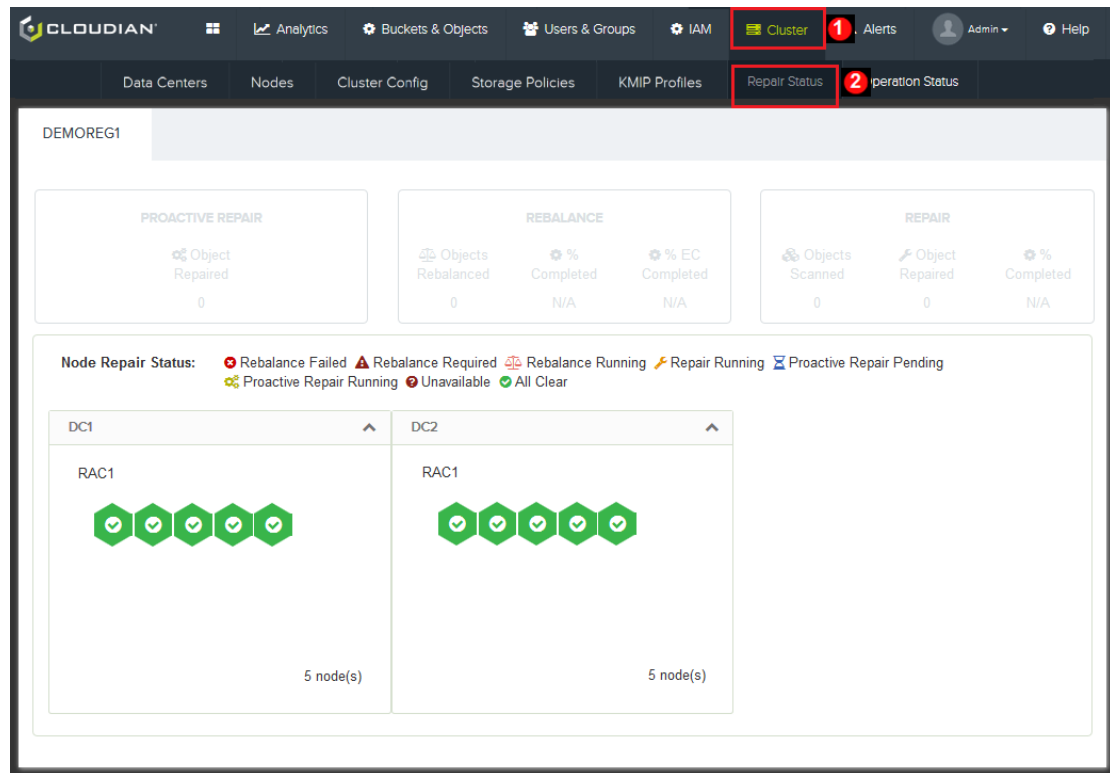
Note If you don't want to wait for an in-progress repair or cleanup of a node to complete you have the option of terminating the operation. To do so, go to the **Node Advanced** page and for that node execute [hsstool repair](#) or [hsstool repairec](#) or [hsstool cleanup](#) or [hsstool cleanupec](#) with the "stop" option selected. Note that an *hsstool rebalance* operation does not support a stop option and cannot be terminated while in progress.

5. In the **Repair Status** page, make sure there are no proactive repairs currently in progress in the service region.

More detail:

If any proactive repairs are in progress, wait until they complete before removing a node from your cluster. The CMC's **Uninstall Node** function will not let you remove a node if a proactive repair is

running in the service region.



Note If you don't want to wait for an in-progress proactive repair of a node to complete you have the option of terminating the repair. To do so, go to the **Node Advanced** page and for that node execute [hsstool proactiverepairq](#) with the "stop" option selected.

6. If the node you want to remove is the active Config Controller node, manually fail over the Config Controller role from the primary instance to the backup instance.

More detail:

If you're not sure whether the node you want to remove is the Config Controller host, check the CMC's **Cluster Information** page. That page also shows which host is the Config Controller backup host.

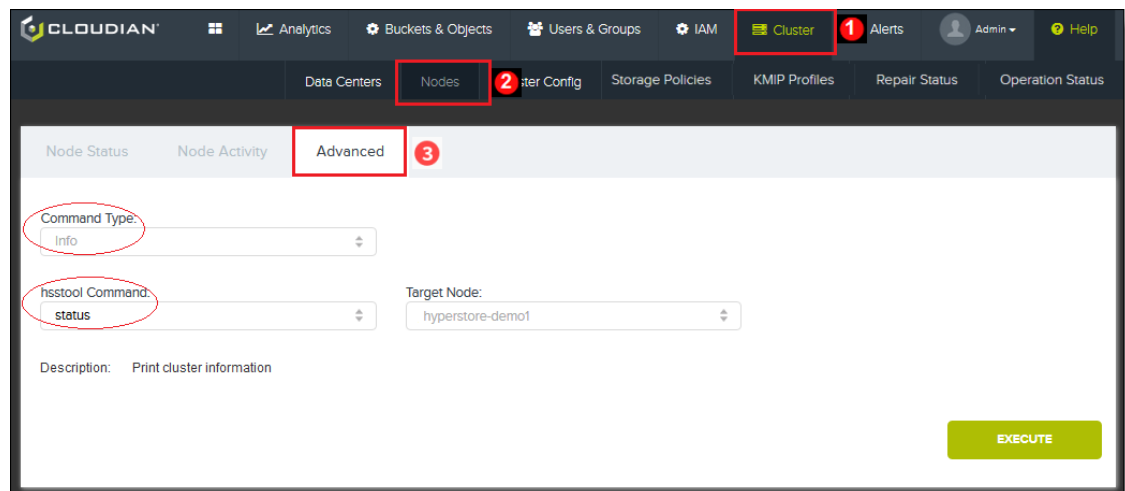
For instructions on failing over the Config Controller from primary to backup, see ["Manually Fail Over the Config Controller Role from the Primary to the Backup"](#) (page 626).

Note Any other specialized services on the node -- such as Redis or Redis Monitor -- will be automatically moved to other nodes in the cluster by the CMC's **Uninstall Node** function.

7. In the **Node Advanced** page, from the Info command type group, submit the `status` command to any healthy node to retrieve status information for all nodes in the region. The **HyperStore Service must be up on all nodes** including the node that you are going to remove. The **Cassandra Service must be up on all nodes other than the one you are going to remove**. If Cassandra is down or unreachable on the node you are removing -- if the node is "dead" in terms of its relation to the Cassandra cluster -- you can still proceed with removing the node, but an extra step will be required toward the end of the node removal procedure (as described in ["Removing a Node"](#) (page 607)).

More detail:

The target node for the command can be any healthy node in the same service region as the node you want to remove -- the command returns status information for the cluster as a whole.



- a. In the command response, for the **node that you want to remove** check the status of the Cassandra Service (in the "Cassandra" column) and the HyperStore Service (in the "Hss" column).
 - If the **Cassandra status and the HyperStore Service status are both Up**, the data redistribution that will be required in the cluster when you remove the node will be executed by a decommissioning process that will be automatically invoked by the CMC's **Uninstall Node** function.
 - If the **Cassandra status is Down or unreachable ("?)**, data redistribution by decommissioning is not supported. Instead, at the end of the remove node procedure you will need to run repair on all the remaining nodes in order to redistribute data in the cluster. Details are in the procedure below.
- Note** If possible, start Cassandra on the node that you want to remove, so that the **Uninstall Node** function can automatically implement a decommissioning process and you won't have to perform repairs.
- If **Cassandra is Up but HyperStore Service is Down or unreachable**, the CMC's **Uninstall Node** function will abort if you try to run it. Before trying to remove the node, start the HyperStore Service on the node (or resolve the network access problem if there is one).
 - b. For **all the other nodes** confirm that Cassandra and the HyperStore Service are Up. The CMC's **Uninstall Node** function will abort if Cassandra or the HyperStore Service are Down or unreachable on any other node in the cluster. If those services are down on any of the other nodes, start the services (or resolve the network access problem if there is one) before trying to remove a node.

7.4.2. Removing a Node

1. If you have not already done so, log in to a CMC instance on a node **other than the node that you are going to remove**, but in the same service region as the node that you are going to remove.

`https://<IP_address_of_node_other_than_removal_node>:8443/Cloudian`

- In the CMC's **Node Advanced** page, from Command Type drop-down list select **Start Maintenance Mode**. For the target node select the node that you want to remove from the cluster.

The screenshot shows the Cloudian CMC interface. The top navigation bar includes 'Cluster' (highlighted with a red box and a red '1') and 'Alerts'. The main navigation bar includes 'Nodes' (highlighted with a red box and a red '2'). The 'Node Advanced' page has three tabs: 'Node Status', 'Node Activity', and 'Advanced' (highlighted with a red box and a red '3'). The 'Command Type' dropdown is set to 'Start Maintenance Mode' (circled in red). The 'Target Node' dropdown is set to 'hyperstore-demo4' (circled in red). The description states: 'Start maintenance mode on target node. By starting maintenance mode all the alert notifications from this node will not be generated and all the S3 operations to this node will be disabled.' An 'EXECUTE' button is at the bottom right.

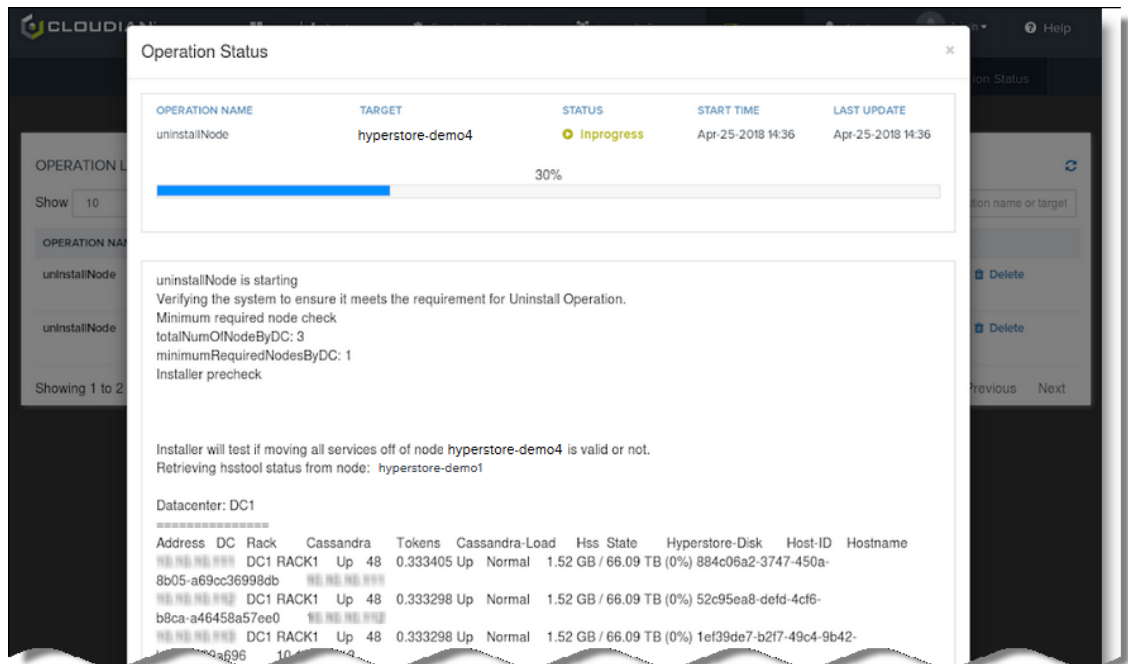
Then click **Execute**. This directs the rest of the cluster to stop sending S3 requests to the specified node.

- In the **Node Advanced** page, from Command Type drop-down list select **Uninstall Node**. For the "Node to Uninstall" list select the node that you want to remove from the cluster. This operation will recreate the selected node's data on to other nodes remaining in the cluster (if the node you are removing is a "live" node), remove the node from the cluster, and remove HyperStore software from the node. If you want also for the HyperStore data and logs to be removed from the node, select that option.

The screenshot shows the Cloudian CMC interface. The top navigation bar includes 'Cluster' (highlighted with a red box and a red '1') and 'Alerts'. The main navigation bar includes 'Nodes' (highlighted with a red box and a red '2'). The 'Node Advanced' page has three tabs: 'Node Status', 'Node Activity', and 'Advanced' (highlighted with a red box and a red '3'). The 'Command Type' dropdown is set to 'Uninstall Node' (circled in red). The 'Node to Uninstall' dropdown is set to 'hyperstore-demo4'. The 'Options' section has a checkbox 'Remove data and logs' which is unchecked. The description states: 'Uninstall a node from the HyperStore cluster. For information about using this operation, including steps to take before and after uninstalling a node, please see the online Help.' An 'EXECUTE' button is at the bottom right.

Then click **Execute**. After you confirm that you want to proceed, the operation is initiated.

- Use the **Operation Status** page to periodically check on the progress of the **Uninstall Node** operation. To pop up a detailed status report click **View** next to the summary status line.



If the node you are removing is live, the **Uninstall Node** operation will include decommissioning the node -- streaming copies of its data to the remaining nodes in the cluster -- and this may take up to several days or more to complete. If the node you are removing is dead, the **Uninstall Node** operation will be much briefer.

Note If you want to remove multiple nodes, wait until the **Uninstall Node** operation completes for one node before you start to uninstall the next node. The system does not support uninstalling multiple nodes concurrently.

- After the **Operation Status** page shows that the status of the **Uninstall Node** operation is Completed, go to the **Node Status** page (**Cluster -> Nodes -> Node Status**) and confirm that the removed node no longer appears in the "Host" drop-down list.
- If the node you removed was "dead" -- meaning that the node's Cassandra Service was down or unreachable when you checked it (as described in ["Preparing to Remove a Node" \(page 603\)](#)) -- you must repair each of the remaining nodes in the region.

More detail:

If the node you removed was "dead", take the following actions to recreate the removed node's data on the remaining nodes in the cluster. (If the node you removed was "live" -- in which case the **Uninstall Node** operation executed a decommissioning process -- these actions are not needed and you can jump down to Step 7.)

Note If you have removed a dead node from your cluster as a precursor to adding a new node to your cluster -- if you wish you can defer the time-consuming repair operations called for below until after you have added the new node(s). If that's the case, you can now perform the ["Adding Nodes" \(page 579\)](#) procedure, and that procedure indicates the point at which you should initiate the deferred repairs. If you have removed a dead node and are **not** adding a new node, perform the repair now as described below.

From the **Node Advanced** page, run [hsstool repair](#) on each of the remaining nodes in the service region. When repairing each node, use the **allkeyspaces** option (so as to repair Cassandra metadata as well as S3 object data) and also the **-pr** option (this makes for more efficient repairs when repairing multiple nodes). Leave the **-l** and **-m** options selected, as they are by default. Since you are using the **-pr** option you can run repair on multiple nodes concurrently (in the GUI you will have to execute the repair command for each node individually, but you do not need to wait for the repair operation on one node to complete before you execute the command on the next node).

The screenshot shows the Cloudfian Node Advanced page. The 'Cluster' tab is selected in the top navigation bar. The 'Nodes' tab is selected in the sub-navigation bar. The 'Advanced' tab is selected in the 'Node Status' section. The 'Command Type' is set to 'Maintenance'. The 'hsstool Command' is set to 'repair'. The 'Target Node' is set to 'hyperstore-demo1'. The 'Options' section shows the following checked options: **allkeyspaces**, **-pr**, **-l**, and **-m**. The 'Description' field contains the text: 'Repair HyperStore node data. For Cassandra Repair please choose allkeyspaces option.' The 'EXECUTE' button is visible at the bottom right.

Also, if you have any erasure coded object data in your system, from the **Node Advanced** page run [hsstool repairrec](#) on one node in each HyperStore data center in the region. It doesn't matter which node you run it on, as long as you do it for one node in each DC in the region. (Note that you do not need to wait for the in-progress *hsstool repair* operations to finish before launching *hsstool repairrec* -- it's OK to run *hsstool repairrec* and *hsstool repair* concurrently.)

The screenshot shows the Cloudfian Node Advanced page. The 'Cluster' tab is selected in the top navigation bar. The 'Nodes' tab is selected in the sub-navigation bar. The 'Advanced' tab is selected in the 'Node Status' section. The 'Command Type' is set to 'Maintenance'. The 'hsstool Command' is set to 'repairrec'. The 'Target Node' is set to 'hyperstore-demo1'. The 'Options' section shows the following checked options: **-l**. The 'Description' field contains the text: 'Repair erasure code HyperStore node data'. The 'EXECUTE' button is visible at the bottom right.

Use the **Operation Status** page to track the progress of all repairs. After **all repairs have completed** proceed to Step 7.

Note Because the *repair* operation repairs erasure coded data on all hosts in a data center, it's potentially a very long running operation. In a large cluster with high data volume it may take multiple weeks to complete.

7. In the **Node Advanced** page, from the Maintenance command group execute the *autorepair* command with the Enable option to **re-enable the HyperStore automated repair features** in the service region. The target node can be any node in the region. Leave the "Type" option unselected so that all repair types are enabled. This also re-enables proactive repair.

This completes the procedure for removing a node from your cluster.

Note that the CMC's **Uninstall Node** function deletes the node from the HyperStore cluster configuration and removes all HyperStore software from the node. It does not delete the Cassandra metadata or HyperStore object data from the node.

IMPORTANT ! If the node was "live" when you removed it -- so that the **Uninstall Node** operation included a decommissioning process -- make sure that in the **Operation Status** page the **Uninstall Node** operation shows as having completed successfully with no errors, before you consider manually deleting the data that remains on the removed node. If the node was "dead" when you removed it -- so that you had to subsequently run repair operations on the remaining nodes in the cluster -- make sure that in the **Operation Status** page the **Uninstall Node** operation and also all those repair operations show as having completed successfully with no errors, before you consider manually deleting the data that remains on the removed node.

IMPORTANT ! If you want to re-use a removed node's host machine by adding it back to your **HyperStore cluster as a new node**, then before using the host machine as a new HyperStore node you must remove all data from the host and re-install the operating system.

7.5. Replacing a Node

If you want to replace a **dead node** (for example a node that has failed due to hardware problems) with a new node:

- First perform the procedure for ["Removing a Node" \(page 603\)](#) (for the dead node).

Note If your current number of nodes is at the minimum required by your storage policies, the system will not allow you to remove the node in the standard way. If this is your circumstance -- you have the minimum number of nodes required by your storage policies and one of those nodes is dead -- please contact Cloudbian Support for guidance.

- Then perform the procedure for ["Adding Nodes" \(page 579\)](#) (for the new node).

If you want to replace a **live node** (a node on which the Cassandra Service and HyperStore Service are running and reachable) with a new node:

- First perform the procedure for ["Adding Nodes" \(page 579\)](#) (for the new node).
- Then perform the procedure for ["Removing a Node" \(page 603\)](#) (for the live node that you want to remove).

7.6. Changing a Node's IP Address

Changing the IP address of an existing installed HyperStore node is not recommended. There is no standardized, supported way of changing a HyperStore node's IP address through the CMC or through command line tools. For more detail on this topic contact Cloudbian Support.

7.7. Restoring a Node That Has Been Offline

- ["Restoring a Node That Has Been Down for Less Than Four Hours" \(page 612\)](#)
- ["Restoring a Node That Has Been Down for More Than Four Hours" \(page 614\)](#)
- ["Restoring a Node That Has Been Down for More Than a Few Days" \(page 614\)](#)

7.7.1. Restoring a Node That Has Been Down for Less Than Four Hours

When a node has been down or inaccessible for just a short while -- less than four hours -- and then you bring it back online, the HyperStore system **automatically** performs the most important actions necessary for restoring the node to a correct and up-to-date condition:

- HyperStore will use [proactive repair](#) to automatically populate the node with any data that the node is responsible for storing but that is missing due to the node having been offline.

IMPORTANT ! The longest node outage that a proactive repair can cover for is four hours (by default configuration). **If a node is down for more than four hours you need to take manual steps to fully repair it.** For detail see ["Restoring a Node That Has Been Down for More Than Four Hours" \(page 614\)](#) or (if applicable to your circumstances) ["Restoring a Node That Has Been Down for More Than a Few Days" \(page 614\)](#).

- If the node that you are restoring is a Redis slave node (for either the Redis Credentials DB or the Redis QoS DB), when you bring the node back online it will automatically sync with the Redis master node to get the most current data.

If you made any configuration changes to your cluster while the node was down, from the Config Controller node [push the changes out to the cluster](#) after the node is back up.

Optionally, you can run a cleanup on the node in order to remove from the node any data that should no longer be there. This would apply, for example, if service users deleted some of their stored objects from the system while the node was down. In this case after being brought back into service the node will still be storing replicas or erasure coded fragments of those objects, resulting in wasted use of disk space. Cleaning the node removes this "garbage" data. For cleanup command details see ["hsstool cleanup"](#) (page 864) (if you have erasure coded data in your system use the command's `-a` option so that it cleans erasure coded data as well as replica data).

Note Before cleaning a node you should wait until any proactive repair that's automatically run on the node has completed. You can check this on the CMC's [Repair Status](#) page. Wait until the node's status displays as "All Clear", and then you can clean the node.

Also optionally, you can return to the node any specialized service role that the node was playing before it went down. If the node had been acting as a "master" or "primary" within one of the HyperStore system's specialized services, then when the node went offline that role would have failed over to a different node. If you want you can return the master or primary role to the restored node after it's back online — though it is not necessary to do so.

How going down and then being brought back up affects a node's specialized service roles...

The table below shows how having been down impacts a node's specialized service role(s).

If before going down the node was...	Then while the node was down that role...	And when brought back online the node is now...
Credentials DB master	Automatically failed over to a Redis Credentials slave	Redis Credentials slave
QoS DB master	Automatically failed over to a Redis QoS slave	Redis QoS slave
Redis Monitor primary	Automatically failed over to the Redis Monitor backup	Redis Monitor backup
Cron job primary	Automatically failed over to the Cron job backup	Cron job backup
Config Controller primary	May or may not have been manually failed over to the Config Controller backup by an administrator	Still Config Controller primary, or else Config Controller backup

To see what specialized service role(s) a restored node is currently playing, go to the CMC's [Cluster Information](#) page.

If you want to change the node's current role assignment(s), see the instructions for ["Change Node Role Assignments"](#) (page 616).

7.7.2. Restoring a Node That Has Been Down for More Than Four Hours

The setting "[hyperstore.repair.proactive.retryQueue.timeout](#)" (page 774) sets the maximum time for which proactive repair jobs can be queued for a node that is unavailable. The default is 4 hours. This time limit prevents the Metadata DB (Cassandra) from being over-loaded with metadata relating to proactive repair, and ensures that proactive repair is used only for its designed purpose, which is to repair object data from a relatively brief time period.

If a node is unavailable for longer than *hyperstore.repair.proactive.retryQueue.timeout*, then the metadata required for implementing proactive repair on the node will stop being written to the Metadata DB and an alert will display in the CMC's [Alerts](#) page. When the node comes back online you will need to:

1. Monitor the automatic proactive repair that initiates on the node when the node starts up, until the proactive repair completes. You can check the CMC's **Repair Status** page periodically to see whether proactive repair is still running on the node that you've brought back online. This proactive repair will repair the new and changed objects from during the period when proactive repair metadata was still being written to the Metadata DB for the node.
2. After proactive repair on the node completes, manually initiate a full repair of the node. For information on manually initiating a repair see [hsstool repair](#) and [hsstool repairec](#). This will repair the new and changed objects from the period after the proactive repair queueing time maximum was reached and before the node came back online.

7.7.3. Restoring a Node That Has Been Down for More Than a Few Days

If you have a node that has been down for more than a few days, consult with Clodian Support for guidance on restoring the node. Special precautions may need to be taken in regard to the Metadata DB (Cassandra) on the node.

7.8. Putting a Node or DC into Maintenance

When you need to perform maintenance on a node or an entire data center, HyperStore supports temporarily removing that node or DC from service and disabling alerts from that node or DC.

7.8.1. Putting a Node into Maintenance

For instructions on putting a node into maintenance see "[Start Maintenance Mode](#)" (page 489).

7.8.2. Putting a Data Center into Maintenance

At a high level, the procedure for putting a data center into and out of maintenance is as described below. Do not put more than one DC into maintenance at a time.

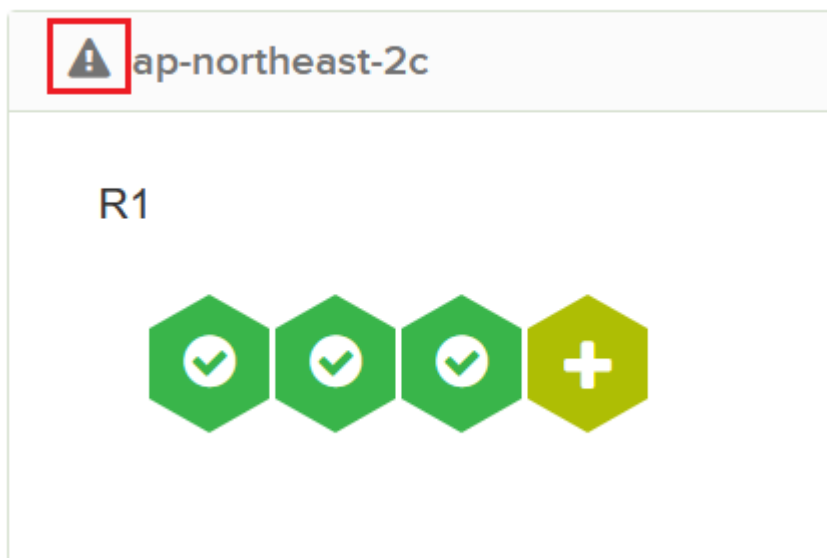
1. For all "Replication Across Data Centers" and "Replicated EC" storage policies that you have configured in your system and that include the DC that you want to put into maintenance, use the CMC's [Storage Policies](#) page to change the read and write consistency requirement to "Local Quorum" (if that is not already the consistency requirement that you are using).
2. Use the CMC's [Cluster Information](#) page to see whether any of the following specialized service roles are being performed by nodes in the DC that you want to put into maintenance; and if so, move each

such role to a node in a different DC. For instructions see ["Change Node Role Assignments"](#) (page 616).

- Credentials DB master node
 - QoS DB master node
 - Redis Monitor primary node
 - Cron job primary node
 - Config Controller primary node
3. Use the CMC's [Alert Rules](#) page to disable alerts related to the DC that you want to put into maintenance. Toward the bottom of the page, in the **DC Alert Configuration** section, set the DC to Disabled.

DC ALERT CONFIGURATION			
ap-northeast-2a	 Enabled	ap-northeast-2c	 Disabled

To confirm that alerts for the DC are disabled, go to the [Data Centers](#) page and next to the DC name you should see a gray triangle with an exclamation mark.



4. Perform your planned maintenance on the DC.

IMPORTANT ! Do not change the IP addresses of the nodes in the DC for which you are performing maintenance.

5. After completing maintenance on the DC, verify network connectivity from your operational DC(s) to the HyperStore nodes in the DC for which you performed maintenance.
6. Use the CMC's **Alert Rules** page to disable alerts related to the DC that you want to put into maintenance. Toward the bottom of the page, in the **DC Alert Configuration** section, set the DC to Enabled. To confirm that alerts for the DC are enabled, go to the **Data Centers** page and next to the DC name you should no longer see a gray triangle with an exclamation mark.

7. For all "Replication Across Data Centers" and "Replicated EC" storage policies that you have configured in your system and that include the DC for which you performed maintenance, use the CMC's **Storage Policies** page to change the read and write consistency requirement back to your desired settings.
8. Use the CMC's [Node Advanced](#) page to execute *hssstool repair* on each node in the DC for which you performed maintenance, one node at a time, allowing the repair to complete on one node before starting a repair on the next node. These are long-running operations. (This step is applicable only if you are using one or more replication [not erasure coding] storage policies in the DC.)
9. Use the CMC's **Node Advanced** page to execute *hssstool repairec* on any one node in the DC for which you performed maintenance. This repairs all erasure coded data in the DC. This is a long-running operation. (This step is applicable only if you are using one or more erasure coding storage policies in the DC.)

7.9. Changing Node Role Assignments

7.9.1. Change Node Role Assignments

In a HyperStore cluster there are certain specialized service roles that are assigned to some nodes and not to others. When you install a HyperStore system, the installer assigns these roles automatically and in a way that's appropriate to your cluster topology. The roles are implemented in such a way that no node constitutes a single point of failure.

For a summary of where specialized service roles are currently assigned in your cluster, go to the CMC's [Cluster Information](#) page.

If you wish you can change node role assignments, by using the HyperStore installer on the Config Controller node. The installer supports the role assignment operations listed below.

Note Each of the role assignment change operations listed below entails doing a configuration push and a restart of the affected service (as described in the instructions for each operation). If you need to perform multiple of these role assignment change operations, do them one operation at a time and with a configuration push and affected service restart at the end of each operation. Do **not** perform multiple different role assignment change operations while deferring the configuration push and service restart.

- ["Move the Credentials DB Master Role or QoS DB Master Role"](#) (page 627)
- ["Move or Add a Credentials DB Slave or QoS DB Slave"](#) (page 617)
- ["Move the Cassandra Seeds Role"](#) (page 618)
- ["Reduce or Change the List of CMC Hosts"](#) (page 620)
- ["Move the Redis Monitor Primary or Backup Role"](#) (page 631)
- ["Move the Cron Job Primary or Backup Role"](#) (page 621)
- ["Move the Config Controller Primary or Backup Role"](#) (page 624)
- ["Change Internal NTP Servers or External NTP Servers"](#) (page 622)

7.9.2. Move or Add a Credentials DB Slave or QoS DB Slave

By default the HyperStore installer deploys two [Credentials DB slaves](#) and one [QoS DB slave](#) per data center. The system supports moving a slave from its current host to a different host -- for example, moving a Credentials DB slave to a host that is not currently running the Credentials DB service. It also supports adding a new Credentials DB or QoS DB slave rather than moving one — so that you end up with more slaves than when you started.

1. On the Config Controller node, change into the installation staging directory (`/opt/cloudian-staging/8.2`) and then launch the HyperStore installer.

```
# ./cloudianInstall.sh
```

If you are using the HyperStore Shell

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Config Controller node you can launch the installer with this command:

```
$ hspkg install
```

Once launched, the installer's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

2. From the installer's main menu, select "Advanced Configuration Options". Then from the Advanced Configuration Options menu select "Change server role assignments". This displays the Change Server Role Assignments menu.

```
Change server role assignments
-----

1 ) Credentials master node (one per whole system)
2 ) Credentials slave nodes to serve read operations
3 ) QOS master nodes (one per region)
4 ) QOS slave nodes serving as read primary
5 ) Cassandra seed nodes
7 ) Primary Redis monitor node
8 ) Backup Redis monitor node
a ) Cronjob/Cluster Monitor node
b ) Config Controller Backup node
c ) NTP Server Configuration
d ) Audit log nodes
r ) Review cluster configuration

x ) Return to previous menu

Choice: █
```

3. From the Change Server Role Assignments menu select the option for Credentials slave nodes or the option for QoS slave nodes.
4. The installer prompts you to specify a comma-separated list of all the hosts on which you want slaves to run. The installer's prompt text indicates [in brackets] which hosts are the **current** slaves. You can use your entry at the prompt to either move a slave or add a slave.

Example of moving a slave:

```
Enter a comma separated list of Credentials slave hosts in region1's
DC1 data center [deneb,vega]: deneb,altair
```

Example of adding a slave:

```
Enter a comma separated list of Credentials slave hosts in region1's
DC1 data center [deneb,vega]: deneb,vega,altair
```

Note If your HyperStore system has multiple data centers, the installer will prompt you separately for each data center's slave host list. If for some data centers you want to continue using the same slave(s) you can just press enter at the prompt rather than entering a host list.

5. After completing the interaction for specifying the slave locations, return to the Change Server Role Assignments menu and select "Review cluster configuration". Then at the prompt confirm that you want to apply the updated configuration.
6. Return to the installer's main menu, then select "Cluster Management". Then at the Cluster Management menu select "Apply Configuration to Cluster". This operation will take a moment as your configuration change is pushed out to each node in the cluster. The terminal will display the status as the operation progresses and completes.
7. Exit the installer. On the Config Controller node, [use the `hscctl` command line tool to restart the following services:](#)
 - Redis Credentials or Redis QoS (whichever service you moved or added a slave for)
 - Redis Monitor
 - S3 Service (restarting this service also results in a restart of the Admin Service)

To verify your change, log into the CMC and go to the [Node Status](#) page. Review the service status information for the node(s) on which you've located the slave(s). Among the listed services on the node(s) should be "Redis Cred" (for a Redis Credentials slave) or "Redis QoS" (for a Redis QoS slave). The absence of "(Master)" appended to the service name indicates it's a slave instance, not a master.

7.9.3. Move the Cassandra Seeds Role

HyperStore's Metadata DB is built on Cassandra, which runs on every node in your HyperStore cluster. A subset of the nodes serve as Cassandra "seed" nodes. Cassandra seed nodes serve to bootstrap the Gossip process for new nodes when you add new nodes to your cluster. Gossip is the peer-to-peer communication protocol by which Cassandra nodes regularly exchange state information. The recommended configuration is to have three of your Cassandra nodes act as seed nodes in each data center in which you have deployed HyperStore. The HyperStore installer automatically makes these role assignments when you install your system.

If you wish you can change the list of Cassandra seed nodes for your system, by following the steps below.

Note Any of your nodes can play the "seed" role. But bear in mind the recommendation that you have three seed nodes per data center. For performance reasons it's not advisable to have more seed nodes than this.

1. On the Config Controller node, change into the installation staging directory (`/opt/cloudian-staging/8.2`) and then launch the HyperStore installer.

```
# ./cloudianInstall.sh
```

If you are using the HyperStore Shell

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Config Controller node you can launch the installer with this command:

```
$ hspkg install
```

Once launched, the installer's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

- From the installer's main menu, select "Advanced Configuration Options". Then from the Advanced Configuration Options menu select "Change server role assignments". This displays the Change Server Role Assignments menu.

```
Change server role assignments
-----

1 ) Credentials master node (one per whole system)
2 ) Credentials slave nodes to serve read operations
3 ) QOS master nodes (one per region)
4 ) QOS slave nodes serving as read primary
5 ) Cassandra seed nodes
7 ) Primary Redis monitor node
8 ) Backup Redis monitor node
a ) Cronjob/Cluster Monitor node
b ) Config Controller Backup node
c ) NTP Server Configuration
d ) Audit log nodes
r ) Review cluster configuration

x ) Return to previous menu

Choice: █
```

- From the Change Server Role Assignments menu select "Cassandra seed nodes".
- At the prompt enter a comma-separated list of hosts that you want to serve as Cassandra seed nodes. If you want to keep the same host just press Enter rather than specifying a different host. (If you have a multi-region system, you will be prompted for a list of Cassandra seed nodes for each region.)
- After completing the interaction for specifying Cassandra seed nodes, return to the Change Server Role Assignments menu and select "Review cluster configuration". Then at the prompt confirm that you want to apply the updated configuration.
- Return to the installer's main menu, then select "Cluster Management". Then at the Cluster Management menu select "Apply Configuration to Cluster". This operation will take a moment as your configuration change is pushed out to each node in the cluster. The terminal will display the status as the operation progresses and completes.
- Exit the installer. On the Config Controller node, [use the `hscctl` command line tool to restart the Cassandra Service](#).

To verify your change, you can look at the `/opt/cassandra/conf/cassandra.yaml` configuration file on any one of your nodes and confirm that the "seeds:" parameter is set to the list of hosts that you specified.

7.9.4. Reduce or Change the List of CMC Hosts

By default the Cloudbian Management Console (CMC) service is installed and runs on all of your HyperStore hosts. If you wish, after initial deployment of your system you can use the installer to specify a list of HyperStore hosts on which to have the CMC run, rather than having it run on all nodes.

1. On the Config Controller node, change into the installation staging directory (*/opt/cloudian-staging/8.2*) and then launch the HyperStore installer.

```
# ./cloudianInstall.sh
```

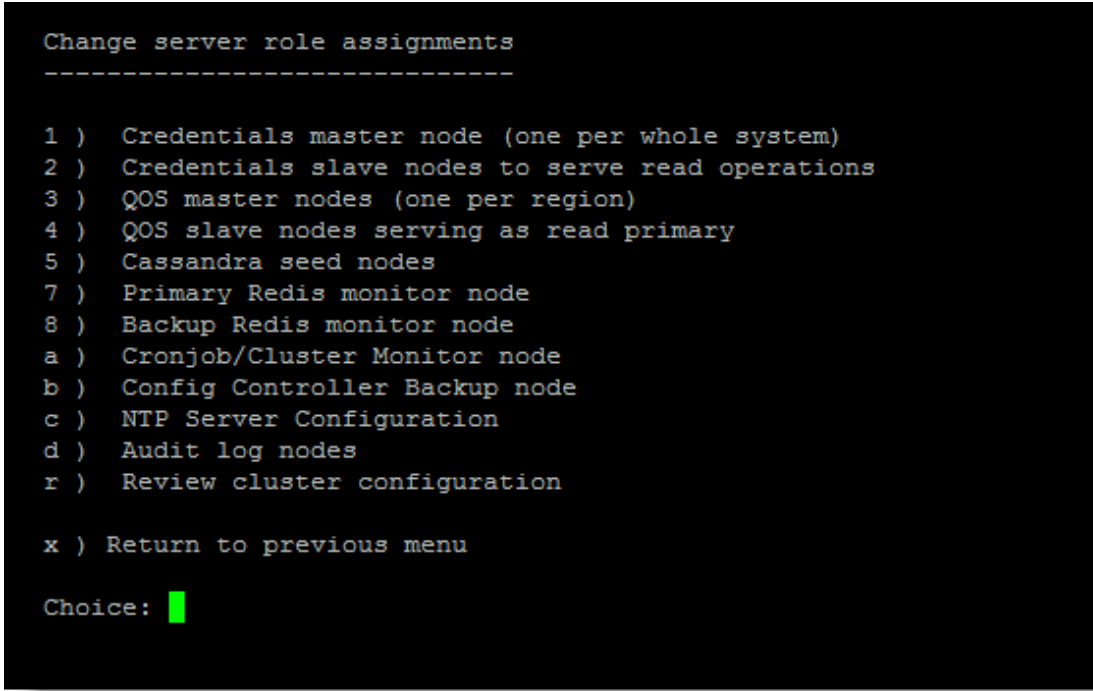
If you are using the HyperStore Shell

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Config Controller node you can launch the installer with this command:

```
$ hspkg install
```

Once launched, the installer's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

2. From the installer's main menu, select "Advanced Configuration Options". Then from the Advanced Configuration Options menu select "Change server role assignments". This displays the Change Server Role Assignments menu.



```
Change server role assignments
-----

1 ) Credentials master node (one per whole system)
2 ) Credentials slave nodes to serve read operations
3 ) QOS master nodes (one per region)
4 ) QOS slave nodes serving as read primary
5 ) Cassandra seed nodes
7 ) Primary Redis monitor node
8 ) Backup Redis monitor node
a ) Cronjob/Cluster Monitor node
b ) Config Controller Backup node
c ) NTP Server Configuration
d ) Audit log nodes
r ) Review cluster configuration

x ) Return to previous menu

Choice: █
```

3. From the Change Server Role Assignments menu select "CMC nodes".
4. At the prompt enter a comma-separated list of hosts on which you want the CMC to run.
5. After completing the interaction for specifying your CMC host list, return to the Change Server Role Assignments menu and select "Review cluster configuration". Then at the prompt confirm that you want to apply the updated configuration.
6. Return to the installer's main menu, then select "Cluster Management". Then at the Cluster Management menu select "Apply Configuration to Cluster". This operation will take a moment as your

configuration change is pushed out to each node in the cluster. The terminal will display the status as the operation progresses and completes.

7. After the applying of the configuration change to the cluster has completed, exit the installer. On the Config Controller node, [use the `hsctl` command line tool to restart the CMC](#).

7.9.5. Move the Cron Job Primary or Backup Role

Certain HyperStore [system maintenance tasks are implemented by cron jobs](#) that run on a regular schedule, as configured by a crontab. When your HyperStore system is installed the install script designates one node to host the crontab configuration and run the cron jobs, and a second node to serve as a backup. In the event that the primary cron job host goes offline (or if `crond` goes down) the backup host **automatically** takes over as the active cron job host.

The HyperStore [Monitoring Data Collector](#) resides on the same primary host and backup host as the cron jobs. If the cron jobs role automatically fails over from the primary host to the backup host, the Monitoring Data Collector role also fails over to the backup.

The system supports moving the primary cron job role (and with it, the primary Monitoring Data Collector role) to a different host as described below. The same procedure also supports moving the backup cron job role (and with it, the backup Monitoring Data Collector role) to a different host.

Note Do not assign the primary cron job role to the same host as your Config Controller role. If the cron job primary and the Config Controller are on the same host and that host goes down, automated fail-over from the cron job primary to the cron job backup will not work.

1. On the Config Controller node, change into the installation staging directory (`/opt/cloudian-staging/8.2`) and then launch the HyperStore installer.

```
# ./cloudianInstall.sh
```

If you are using the HyperStore Shell

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Config Controller node you can launch the installer with this command:

```
$ hspkg install
```

Once launched, the installer's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

2. From the installer's main menu, select "Advanced Configuration Options". Then from the Advanced Configuration Options menu select "Change server role assignments". This displays the Change Server Role Assignments menu.

```

Change server role assignments
-----

1 ) Credentials master node (one per whole system)
2 ) Credentials slave nodes to serve read operations
3 ) QOS master nodes (one per region)
4 ) QOS slave nodes serving as read primary
5 ) Cassandra seed nodes
7 ) Primary Redis monitor node
8 ) Backup Redis monitor node
a ) Cronjob/Cluster Monitor node
b ) Config Controller Backup node
c ) NTP Server Configuration
d ) Audit log nodes
r ) Review cluster configuration

x ) Return to previous menu

Choice: █

```

3. From the Change Server Role Assignments menu select "Cronjob/Cluster Monitor node".
4. At the prompts specify your desired primary host and backup host for the cron jobs / cluster monitor.
5. After completing the interaction for specifying cron job hosts, return to the Change Server Role Assignments menu and select "Review cluster configuration". Then at the prompt confirm that you want to apply the updated configuration.
6. Return to the installer's main menu, then select "Cluster Management". Then at the Cluster Management menu select "Apply Configuration to Cluster". This operation will take a moment as your configuration change is pushed out to each node in the cluster. The terminal will display the status as the operation progresses and completes.

To verify your change, log into the CMC and go to the [Cluster Information](#) page. Review the service information section to confirm that your System Monitoring/Cronjob primary and backup hosts are what you want them to be.

7.9.6. Change Internal NTP Servers or External NTP Servers

When you install your HyperStore cluster, for each of your data centers the installation script automatically configures four of your HyperStore nodes to act as internal NTP servers for the cluster. If a data center has four or fewer HyperStore nodes in it, then all of the nodes act as internal NTP servers.

These internal NTP servers synchronize with external NTP servers. By default the set of external servers is:

- 0.rocky.pool.ntp.org
- 1.rocky.pool.ntp.org
- 2.rocky.pool.ntp.org
- 3.rocky.pool.ntp.org

Note For more on how HyperStore automatically configures an NTP set-up for the cluster, see ["NTP Automatic Set-Up"](#) (page 857)

As described below, the system supports changing the list of internal NTP servers (for data centers in which you have more than four HyperStore nodes) and/or changing the list of external NTP servers.

1. On the Config Controller node, change into the installation staging directory (`/opt/cloudian-staging/8.2`) and then launch the HyperStore installer.

```
# ./cloudianInstall.sh
```

If you are using the HyperStore Shell

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Config Controller node you can launch the installer with this command:

```
$ hspkg install
```

Once launched, the installer's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

2. From the installer's main menu, select "Advanced Configuration Options". Then from the Advanced Configuration Options menu select "Change server role assignments". This displays the Change Server Role Assignments menu.

```
Change server role assignments
-----

1 ) Credentials master node (one per whole system)
2 ) Credentials slave nodes to serve read operations
3 ) QOS master nodes (one per region)
4 ) QOS slave nodes serving as read primary
5 ) Cassandra seed nodes
7 ) Primary Redis monitor node
8 ) Backup Redis monitor node
a ) Cronjob/Cluster Monitor node
b ) Config Controller Backup node
c ) NTP Server Configuration
d ) Audit log nodes
r ) Review cluster configuration

x ) Return to previous menu

Choice: █
```

3. From the Change Server Role Assignments menu select "NTP Server Configuration".
4. At the first prompt specify the HyperStore hosts that you want to act as the internal NTP servers, as a comma-separated list. You must use host names, not IP addresses. If you want to keep the same hosts that the system is currently using, just press Enter rather than specifying different hosts.

Note If you have multiple HyperStore data centers you will be prompted separately for the internal NTP host list for each data center.

5. At the next prompt specify the external NTP servers with which you want the internal NTP servers to synchronize, as a comma-separated list. For these external servers you can use either FQDNs or IP addresses. If you want to keep the same external servers that the system is currently using, just press Enter rather than specifying a different server list.
6. After completing the interaction for specifying NTP Server Configuration, return to the Change Server Role Assignments menu and select "Review cluster configuration". Then at the prompt confirm that you want to apply the updated configuration.
7. Return to the installer's main menu, then select "Cluster Management". Then at the Cluster Management menu select "Apply Configuration to Cluster". This operation will take a moment as your configuration change is pushed out to each node in the cluster. The terminal will display the status as the operation progresses and completes.

To verify your change, log into the CMC and go to the [Cluster Information](#) page. Review the service information section to confirm that your internal NTP hosts and external NTP hosts are what you want them to be.

7.9.7. Move the Config Controller Primary or Backup Role

Note HyperStore supports a configurable option to have the Config Controller primary role fail over to the backup host automatically rather than requiring manual intervention. See [com-mon.configController.failover](#). By default automatic failover is disabled and manual intervention is required as described below in ["Manually Fail Over the Config Controller Role from the Primary to the Backup"](#) (page 626).

When you install your HyperStore system, you choose the node that you want to serve as the Config Controller for cluster configuration management, and you run the installer on that node. The installer configures that node to be the primary Config Controller, and also configures a second node to be the Config Controller backup. If the primary goes down, you can manually fail over the active Config Controller role to the backup host.

There are two different operations that you can perform in regard to the Config Controller role:

- ["Move the Config Controller Backup"](#) (page 624)
- ["Manually Fail Over the Config Controller Role from the Primary to the Backup"](#) (page 626)

7.9.7.1. Move the Config Controller Backup

In this scenario your primary Config Controller is up and running properly, and you're simply looking to relocate the Config Controller backup role to a different host than it is currently on. Note that in a multi-region system, the Config Controller backup role must reside in the default service region (as well as the Config Controller primary role).

Note You can find out which host is currently the Config Controller backup host in the CMC's [Cluster Information](#) page.

1. On the Config Controller primary host, change into the installation staging directory (`/opt/cloudian-staging/8.2`) and then launch the HyperStore installer.

```
# ./cloudianInstall.sh
```

If you are using the HyperStore Shell

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Config Controller node you can launch the installer with this command:

```
$ hspkg install
```

Once launched, the installer's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

2. From the installer's main menu, select "Advanced Configuration Options". Then from the Advanced Configuration Options menu select "Change server role assignments". This displays the Change Server Role Assignments menu.

```
Change server role assignments
-----

1 ) Credentials master node (one per whole system)
2 ) Credentials slave nodes to serve read operations
3 ) QOS master nodes (one per region)
4 ) QOS slave nodes serving as read primary
5 ) Cassandra seed nodes
7 ) Primary Redis monitor node
8 ) Backup Redis monitor node
a ) Cronjob/Cluster Monitor node
b ) Config Controller Backup node
c ) NTP Server Configuration
d ) Audit log nodes
r ) Review cluster configuration

x ) Return to previous menu

Choice: █
```

3. From the Change Server Role Assignments menu select "Config Controller Backup node".
4. At the prompt specify the host to which you want to move the Config Controller backup role.
5. After completing the interaction for specifying the Config Controller backup host, return to the Change Server Role Assignments menu and select "Review cluster configuration". Then at the prompt confirm that you want to apply the updated configuration.
6. Return to the installer's main menu, then select "Cluster Management". Then at the Cluster Management menu select "Apply Configuration to Cluster". This operation will take a moment as your configuration change is pushed out to each node in the cluster. The terminal will display the status as the operation progresses and completes.
7. After the applying of the configuration change to the cluster has completed, exit the installer and wait at least one minute, then log into the CMC and go to the [Cluster Information](#) page. Review the service information section to confirm that your Config Controller primary and backup hosts are what you want them to be.

7.9.7.2. Manually Fail Over the Config Controller Role from the Primary to the Backup

In this scenario there is a problem with your Config Controller primary node and you want the Config Controller backup to become active.

Note that in the procedure below you will be logging into the current Config Controller **backup** host. If you're not sure which node is the Config Controller backup host you can check on the CMC's **Cluster Information** page.

1. **On the Config Controller backup host**, change into the installation directory and then launch the Hyper-Store installer.

```
# ./cloudianInstall.sh
```

2. If the current Config Controller primary host **is not** down or inaccessible, jump ahead to Step 3.

If the current Config Controller primary host **is** down or inaccessible, then in the installer main menu select "Advanced Configuration Options" and then select "Exclude host(s) from configuration push and service restarts". Following the prompts, choose to exclude the current Config Controller primary host from configuration push and service restarts (you will specify it by its hostname). Then return to the "Advanced Configuration Options" menu and proceed with Step 3.

3. In the installer main menu select "Advanced Configuration Options" and then select "Change server role assignments". This displays the Change Server Role Assignments menu.

```
Change server role assignments
-----

1 ) Credentials master node (one per whole system)
2 ) Credentials slave nodes to serve read operations
3 ) QOS master nodes (one per region)
4 ) QOS slave nodes serving as read primary
5 ) Cassandra seed nodes
7 ) Primary Redis monitor node
8 ) Backup Redis monitor node
a ) Cronjob/Cluster Monitor node
b ) Config Controller Backup node
c ) NTP Server Configuration
d ) Audit log nodes
r ) Review cluster configuration

x ) Return to previous menu

Choice: █
```

4. From the Change Server Role Assignments menu select "Config Controller Backup node".
5. At the prompt specify the host to which you want to move the Config Controller backup role. You need a new backup because you are converting the original backup into the primary (by running through this procedure using the installer on the original backup -- this has the effect of turning it into the new primary). Note that in a multi-region system, the Config Controller backup role must reside in the default service region.

6. After completing the interaction for specifying the Config Controller backup host, return to the Change Server Role Assignments menu and select "Review cluster configuration". Then at the prompt confirm that you want to apply the updated configuration.
7. Return to the installer's main menu, then select "Cluster Management". Then at the Cluster Management menu select "Apply Configuration to Cluster". This operation will take a moment as your configuration change is pushed out to each node in the cluster. The terminal will display the status as the operation progresses and completes.
8. After the applying of the configuration change to the cluster has completed, exit the installer and use the `hscctl` command line tool to [restart the CMC](#).

To verify your change, log into the CMC and go to the [Cluster Information](#) page. Review the service information section to confirm that your Config Controller primary and backup hosts are what you want them to be. The former backup (from which you implemented the above procedure) should now be the primary and the new backup should be as you specified during the procedure.

Note If the Config Controller primary is now on the same host as the cron job primary role you should [move the cron job primary role](#) to a different host. If the cron job primary and the running Config Controller are on the same host and that host goes down, automated fail-over from the cron job primary to the cron job backup will not work.

Note If the former Config Controller primary node was down or inaccessible and you excluded it from configuration updates (Step 2 in the procedure above), then when that host comes back online remove it from its excluded status. You can do this by using the installer on the new Config Controller primary node to do "Advanced Configuration Options" → "Exclude host(s)..." and when prompted for which nodes to exclude, enter "none". This does not make that node the Config Controller primary or backup node -- it just makes the node eligible to receive cluster configuration updates again.

7.9.8. Move the Credentials DB Master Role or QoS DB Master Role

The [Credentials DB Master](#) role is assigned to just one node in your entire HyperStore system (the system has just one Credentials DB Master even if there are multiple service regions). The [QoS DB Master](#) role is assigned to one node in each of your service regions (each region has its own QoS DB Master). The HyperStore installer automatically makes these role assignments when you install your system.

If you wish you can move the Credentials DB Master role to a current Credentials DB slave node, or move a QoS Master DB role to a current QoS DB slave node, by following the procedure in this section. If you do so, the node that had been Master will automatically become a slave.

Note The system does not support moving the Master role to a node that is not currently a slave.

7.9.8.1. Preparing to Move the Master Role

For safety, before moving the Credentials DB Master role or QoS DB Master role make a backup copy of the current database from the Master. You can do so by using the Redis CLI command [BGSAVE](#) as described below.

To back up a **Credentials DB Master** database:

```
# /opt/redis/redis-cli -h <internal_IPaddress_of_master_node> -p 6379 BGSAVE
```

The above command will update the *dump-credentials.rdb* file in the Redis data directory (*/var/lib/redis* by default).

To back up a **QoS DB Master** database:

```
# /opt/redis/redis-cli -h <internal_IPaddress_of_master_node> -p 6380 BGSAVE
```

The above command will update the *dump-qos.rdb* file in the Redis data directory (*/var/lib/redis* by default).

Note The BGSAVE operation saves the database in the background. To check to see whether the save is completed yet you can use the Redis CLI [LASTSAVE](#) command, which returns the Unix time of the most recently completed save. Do not move the Master role until the saving of the database backup is completed.

7.9.8.2. Moving the Credentials DB Master Role or QoS DB Master Role

1. Submit a Redis CLI *info* command to the **slave node** to which you want to move the Master role. Confirm that the slave node returns the following status information:

- *role* is slave
- *master_host* is the current Master
- *master_link_status* is up
- *master_sync_in_progress* is 0

When submitting the Redis CLI command use port 6379 if the target node is a Credentials DB slave, or use port 6380 if the target node is a QoS DB slave.

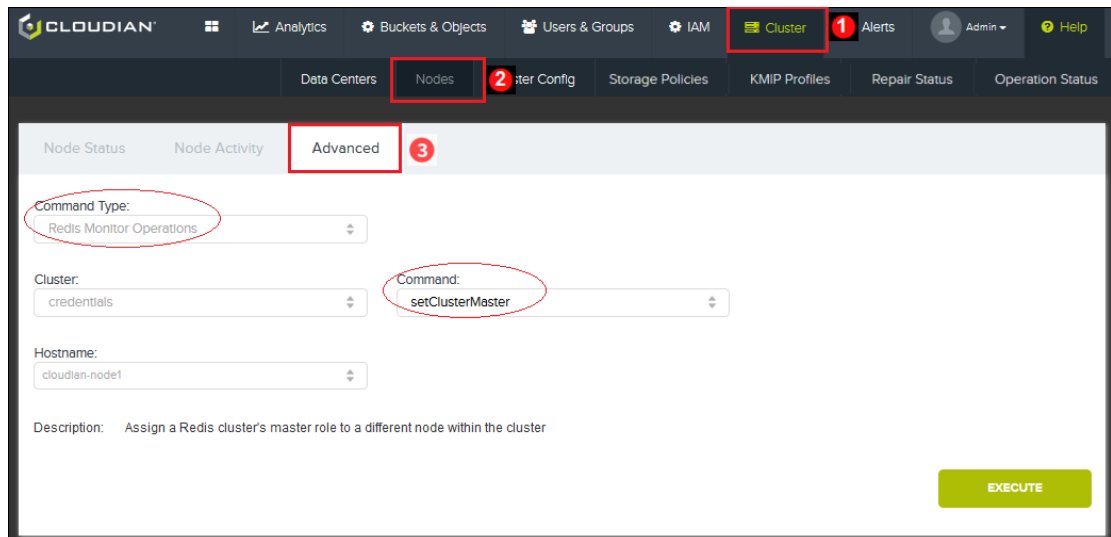
For example, if the Credentials DB Master role is currently on host "cloudian-node2" and you want to move it to host "cloudian-node7" where there is currently a Credentials DB slave:

```
# /opt/redis/redis-cli -h <internal_IPaddress_of_cloudian-node7> -p 6379 info | egrep
"role|master_host|master_link_status|master_sync_in_progress"

role:slave
master_host:cloudian-node2
master_link_status:up
master_sync_in_progress:0
```

2. Dynamically switch the Master role to the slave:

Log into the CMC and go to **Cluster** → **Nodes** → **Advanced**. Select Command Type "Redis Monitor Operations", then select a Cluster type (Credentials or QoS) and select Command "setClusterMaster".



For "Hostname" select the host to which you want to move the Master role. The drop-down list will show only nodes that are currently slaves within the cluster type (Credentials or QoS) that you selected.

After making your selections, click **Execute**. The chosen slave will become the Master, while the Master becomes a slave. The change happens immediately upon command execution.

3. Make the switch of the Master and slave permanent by updating your system configuration:
 - a. On the Config Controller node (not the Credentials DB Master or QoS DB Master but rather the Config Controller from which cluster configuration is managed) change into the installation staging directory (`/opt/cloudian-staging/8.2`) and then launch the HyperStore installer.

```
# ./cloudianInstall.sh
```

If you are using the HyperStore Shell

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Config Controller node you can launch the installer with this command:

```
$ hspkg install
```

Once launched, the installer's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

- b. From the installer's main menu, select "Advanced Configuration Options". Then from the Advanced Configuration Options menu select "Change server role assignments". This displays the Change Server Role Assignments menu.

```

Change server role assignments
-----

1 ) Credentials master node (one per whole system)
2 ) Credentials slave nodes to serve read operations
3 ) QOS master nodes (one per region)
4 ) QOS slave nodes serving as read primary
5 ) Cassandra seed nodes
7 ) Primary Redis monitor node
8 ) Backup Redis monitor node
a ) Cronjob/Cluster Monitor node
b ) Config Controller Backup node
c ) NTP Server Configuration
d ) Audit log nodes
r ) Review cluster configuration

x ) Return to previous menu

Choice: █

```

- c. From the Change Server Role Assignments menu select the option for the master that you want to move — Credentials DB Master or QoS DB Master.
- d. At the prompt indicate the host to which you want to move the Master role. This should be the same host that you checked in Step 1.
- e. After completing the interaction for specifying the new Master host, return to the Change Server Role Assignments menu and select "Review cluster configuration". Then at the prompt confirm that you want to apply the updated configuration.
- f. Return to the installer's main menu, then select "Cluster Management". Then at the Cluster Management menu select "Apply Configuration to Cluster". This operation will take a moment as your configuration change is pushed out to each node in the cluster. The terminal will display the status as the operation progresses and completes.
- g. After the applying of the configuration change to the cluster has completed, exit the installer. On the Config Controller node, [use the `hscctl` command line tool to restart the following services](#), one service at a time:
 - The DB service for which you've changed the master role (either Redis Credentials or Redis QoS)
 - HyperStore Service
 - S3 Service
 - Redis Monitor
 - IAM Service

To verify your change, log into the CMC and go to the [Cluster Information](#) page. Review the service information section to confirm that the Master that you moved is now where you want it to be.

Now, the former Credentials DB or QoS DB slave has been promoted to Master and the former Master has been demoted to slave.

IMPORTANT ! If you want to remove the demoted host (the former Master that's now a slave) from your cluster, you must first move its slave role to a different host in your cluster. For instructions see ["Move or Add a Credentials DB Slave or QoS DB Slave"](#) (page 617).

7.9.9. Move the Redis Monitor Primary or Backup Role

The [Redis Monitor](#) runs on two nodes in your cluster, with one being the primary Redis Monitor instance and the other being a backup instance. In the event that the Redis Monitor primary goes offline the Redis Monitor backup **automatically** detects this and takes over as the active Redis Monitor.

IMPORTANT ! In a multi- data center HyperStore system, the Redis Monitor backup must remain in the same data center as the Redis Monitor primary, and this must be the same data center as where the Credentials DB Master is located.

The system supports moving the primary or backup Redis Monitor to a different host as described below.

1. On the Config Controller node, change into the installation staging directory (*/opt/cloudian-staging/8.2*) and then launch the HyperStore installer.

```
# ./cloudianInstall.sh
```

If you are using the HyperStore Shell

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Config Controller node you can launch the installer with this command:

```
$ hspkg install
```

Once launched, the installer's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

2. From the installer's main menu, select "Advanced Configuration Options". Then from the Advanced Configuration Options menu select "Change server role assignments". This displays the Change Server Role Assignments menu.

```

Change server role assignments
-----

1 ) Credentials master node (one per whole system)
2 ) Credentials slave nodes to serve read operations
3 ) QOS master nodes (one per region)
4 ) QOS slave nodes serving as read primary
5 ) Cassandra seed nodes
7 ) Primary Redis monitor node
8 ) Backup Redis monitor node
a ) Cronjob/Cluster Monitor node
b ) Config Controller Backup node
c ) NTP Server Configuration
d ) Audit log nodes
r ) Review cluster configuration

x ) Return to previous menu

Choice: █

```

3. From the Change Server Role Assignments menu select the option for the Redis Monitor instance that you want to move (Primary Redis Monitor or Backup Redis Monitor).
4. At the prompt specify the host to which you want to move the Redis Monitor instance.
5. After completing the interaction for specifying the new Redis Monitor location, return to the Change Server Role Assignments menu and select "Review cluster configuration". Then at the prompt confirm that you want to apply the updated configuration.
6. Return to the installer's main menu, then select "Cluster Management". Then at the Cluster Management menu select "Apply Configuration to Cluster". This operation will take a moment as your configuration change is pushed out to each node in the cluster. The terminal will display the status as the operation progresses and completes.
7. Exit the installer. On the Config Controller node, [use the `hsctl` command line tool to restart the Redis Monitor service](#).

To verify your change, log into the CMC and go to the [Cluster Information](#) page. Review the service information section to confirm that your Redis Monitor primary and backup hosts are what you want them to be.

7.10. Cron Jobs and Automated System Maintenance

The HyperStore system automatically executes a variety of maintenance tasks. This section describes the automated maintenance jobs that the system implements. The covered topics are:

- ["System cron Jobs" \(page 632\)](#)
- ["Cassandra Data Compaction" \(page 636\)](#)

7.10.1. System cron Jobs

Most HyperStore automated maintenance routines are implemented as cron jobs. Maintenance cron jobs are run from one HyperStore node in each of your service regions. To see which of your nodes is running the cron

jobs, go to the CMC's **Cluster Information** page (**Cluster -> Cluster Config -> Cluster Information**). In that page's Service Information section you will see the identity of the System Monitoring / Cronjob Primary Host, from which the cron jobs are run.

Note You will also see the identity of the System Monitoring / Cronjob Backup Host. If the primary cron job instance goes down (due to the host going down or *crond* going down on the host) and remains down for 10 minutes, the backup cron job instance will automatically take over the primary role and start running the system cron jobs.

The cron jobs themselves are specified in the `/etc/cron.d/cloudian-crontab` file on the System Monitoring / Cronjob host node. Do not edit this file directly. Instead, if you need to enable/disable a cron job or change a cron job's scheduling, you can do so as described below:

To enable or disable a cron job, on the Config Controller node -- which may be a different node than the Cronjob host node -- run the following commands (where `<jobName>` is one of the cron job names from the table further below, and `<value>` is *true* or *false*):

```
# hsctl config set jobs.system.<jobName>.enabled=<boolean>
# hsctl config apply jobs
```

For example:

```
# hsctl config set jobs.system.kmipRekey.enabled=true
# hsctl config apply jobs
```

Note All cron jobs are enabled by default with the exception of the *onDemandLogCollector* job and the *kmipRekey* job which are disabled by default.

To change the scheduling of a cron job, on the Config Controller node run the following commands (where `<value>` is specified in standard crontab schedule formatting, enclosed in quotes):

```
# hsctl config set jobs.system.<jobName>.schedule=<string>
# hsctl config apply jobs
```

For example:

```
# hsctl config set jobs.system.bucketLifecycle.schedule="0 4 * * *"
# hsctl config apply jobs
```

When enabling/disabling a cron job, or changing a cron job's schedule, no service restart is required.

The table below lists the configurable cron jobs along with their default schedule and their purpose.

Cron Job Name	Schedule	Purpose
bucketInventory	0 0 * * * (0:00am UTC daily)	This cron job generates bucket content inventories, for any buckets that have bucket inventories configured.
bucketLifecycle	0 23 * * * (11pm UTC daily)	This cron job performs two tasks in support of S3 bucket lifecycle policies: Transitions (auto-tiers) objects to a tiering destination system, if the objects have reached their scheduled auto-tiering interval or date.

Cron Job Name	Schedule	Purpose
		Deletes objects from HyperStore storage (or from the remote tiered storage system if they've been auto-tiered), if the objects have reached their scheduled expiration interval or date. The cron job distributes auto-tiering and auto-expiry processing workload across the cluster. Specifically, the auto-tiering and auto-expiry processing workload associated with a given storage policy is spread out across all the nodes that participate in that storage policy. For example, suppose you have a HyperStore system with two data centers named DC1 and DC2, and one of your storage policies stores data only in DC1. When the <code>./system/autoexpire</code> cron job runs, the auto-tiering and auto-expiry processing work for buckets that use that storage policy will be allocated among the nodes in DC1, so that each node in DC1 is doing the processing for a subset of those buckets. If you have a storage policy that stores data in both DC1 and DC2, then the auto-tiering and auto-expiry processing workload associated with buckets that use that storage policy will be spread among the nodes in both data centers. Note For more information on the HyperStore auto-tiering feature, see "Auto-Tiering Feature Overview" (page 203).
bucketLifecycleRestore	0 */6 * * * (every 6 hours)	This cron job executes queued object-restore jobs. Restore jobs are placed in queue when S3 clients invoke the S3 API method <i>RestoreObject</i> , to restore a local copy of objects that have been auto-tiered to a remote storage system.
bucketLogging	*/10 * * * * (every 10 minutes)	This cron job moves bucket logging data from the Metadata DB's BucketLog column family into the S3 storage system, in support of the S3 Bucket Logging feature.
bucketProtectionPolicy	0 23 * * * (11:00pm UTC daily)	This cron job processes any pending storage policy delete jobs. System operators can initiate the deletion of an unused storage policy (a storage policy that is not assigned to any buckets) through the CMC's Storage Policies page. This operator action marks the policy with a "DELETED" flag and makes it immediately unavailable to service users. However, the full deletion of the storage policy from the system (specifically, the deletion of the Metadata DB keyspace associated with the policy) is not processed until this cron job runs. This cron job also processes any pending storage policy creation jobs, in the event that multiple storage policy creation requests have been initiated in a short amount of time -- which can result in queueing of storage policy creation jobs. More typically, storage policy creation completes shortly after the creation is initiated through the CMC.
cleanTombstones	0 * * * * (every hour)	This hourly operation cleans (removes) Cassandra "tombstones", which are markers that indicate that a table cell or row has been deleted. Dealing with Excessive Tombstone Build-Up

Cron Job Name	Schedule	Purpose
		Under normal circumstances the hourly running of the <code>./system/cleantombstones</code> cron job should ensure that there is no excessive build-up of tombstones in Cassandra (the Metadata DB). However, it is possible to encounter <code>TombstoneOverwhelmingException</code> errors in Cassandra logs and an inability to successfully execute an <code>S3 ListObjects</code> operation against a specific bucket, in unusual circumstances such as an extremely high rate of recent S3 deletes or a failure of the <code>./system/cleantombstones</code> cron job. You can trigger tombstone cleanup on-demand by running this command on any HyperStore node: <pre>hsstool -h <IP address> cleanuptombstone -start</pre> For more information see hsstool cleanuptombstone.
cleanupUsers	0 1 * * * (1am UTC daily)	This cron job attempts to complete the user deletion process for users for whom the deletion process failed to complete on the original attempt. These are users who are stuck in "Deleting" status for one reason or another.
crossRegionReplication	0 */4 * * * (every 4 hours)	This cron job processes any pending cross-region replication tasks. Typically with cross-region replication, objects are replicated to the destination bucket as soon as they are uploaded to the source bucket. Pending cross-region replication tasks result when an attempt to replicate an object from the source bucket to the destination bucket returns a temporary error such as a connection failure or an HTTP 5xx error. This cron job retries the replication of such objects. For such objects, the retries will recur once every four hours until either the objects are successfully replicated to the destination system or a permanent error is encountered. Note If you set the configuration parameter <code>"s3.crr.syncBucket"</code> (page 817) to <code>true</code>, then for each source bucket for which there are pending replication retry tasks this cron job will also perform a complete sync-up of the destination bucket contents to the source bucket contents. By default <code>s3.crr.syncBucket</code> is set to <code>false</code>.
kmipRekey	0 0 1 * * (1st of every month)	By default this cron job is disabled. For information about enabling the cron job, see "Re-Keying Buckets" (page 194).
notifications	0 * * * * (every hour)	This cron job retries S3 Bucket Notification messages that failed to be successfully sent on previous attempts. For more information about the S3 Notification feature see the SQS section in the <i>Cloudian HyperStore AWS APIs Support Reference</i> .
onDemandLogCollector	* /10 * * * *	This cron job is disabled by default. If you enable it, this cron job will automatically upload to Cloudian Support the node diagnostics logs that Cloudian Support needs to process any support cases that you

Cron Job Name	Schedule	Purpose
	(every 10 minutes)	open. If you enable this cron job, you will not need to collect and upload logs to Cloudbian Support yourself when you open support cases. (Note that in HyperStore 7.5.x this feature was enabled/disabled by a <i>log-collect_enable</i> setting. Now it is enabled/disabled by the setting <i>job-s.system.onDemandLogCollector.enabled</i>)
phoneHome	13 2 * * * (2:13am UTC daily)	This cron job uploads a daily system diagnostics file to a configurable S3 URI, if the HyperStore "Smart Support" feature (also known as "Phone Home") is enabled. By default Smart Support is enabled and the S3 URI is the S3 URI of Cloudbian Support. For more information see "Smart Support and Diagnostics Feature Overview" (page 117). Note The upload will occur within an hour of the time specified in the crontab. A random wait time is built into the upload process so that not all Cloudbian customer environments are uploading to Cloudbian Support at the same time.
searchSync	0 * * * * (every hour)	If you are using the HyperStore Search Service, this hourly cron job triggers the following actions in the HyperStore Search Service: • Create new search indices for buckets that were created within the past hour and that use a search-enabled storage policy. • Update search indices for all buckets that use a search-enabled storage policy, to reflect changes to bucket content during the past hour -- such as object uploads or object deletions. • Delete search indices for buckets that were deleted during the past hour and that had used a search-enabled storage policy. Until the cron job runs, these search index update tasks are queued in the Metadata DB. Any tasks that fail when this cron job runs are retried at the next running of the cron job.
usageRepairUsers	0 */12 * * (every 12 hours)	This cron job repairs Redis QoS 'stored bytes' and 'stored object' counts for up to 1000 recently active users.

Note A small number of HyperStore system cron jobs are hard-coded (are not user-configurable) and do not appear in this table.

7.10.2. Cassandra Data Compaction

The Metadata DB is built on Cassandra. Cassandra stores data to disk in the form of "Sorted String Tables" (SSTables), a type of append-only commit log. During Cassandra operations, many SSTables may be written to disk for each column family. It's important that SSTables be compacted regularly to minimize the amount of disk space that they use. Compaction merges multiple SSTables into one.

Cassandra regularly implements a "minor" compaction process that occurs **automatically**. This automatic compaction process is sufficient in the context of the HyperStore system. For the HyperStore system, **you do not need perform "major" compactions** using the *nodetool* utility.

You can monitor the compaction process on a node by using JConsole or another JMX client to connect to the Cassandra service's JMX listening port (7199 by default). By accessing the *CompactionManagerMBean* through the JMX console, you can check the progress of any compactions that are currently executing, and also check on the number of completed and pending compactions.

This page left intentionally blank

Chapter 8. Disk Operations

8.1. Disabling a HyperStore Data Disk

Subjects covered in this section:

- *Introduction (immediately below)*
- ["The Impact of Disabling a Disk" \(page 639\)](#)
- ["Disabling a Disk" \(page 639\)](#)

HyperStore supports a method for **temporarily disabling a HyperStore data disk drive** so that you can perform planned maintenance such as a firmware upgrade.

Note that you **cannot disable a disk that is the sole available data disk on the node** -- that is, you cannot disable a data disk if all the other data disks on the node are already disabled or are in [stop-write condition](#).

Note If you are **replacing** a disk, follow the instructions for ["Replacing a HyperStore Data Disk" \(page 642\)](#) rather than the instructions below.

8.1.1. The Impact of Disabling a Disk

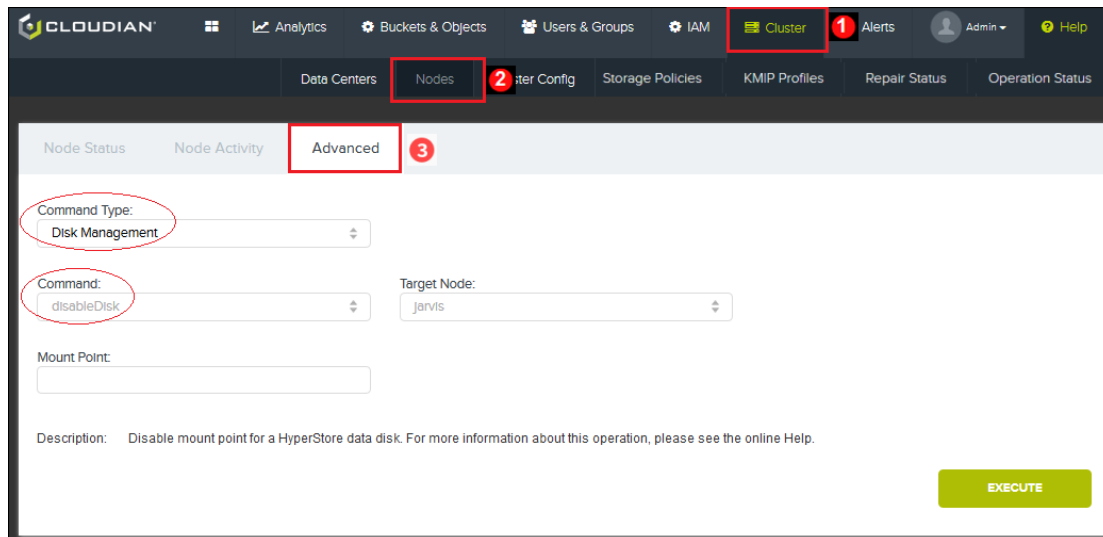
When you execute the HyperStore *disableDisk* function, the system automatically does the following:

- Unmounts the disk's file system, comments out its entry in */etc/fstab*, and marks the disk as unavailable for HyperStore reads and writes.
- Moves the disk's assigned storage tokens to the remaining HyperStore data disks on the same host, in a way that's approximately balanced across those disks. This is a temporary migration of tokens which will be reversed when you later re-enable or replace the disk. While the tokens are on the other disks, writes of new or updated S3 object data that would have gone to the disabled disk will go to the other disks on the host instead.

The existing object data on the disabled disk is **not** recreated on the other disks and therefore that data will be unreadable on the host. Whether the system as a whole can still provide S3 clients with read access to the affected S3 objects depends on the availability of other replicas or erasure coded fragments for those objects, elsewhere within the cluster.

8.1.2. Disabling a Disk

1. In the CMC's **Node Advanced** page, select command type "Disk Management" and then select the "disableDisk" command.



2. Choose the Target Node (the node on which the disk resides), and enter the Mount Point of the disk that you want to disable (for example `/cloudian6`).
3. Click **Execute**.

After the `disableDisk` operation completes, go to the CMC's [Node Status](#) page. In the "Disk Detail Info" section, the device that you disabled should now have this red status icon (indicating that it's disabled and its tokens have been migrated to other disks on the same host):



Later, after completing your maintenance work on the disk, follow the instructions for ["Enabling a HyperStore Data Disk"](#) (page 640). When you re-enable the disk, the tokens that had been moved away from the disk will be moved back to it.

8.2. Enabling a HyperStore Data Disk

Subjects covered in this section:

- *Introduction (immediately below)*
- ["The Impact of Enabling a Disk"](#) (page 641)
- ["Enabling a Disabled Disk"](#) (page 641)

HyperStore supports a method for **enabling an existing HyperStore data disk that is currently disabled**. You can tell that a disk is disabled by viewing its status in the ["Disk Detail Info"](#) section of the CMC's **Node Status** page. A disk can go into a disabled state either because you disabled it by using the HyperStore `disableDisk` function (as described in ["Disabling a HyperStore Data Disk"](#) (page 639)) or because the HyperStore system automatically disabled it in response to disk errors (as described in ["Automated Disk Usage Management Feature Overview"](#) (page 124)).

You can enable a disk if you know that the disk problem was only temporary and that the disk is still healthy enough to use.

Note In the case of a multipath (MPIO) disk you can re-enable the disk if at least one path to the disk is back up.

Note If you are **replacing** a disk, follow the instructions for "[Replacing a HyperStore Data Disk](#)" (page 642) rather than the instructions below.

8.2.1. The Impact of Enabling a Disk

When you execute the HyperStore *enableDisk* function, the system automatically does the following:

- Remounts the disk (using the same mount point that the disk previously had), uncomments its entry in */etc/fstab*, and marks the disk as available for HyperStore reads and writes.
- Moves back to the disk the same set of storage tokens that were automatically moved away from the disk when it was disabled.

After a disk is re-enabled in this way, writes of new or updated S3 object data associated with the returned tokens will go to the re-enabled disk. And the existing object data that was already on the disk will once again be readable. Meanwhile object data that was written to the affected token ranges while the disk was disabled — while the tokens were temporarily re-assigned to other disks on the host — will remain on those other disks and will be readable from those disks. That data will not be moved to the re-enabled disk.

Note For information on how HyperStore tracks token location over time so that objects can be written to and read from the correct disks, see "[Dynamic Object Routing](#)" (page 127).

8.2.2. Enabling a Disabled Disk

1. In the CMC's **Node Advanced** page, select command type "Disk Management" and then select the "enableDisk" command.

The screenshot shows the Cloudian CMC interface. The top navigation bar includes 'Cluster' (highlighted with a red box and a red circle with '1'), 'Alerts', 'Admin', and 'Help'. The sub-navigation bar includes 'Data Centers', 'Nodes' (highlighted with a red box and a red circle with '2'), 'Cluster Config', 'Storage Policies', 'KMIP Profiles', 'Repair Status', and 'Operation Status'. The 'Node Status' section has tabs for 'Node Status', 'Node Activity', and 'Advanced' (highlighted with a red box and a red circle with '3'). The 'Advanced' page shows a 'Command Type' dropdown set to 'Disk Management' and a 'Command' dropdown set to 'enableDisk'. The 'Target Node' dropdown is set to 'jarvis'. The 'Mount Point' field is empty. A description at the bottom reads: 'Re-enable mount point for an existing disk which is currently disabled. For more information about this operation, please see the online Help.' An 'EXECUTE' button is at the bottom right.

2. Choose the Target Node (the node on which the disk resides), and enter the Mount Point of the disk that you want to enable.

3. Click **Execute**.

After the `enableDisk` operation completes, go to the CMC's [Node Status](#) page. In the "Disk Detail Info" section, the device that you enabled should now have this green status icon (indicating that its status is OK):



If instead the disk icon is displaying in red (indicating an "Error" status), click the **Clear Error History** button. Doing so should return the disk to OK status.

Note If the CMC continues to show status information for the disk's old device address (as well as a new device address), and if clicking the **Clear Error History** button fails to clear the old information, `ssh` into the node on which you enabled the disk and run the command `systemctl restart cloudian-agent`. Then wait at least one minute, and check the CMC again. If the old information is still displaying, click the **Clear Error History** button again.

8.3. Replacing a HyperStore Data Disk

Subjects covered in this section:

- *Introduction (immediately below)*
- ["The Impact of Replacing a HyperStore Data Disk" \(page 643\)](#)
- ["Replacing a HyperStore Data Disk" \(page 643\)](#)

HyperStore supports a method for **activating a replacement HyperStore data disk and restoring data to it**. This procedure applies to either of these circumstances:

- You are replacing a disk that is currently disabled. You can tell that a disk is disabled by viewing its status in the ["Disk Detail Info"](#) section of the CMC's **Node Status** page. A disk can go into a disabled state either because you disabled it by using the HyperStore `disableDisk` function (as described in ["Disabling a HyperStore Data Disk" \(page 639\)](#)) or because the HyperStore system automatically disabled it in response to disk errors (as described in ["Automated Disk Usage Management Feature Overview" \(page 124\)](#)).
- You are replacing a disk that is not currently disabled. When you pull the disk from the host machine HyperStore will automatically disable the associated mount point, and you can proceed to replace the disk. (Note that you do not need to use the `disableDisk` function before replacing the disk, since HyperStore will disable the disk automatically.)

After you've pulled the bad disk and physically installed the replacement disk, HyperStore will take care of all the remaining set-up and restoration tasks when you follow the steps in ["Replacing a HyperStore Data Disk" \(page 643\)](#) below.

You can replace disks on multiple nodes concurrently, if there is only erasure coded data in your system. Replacing multiple disks concurrently is **not** supported in either of these circumstances:

- You have replication storage policies in your system (storage policies that use object replication rather than erasure coding). Note that in a multi-DC environment, "Replicated EC" is considered an erasure coding storage policy not a replication storage policy.

- You are using the new "hybrid" storage policies introduced in HyperStore 7.5.1 (which use replication for small objects and erasure coding for large objects).

Also, you cannot replace multiple disks on the same node concurrently. You can only replace multiple disks concurrently if the disks are on different nodes, so that on each targeted node only one disk is being replaced at a time.

8.3.1. The Impact of Replacing a HyperStore Data Disk

When you physically install a new disk and then execute the HyperStore *replaceDisk* function, the system automatically does the following:

- Creates a primary partition and an *ext4* file system on the new disk.
- Establishes appropriate permissions on the mount.
- Remounts the new disk (using the same mount point that the prior disk had), uncomments its entry in */etc/fstab*, and marks the disk as available for HyperStore reads and writes.
- Moves back to the new disk the same set of storage tokens that were automatically moved away from the prior disk when it was disabled.
- Performs a data repair for the new disk (populating the new disk with its correct inventory of object replicas and/or erasure coded object fragments).

Going forward, writes of new or updated S3 object data associated with the returned tokens will go to the new disk. Meanwhile object data that was written to the affected token ranges while the mount point was disabled — while the tokens were temporarily re-assigned to other disks on the host — will remain on those other disks and will be readable from those disks. That data will not be moved to the new disk.

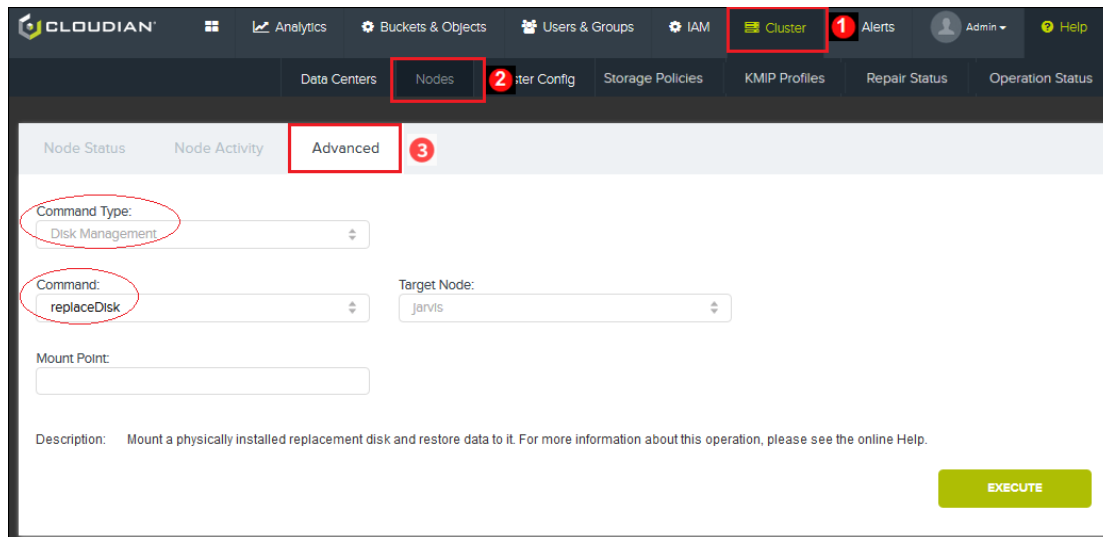
For information on how HyperStore tracks token location over time so that objects can be written to and read from the correct disks, see ["Dynamic Object Routing" \(page 127\)](#).

8.3.2. Replacing a HyperStore Data Disk

Note HyperStore supports replacing a bad disk on an existing node while you are in the midst of expanding your cluster. Wait until the ["Add Node" operation](#) completes successfully, and then you can follow the steps below either before you trigger a rebalance operation on the new node(s) or while the rebalance operation(s) are in progress. You do not need to wait until the rebalance operation(s) complete. However, see ["Extra Step After Disk Replacements Performed During Cluster Expansion" \(page 644\)](#).

After you've physically installed the replacement disk, follow these steps to activate the replacement disk and restore data to it:

1. In the CMC's **Node Advanced** page, select command type "Disk Management" and then select the "replaceDisk" command.



2. Choose the Target Node (the node on which the disk resides), and enter the Mount Point of the replacement disk. This must be the same as the mount point of the disk that you replaced.
3. Click **Execute**.

After the *replaceDisk* operation completes, go to the CMC's [Node Status](#) page. In the "Disk Detail Info" section, the replacement disk should now have this green status icon (indicating that its status is OK):



The system then automatically runs *repair* and *repairc* on the disk mount point. You can monitor the repair progress in the [Operation Status](#) page.

Note that:

- If in the Node Status page the disk icon is displaying in red (indicating an "Error" status), click the **Clear Error History** button. Doing so should return the disk to OK status.
- If the CMC continues to show status information for the old disk (as well as the new disk), and if clicking the **Clear Error History** button fails to clear the old information, *ssh* into the node on which you replaced the disk and run the command *systemctl restart clouidian-agent*. Then wait at least one minute, and check the CMC again. If the old information is still displaying, click the **Clear Error History** button again.

8.3.2.1. Extra Step After Disk Replacements Performed During Cluster Expansion

If you executed the *replaceDisk* during the rebalance operations associated with a cluster expansion (or after the Add Node[s] operation completed and before you kicked off the rebalance operations), then after the *replaceDisk* operation and **all rebalance operations have completed**, run [hssstool repairc](#) on any one of the new nodes. This will replace any EC data that is supposed to be on the new nodes but that may be missing because of the disk having gone bad on an existing node during the cluster expansion.

If you added new nodes to multiple data centers, run [hssstool repairc](#) on one of the new nodes in each data center.

8.4. Replacing a Cassandra Disk

If you need to replace a disk that stores Cassandra (the Metadata DB) and the OS, please consult with Cloudian Support.

8.5. Responding to Data Disks Nearing Capacity

HyperStore implements an [automatic mechanism for helping to ensure balanced disk usage](#) among the disks on a host. However, if service utilization is heavy for the size of your cluster, there may be times when one or more disks nears their capacity.

Note For guidance about HyperStore capacity management and cluster resizing, see ["Capacity Monitoring and Expansion"](#) (page 656).

If an individual disk or a node as a whole is running low on available capacity, HyperStore alerts administrators:

- Alerts are triggered if an individual disk drops below 15% available capacity or if a node as a whole drops below 10% available capacity. When such alerts are triggered, they appear in the CMC's [Node Status](#) page and [Alerts](#) page as well as being sent to the system administrator email address(es). Optionally, alerts can also be transmitted as SNMP traps. Alert thresholds and options are configurable in the [Alerts Rules](#) page.
- If a node as a whole reaches 80% utilization of its data disk capacity, a Warning is display on the CMC's [Dashboard](#) page.

If a **HyperStore data disk** (a disk storing S3 object data) is nearing capacity, the first two things to try are:

- Use [hsstool cleanup](#) (and [hsstool cleanupec](#) if you use erasure coding in your system) on the node to clear it of any data that's no longer supposed to be there. Such "garbage data" may be present if, for example, S3 objects have been deleted from the system as a whole but the deletion operations on the node in question failed.
- Delete S3 objects. Note that the associated files will not be deleted from the disk immediately since HyperStore uses batch processing for purging of S3 object data. The batch purging processing occurs hourly on each node.

Note For additional guidance on removing data to free up disk space, consult with Cloudian Support.

If these measures do not free up sufficient disk space, the solution is to increase system capacity by adding one or more new nodes to your cluster. For the procedure see ["Adding Nodes"](#) (page 579). When you add a node, a portion of the data on your existing nodes is copied to the new node and then (when you subsequently run a cleanup operation) deleted from the existing nodes — thereby freeing up space on the existing nodes. The degree to which space will be freed up on existing nodes depends on the number of new nodes that you add in proportion to the size of your existing cluster — for example, adding two nodes to a four node cluster would free up a larger percentage of the existing nodes' disk space than would adding two nodes to a twenty node cluster.

For information about managing a **Cassandra data disk** (a disk storing system and object metadata stored in the Metadata DB) that is nearing capacity see ["Responding to Cassandra Disks Nearing Capacity"](#) (page 646).

8.6. Responding to Cassandra Disks Nearing Capacity

If a Cassandra (Metadata DB) data drive is nearing capacity, the first two things to try are:

- Use [hsstool cleanup](#) on the host node, using the *allkeyspaces* option. This will clear any Cassandra data that is no longer supposed to be on the node.
- Selectively delete Cassandra data. To do this, consult with Cloudbian Support.

If these measures do not free up sufficient space, the solution is to increase system capacity by adding one or more new nodes. For the procedure see ["Adding Nodes"](#) (page 579). When you add a node, a portion of the Cassandra data on your existing nodes is copied to the new node and then (when you subsequently run a cleanup operation) deleted from the existing nodes — thereby freeing up space on the existing nodes.

8.7. Adding More Disks To a Node is Not Supported

HyperStore does not support adding disks to an existing node within the cluster. To increase cluster storage capacity, the only supported method is to add more nodes as described in ["Adding Nodes"](#) (page 579).

Note For guidance about HyperStore capacity management and cluster resizing, see ["Capacity Monitoring and Expansion"](#) (page 656).

Chapter 9. File Service Operations

This section describes operations applicable to the HyperStore File Service:

- ["Replacing a Node in a Three-Node File Service Pool"](#) (page 647)
- ["Backing Up and Restoring a Single-Node File Service Pool"](#) (page 649)

Note For an introduction to the HyperStore File Service, see ["File Services Feature Overview"](#) (page 238). For information about installing the HyperStore File Service, see "Installing the HyperStore File Service" in the *Cloudian HyperStore Installation Guide*. For information about upgrading the HyperStore File Service to version 8.2 see ["Upgrading the HyperStore File Service"](#) (page 102).

9.1. Replacing a Node in a Three-Node File Service Pool

The following procedure is for replacing one of the nodes in a three-node File Service pool.

Note Do not use this procedure to replace a node in a single-node File Service pool. For that operation see ["Restoring a Replacement Node"](#) (page 650).

Prerequisites:

- The new node must have:
 - The same resource specs and disk configuration as the node that you are replacing.
 - A different hostname and different IP address than the node that you are replacing.
- HyperStore users who own file shares published to this File Service pool should [close the shares](#) before you perform this procedure. (If your HyperStore system has multiple File Service pools, users do not need to close shares that are published only to other File Service pools and not to the pool for which you are replacing a node.)

To replace a node in a three-node File Service pool:

1. Log in to the HyperStore Config Controller node and change into the installation staging directory (`/opt/cloudian-staging/8.2`).
2. Make a backup copy of the File Service pool's topology file and then open the original topology file in a text editor. For example:

```
cp file-pool-1.yaml file-pool-1.yaml.orig
vi file-pool-1.yaml
```

3. In the topology file's "nodes:" section:
 - Find the *name* (hostname) of the node that you are replacing and change it to the hostname of the new node.
 - Also change the *ipAddress* of the node that you are replacing to the IP address of the new node.Before and after, if you are replacing (for example) node *filehost3* with node *filehost4*:

Before:

```
- name: filehost3
  memberOf: file-pool-1
  datacenter: DC-1
  ipAddress: 10.1.254.123
```

After:

```
- name: filehost4
  memberOf: file-pool-1
  datacenter: DC-1
  ipAddress: 10.1.254.124
```

- Don't make any other changes to the file.

Save your changes and exit the text editor.

4. Run this `hsctl` command to execute the node replacement:

```
hsctl file node replace -t <topology-file> <old-node's-hostname> <new-node's-hostname>
```

For example:

```
hsctl file node replace -t file-pool-1.yaml filehost3 filehost4
```

Once the replace operation has completed successfully, log in to the new File Service node as `root` and run this `kubectl` command to confirm that all the Kubernetes pods on the node are running and ready:

```
kubectl get pods --all-namespaces --field-selector spec.nodeName=<NODE-NAME>,status.phase!=Succeeded -o custom-columns=POD:metadata.name,STATUS:status.phase,CONTAINERS:.spec.containers[*].name,READY:status.containerStatuses[*].ready
```

Note This is a one-line command. Replace `<NODE-NAME>` with the hostname of the new File Service node.

The command response should look like this, with the STATUS column showing "Running" for all pods and the Ready column showing "true" for all pods and containers:

POD	STATUS	CONTAINERS	READY
cert-manager-webhook-556f979d7f-4m179	Running	cert-manager	true
cloudian-hf-controller-manager-6459b748bf-6qbh6	Running	kube-rbac-proxy,manager	true,true
hf-frontend-1-1-79c889c58f-wtqfj	Running	samba,ganeshha,grafana-agent	true,true,true
hf-gui-6f9fb56d55-g9q wz	Running	hyperfile-gui	true
hf-volume-1-1-1-2	Running	hyperfile-1	true
hyperfile-promtail-f87f75b-9rl76	Running	promtail	true
k8ssandra-cass-operator-6c44c8ff99-6djl b	Running	cass-operator	true
k8ssandracluster-dc1-rack1-sts-1	Running	cassandra,server-system-logger	true,true
loki-0	Running	loki	true
ingress-nginx-controller-7b6bfd5dcc-6sxs d	Running	controller	true
kadalu-csi-nodeplugin-65rt5	Running	csi-node-driver-registrar,kadalu-nodeplugin,kadalu-logging	true,true,true
cilium-ggl59	Running	cilium-agent	true
cilium-operator-649f49c889-d6fjv	Running	cilium-operator	true
coredns-d76bd69b-xpj jp	Running	coredns	true
kube-vip-ds-7glx9	Running	kube-vip	true
local-path-provisioner-6c79684f77-j78dj	Running	local-path-provisioner	true
metallb-speaker-ppxhn	Running	speaker	true
hf-rabbitmq-server-1	Running	rabbitmq	true
rabbitmq-rabbitmq-cluster-operator-cc45c955f-z74wq	Running	rabbitmq-cluster-operator	true

Once you've completed the node replacement and confirmed that the new node's pods are running and ready, you can have HyperStore users [reopen the file shares](#) that are published to this File Service pool.

9.2. Backing Up and Restoring a Single-Node File Service Pool

File Service pools consist of either one node or three nodes. In a three-node File Service pool, if a node fails the pool will continue to deliver SMB and/or NFS services since the other two nodes in the pool remain live. The same is not true of a single-node File Service pool -- a node failure will result in service disruption. To minimize service disruption for a single-node File Service pool, a backup-and-restore ability is provided.

For a single-node File Service pool, backups occur automatically once a day (at 03:00 UTC). Specifically, the File Service node's Kubernetes resources and the node's volume metadata (stored in Cassandra) are backed up to dedicated S3 buckets that the system automatically creates and maintains in the HyperStore object storage layer.

Using the [hscctl command line tool](#), you have the ability to:

- Create on-demand backups (in addition to the automatically executed daily backups)
- List all available backups
- Restore an existing node to a specified backup (roll-back)
- Restore a replacement node to a specified backup

Note These operations are for single-node File Service pools only. They are not applicable to three-node File Service pools.

9.2.1. Creating an On-Demand Backup

To create a backup of a File Service node, on the HyperStore Config Controller node run this *hscctl* command:

```
hscctl file backup create [-n <file-pool-name>]
```

If you have only one File Service pool in your HyperStore system you can omit the *-n <file pool name>* argument. If you have multiple pools in your system, include the argument to indicate the pool for which you want to create a backup.

For example:

```
# hscctl file backup create -n file-pool-1
backup process started successfully
```

9.2.2. Listing Available Backups

To list available File Service node backups, on the HyperStore Config Controller node run this *hscctl* command:

```
hscctl file backup list [-n <file pool name>]
```

If you have only one File Service pool in your HyperStore system you can omit the *-n <file pool name>* argument. If you have multiple pools in your system, include the argument to indicate the pool for which you want to list available backups.

For example:

```
# hscctl file backup list -n file-pool-1
```

NAME	START TIME	FINISH TIME	STATE	MESSAGE
backup-202403211312	21 Mar 24 09:12 EDT	21 Mar 24 09:13 EDT	COMPLETED	
...				
...				

Note Backups are named as *backup-**<YYYYMMDDhhmm>*** where **<YYYYMMDDhhmm>** is the backup start time in UTC. The list of backups will be chronologically sorted by start time, with the newest backup listed last.

9.2.3. Rolling Back an Existing Node to a Previous State

To restore (roll back) an existing single-node File Service pool to a previous state captured in a backup:

1. On the HyperStore Config Controller node run this *hscctl* command to retrieve a list of available backups for the pool:

```
hscctl file backup list -n <file-pool-name>
```

From the list, identify the backup to which you want to restore the pool. Backups are listed chronologically based on backup start time in UTC.

2. Still on the HyperStore Config Controller node, run this *hscctl* command to restore the existing single-node File Service pool to a specified backup:

```
hscctl file backup restore <backup-name> -n <file-pool-name>
```

For example:

```
# hscctl file backup restore backup-202403280300 -n file-pool-1
Starting restore procedure
Deleting required secrets
Restoring Velero from backup
Restoring Cassandra using Medusa
Waiting for Cassandra to be restored
Restarting HyperFile API
Sending restore request to HyperFile
Backup restore completed successful
```

9.2.4. Restoring a Replacement Node

If the need arises you can decommission the existing node in a single-node File Service pool and replace it with a new node by following the procedure below. The new node should have the same resource specs and disk configuration as the node that you are replacing. (For background information about preparing a new node to host the HyperStore File Service see "File Service Host Requirements" in the *Cloudian HyperStore Installation Guide*.) You will need the HyperStore File Service version 8.2 product package to complete this procedure.

Note Before you start this procedure, have any HyperStore users who have published shares to this single-node File Service pool close all of those shares. They can do this in the CMC. If this cannot be done due to problems with the File Service node that you are going to replace, this preliminary step can be skipped.

1. On the HyperStore Config Controller node run the following *hscctl* command to uninstall HyperStore File Services from the node that you are decommissioning. By including *--skip-backup-delete* you are preserving the node's backups, which are stored in a system-managed bucket in the HyperStore object storage layer.

```
hscctl file uninstall -n <file-pool-name> --skip-backup-delete
```

- On the HyperStore Config Controller node, change into the installation staging directory (*/opt/cloudian-staging/8.1*) and then launch the *system_setup.sh* tool:

```
./system_setup.sh
```

- From the tool's main menu choose **4) Setup Cluster Topology**. Then from the Topology Editor menu choose **c) Load existing topology file**, and load the *<file pool name>.yaml* file for the single-node file pool for which you are replacing the node. Then choose **e) Edit topology**. When editing the topology, **do not change the "File Service pool name"** or the region, data center, or rack. Edit the other entries as needed (for example, the hostname and the host's IP address). Save the topology when you're done, and then exit the *system_setup.sh* tool.
- Put the HyperStore File Service version 8.2 product package in the installation staging directory (if you don't already have it there). Then install the HyperStore File Service on the new node with this command:

```
hsctl file install <package-name> -t <file pool name>.yaml
```

Wait for the installation to indicate that it has successfully completed, then proceed to Step 5.

- Log into the new HyperStore File Service node -- the node on which you installed File Service software in Step 4 -- as *root* and then run these two commands in this order (these commands confirm the accessibility of the backups in the system-managed backup bucket and import them as Kubernetes resources so they can subsequently be managed via the *hsctl* tool):

```
kubectl get backups -n velero
```

```
velero restore create restore-backup-list --from-backup $(kubectl get backups -n velero -o=jsonpath='{.items[-1].metadata.name}')
```

Note The *velero restore* command above is a one-line command.

- Back on the HyperStore Config Controller node, run this *hsctl* command to get the list of backups that had been completed for the single-node File Service pool for which you are performing the node replacement:

```
hsctl file backup list -n <file-pool-name>
```

Copy or make note of the name of the most recent backup in the list (the list is chronologically sorted, so the most recent backup will be the last backup in the list).

- Still on the HyperStore Config Controller node, run this *hsctl* command to restore the File Service pool to the state captured in the most recent backup that you identified in Step 6:

```
hsctl file backup restore <backup-name> -n <file-pool-name>
```

Messages in the terminal will indicate when the backup restore process has completed.

Note that **after the restore process**:

- All file shares published to this File Service pool will be in a closed state.
- HyperStore users who own file shares published to this File Service pool must:
 - [Re-enable bi-modal support](#) for any shares that they want to support bi-modal access (this share-specific configuration setting gets dropped during the restore process). In general bi-modal access is recommended in this scenario since it will result in shares getting updated with any content changes that were made on the S3 side during the node decommissioning and replacement process.

- Use the CMC's **Global File Tenancy Settings** page to [rejoin the Active Directory domain](#) associated with this File Service pool, if the pool had been joined to an Active Directory domain prior to the node replacement operation.
- [Reopen the file shares](#) that have been published to this File Service pool.

IMPORTANT ! After node replacement, front-end IP addresses for file sharing will be assigned to these HyperStore users in the order in which they re-open shares (from the front-end IP range specified in the file pool's topology). Thus a HyperStore user who publishes shares to this File Service pool may end up with a different front-end IP address for this pool than he or she had previously. These HyperStore users must [check their front-end IP address](#) for this File Service pool after opening one or more shares that they publish to this pool; and if their front-end IP address has changed they will need to alert this File Service pool's end users to this change.

Chapter 10. Search Service Operations

This section describes operations applicable to the HyperStore Search Service:

- ["Check Search Service Status"](#) (page 653)
- ["Manage Search Indexes"](#) (page 653)
- ["Manage Search Snapshots"](#) (page 653)

Note For an introduction to the HyperStore Search Service, see ["Object Metadata Based Search"](#) (page 233) and ["Preparing the Metadata Search Feature"](#) (page 236).

For all of these operations, use the `hsctl` command line tool on the HyperStore Config controller node.

10.1. Check Search Service Status

To check basic Search Service status information, including whether the Search Service is installed in your HyperStore system or not:

```
# hsctl search status
```

10.2. Manage Search Indexes

Get a list of search indexes (there should be one index per search-enabled bucket):

```
# hsctl search index list
```

Get the current content of a specified search index:

```
# hsctl search index get <index-name>
```

Delete a specified search index:

```
# hsctl search index delete <index-name>
```

Delete **all** non-protected search indexes (use only in consultation with Clouddian Support):

```
# hsctl search index delete ALL --force
```

10.3. Manage Search Snapshots

A snapshot is a point-in-time copy of Search Service data including all indexes.

Create a repository in which to store Search Service snapshots:

```
# hsctl search snapshot repository create
```

Create a Search Service snapshot:

```
# hsctl search snapshot create <snapshot-name>
```

List all existing Search Service snapshots:

```
# hsctl search snapshot list
```

Delete a specified Search Service snapshot:

```
# hsctl search snapshot delete <snapshot-name>
```

Restore all indexes in the Search Service to a specified snapshot (**use only in consultation with Clouddian Support**):

```
# hsctl search snapshot restore <snapshot-name>
```

Delete the snapshot repository and all snapshots currently in the repository:

```
# hsctl search snapshot repository delete
```

Chapter 11. Monitoring

11.1. Clodian HyperIQ

Clodian HyperIQ is a solution for dynamic visualization and analysis of HyperStore system monitoring data and S3 service usage data. HyperIQ is a separate product available from Clodian that deploys as virtual appliance on VMware or VirtualBox and integrates with your existing HyperStore system. For more information about HyperIQ contact your Clodian representative.



11.2. Using the CMC to Monitor HyperStore

The Clodian Management Console (CMC) provides extensive support for monitoring the health and performance of your HyperStore system. The CMC also supports a configurable mechanism for alerting administrators when system events occur.

The table below shows the system monitoring and alert management tasks that are supported by the CMC.

Category	Tasks	CMC Page
System Monitoring	Check high level status information for each service region, including current storage capacity usage and remaining available capacity, project capacity usage for the next 120 days, recent S3 transaction performance, and whether there are any system alerts	Dashboard
	Check the remaining available storage capacity in each service region, as well as capacity remaining in each data center and on each	Capacity Explorer

Category	Tasks	CMC Page
	node	
	Check how your storage capacity usage has changed over the past 30 days, per service region	Cluster Usage
	For each data center, see which nodes have any alerts and whether any HyperStore services are down on any node	Data Centers
	Check a specific node's storage capacity usage and remaining available capacity, CPU and memory usage, recent S3 transaction performance, and HyperStore services status. Also check capacity usage and status information for individual disks on a node.	Node Status
	Check a specific node's performance trends over the past 30 days, for metrics such as CPU utilization, disk read and write throughput, and S3 transactions per second.	Node Activity
System Alerts Management	Configure system alert rules, for having the system automatically notify administrators regarding system events. Also view pre-configured alert rules that come with the system.	Alert Rules
	Review and acknowledge alerts generated by any node in the system	Alerts
	Review and acknowledge alerts generated by a specific node	Node Status

11.3. Capacity Monitoring and Expansion

Subjects covered in this section:

- *Introduction (immediately below)*
- **"Monitoring Cluster Storage Capacity Utilization" (page 657)**
- **"Adding Nodes to Your System" (page 658)**

HyperStore is horizontally scalable, allowing you to gain additional storage and request processing capacity by adding more nodes to your cluster. When you add new nodes to your cluster, the storage capacity associated with the new nodes becomes immediately available to the system. However, the automated processes that re-distribute data from existing nodes to newly added ones -- thereby reducing storage capacity usage on the existing nodes -- may take several days or more to complete, depending on factors such as data volume and network bandwidth. Therefore it's important to closely monitor your current and projected system capacity usage, plan ahead for needed cluster expansions, and implement such expansions well before you've filled your current capacity.

Use the CMC Dashboard to monitor your system's current and projected storage utilization level (for more information see **"Monitoring Cluster Storage Capacity Utilization" (page 657)**, further below). As best practices for cluster expansion timing, Cloudian recommends the following:

- **Start expansion planning and preparation when either of the following occur (whichever occurs first):**
 - The Dashboard shows your **utilization of system storage capacity has reached 70%**.

- The Dashboard shows your **utilization of system storage capacity is projected to reach 90% within 120 days**. If your system has a high rate of ingest relative to capacity, this projection may occur even if your current usage has not yet reached 70%.

When planning your expansion keep in mind that:

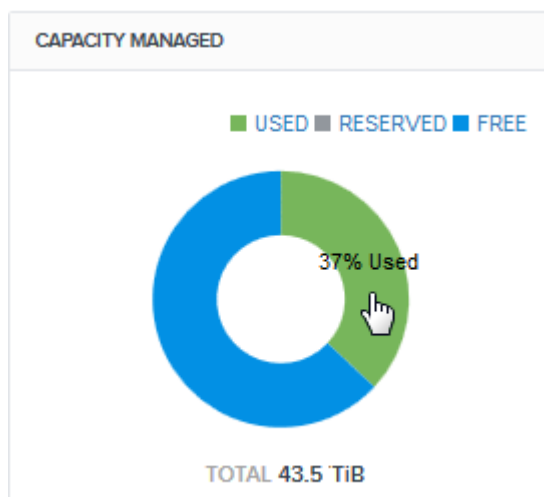
- The minimum unit of expansion is a node. HyperStore does not support adding disks to existing nodes.
- You need to allow time to acquire host machines and prepare them for being added to your cluster.
- Preferably, cluster expansions should be substantial enough that the expanded cluster will allow you to meet your projected storage needs for at least an additional six months after the expansion. In this way you can avoid having to frequently add nodes to your system.
- Cloudian Support is available to review and provide feedback on your expansion plan.
- **Execute your expansion when either of the following occur (whichever occurs first):**
 - The Dashboard shows your **utilization of system storage capacity has reached 80%**.
 - The Dashboard displays a Warning that your **utilization of system storage capacity is projected to reach 90% within 90 days**. If your system has a high rate of ingest relative to capacity, this projection may occur even if your current usage has not yet reached 80%.

IMPORTANT ! Each HyperStore node is designed to [reject new writes if it reaches 90% storage capacity utilization](#). Allowing your system to surpass 80% capacity utilization poses the risk of having to rush into an urgent cluster expansion operation.

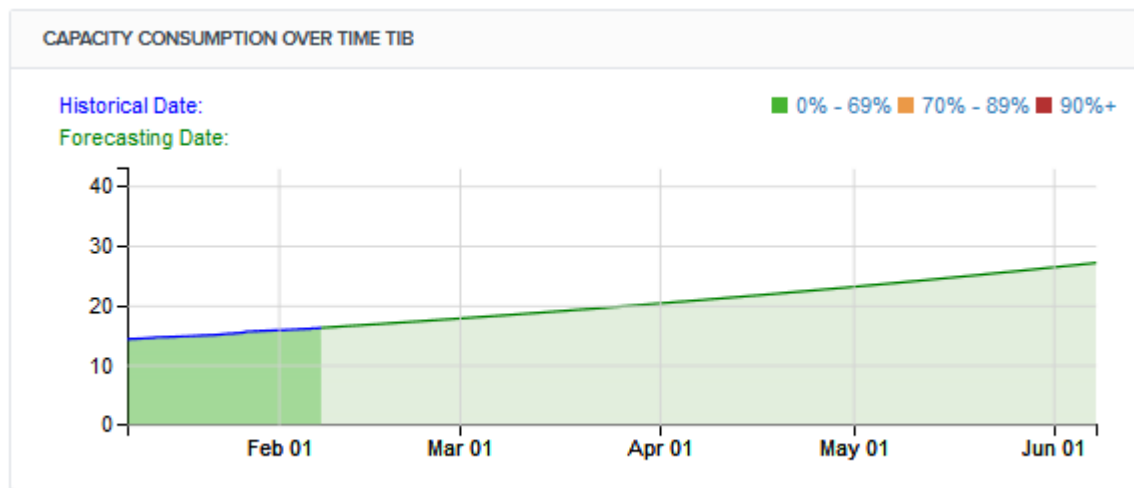
11.3.1. Monitoring Cluster Storage Capacity Utilization

HyperStore makes it easy to regularly monitor your system storage capacity utilization. In the CMC Dashboard you can view:

- Current storage capacity utilization.



- Projected storage capacity utilization over the next 120 days.



In both the current capacity utilization graphic and the projected utilization graphic, color-coding is used to highlight utilization levels of 70% or higher and 90% or higher.

The Dashboard also displays:

- A Warning message if the cluster is projected to reach 90% storage utilization in 90 days or less
- A Critical message if the cluster has reached 90% storage utilization

For more detail on Dashboard functionality and metrics see [Dashboard](#).

In the CMC's [Capacity Explorer](#) page you can view your system's remaining free storage capacity broken down by service region (cluster), data center, and node. If less than 30% storage space remains at any one of these levels -- that is, if more than 70% of capacity is utilized in a given node, data center, or region -- this is highlighted in the interface.

In the CMC's [Node Status](#) page you can view each node's storage capacity utilization as well as the utilization level for each disk on each node.

11.3.2. Adding Nodes to Your System

To add nodes to an existing HyperStore data center, follow the documented procedure ["Adding Nodes"](#) (page 579).

To add a new data center (with new nodes) to an existing HyperStore service region, follow the documented procedure ["Adding a Data Center"](#) (page 590).

To add a new service region (with new nodes) to an existing HyperStore system, follow the documented procedure ["Adding a Region"](#) (page 597).

IMPORTANT ! With the addition of a new DC or service region you will need to create new [storage policies](#) that utilize the new DC or region. **Adding a new DC or region does not create additional storage capacity for existing buckets that use existing storage policies.** Only new buckets that utilize the new storage policies will make use of the additional storage capacity created by adding a new DC or region. In the current HyperStore version, you cannot revise an existing storage policy or reassign a new storage policy to an existing bucket.

To create additional storage capacity for your existing storage policies you must add nodes to your existing data center(s).

11.4. Using the Admin API to Monitor HyperStore

The HyperStore Admin API supports methods for monitoring HyperStore health and performance. The CMC invokes these Admin API methods in implementing the CMC's system monitoring functions. If you wish you can invoke these Admin API methods directly, through a client application of your own making or through third party command line tools that enable you to construct HTTP requests.

For more information see "monitor" section of the *Cloudian HyperStore Admin API Reference*.

11.5. Using Linux Utilities to Monitor System Resource Usage

The Linux OS on which the HyperStore system runs includes several useful commands for checking on system resource utilization on individual host machines. For example, the **top** command returns a summary of host-wide resource utilization as well as a breakdown of resource usage per process. Using **top -c** (which returns more information in the "Command" column than **top** alone) makes it easier to distinguish between the several Java-based HyperStore processes that will be running on each host — including Cassandra, the S3 Service (*cloudian_s3*), the Admin Service (*cloudian_admin*), the HyperStore Service (*storage_s3*), and the CMC (*tom-cat*).

Important statistics are the memory usage and CPU usage:

- VIRT: Virtual memory.
- RES: Resident memory. VIRT minus RES is the amount of memory swapped out.
- %CPU: Percentage of CPU used

Other useful Linux utilities for monitoring system resource usage include:

- **vmstat**
- **iostat**
- **dstat**

11.6. Using JMX to Monitor HyperStore Services

IMPORTANT ! Cloudian recommends that you **do not use JMX for monitoring a HyperStore production system**, as it may negatively impact performance (particularly if you run JConsole on one of your production nodes). However, JMX may be useful for monitoring a HyperStore testing or evaluation system.

The S3 Service, Admin Service, HyperStore Service, and Cassandra Service support monitoring via Java Management Extensions (JMX). You can access JMX statistics using the graphical JMX client JConsole, which comes with your Java platform. By default the full path to the JConsole executable is */usr/java/latest/bin/jconsole*.

After launching JConsole, in the JConsole GUI specify the `<host>:<jmx-port>` that you want to connect to. Each of the HyperStore system's Java-based services has a different JMX listening port as indicated in the sections that follow. **The statistics that you view will be only for the particular node to which you are connected via JConsole.**

Note

* By default a JConsole connection does not require a user name and password, so in the JConsole GUI these fields can be left empty. For general information about using JConsole, including password protection and security options, see the JConsole Help.

* This section on JMX statistics presumes that you are using JConsole, but there are other JMX clients available. Your HyperStore system comes with two command-line JMX clients — `cmdline-jmxclient` and `jmxterm` — which are in the `/opt/cloudian/tools` directory.

S3 Service JMX Statistics

Default JMX port: 19080

For the S3 Service, these categories of JMX statistics are supported:

S3 operation timing and rate stats

In JConsole's MBeans tab, timing performance statistics for S3 operations are available under the *metrics* MBean. Under *metrics* there is *com.cloudian.s3.stats.<operation>*, where *<operation>* is an S3 API operation such as *putObject*, *getObject*, *deleteObject*, *getBucket*, and so on. For each operation type, under *Attributes* there is a set of timing statistics including:

- Count — The total number of executions that were timed.
- Max — The maximum value in milliseconds of the logged execution times.
- Mean — The mean execution time, in milliseconds.
- Min — The minimum value in milliseconds of the logged execution times.
- StdDev — The standard deviation, in milliseconds.

For each operation type there are also rate stats including:

- MeanRate — Average transactions-per-second (TPS) since last restart.
- FifteenMinuteRate — 15 minute exponentially weighted moving average rate for TPS.

The statistics are initialized at each restart of the S3 Service. Statistics will only be available for operations that have been performed since the last S3 Service restart — for example, if no *deleteObject* operations have been performed since the last restart, then no *deleteObject* statistics will be available.

Note These timing and rates stats are implemented with the Metrics Core library.

S3 operation success stats

In JConsole's MBeans tab, success rate statistics for S3 operations are available under *com.gemini.cloudian.s3 → Accounting → Success Rate → Attributes*. For each S3 operation, the success rate is expressed as a double between 0 and 1 indicating the percentage of successful processing for that S3 operation. The success rate statistics are cumulative since the last start of the S3 Service on the node to which you are connected.

For example, the statistic "DeleteBucket" would have a value such as 1.0 or 0.92, indicating a 100% or 92% success rate for S3 "DELETE Bucket" operations since the last start-up of the S3 Service.

S3 Service's Cassandra client stats

For its client interface to Cassandra, the S3 Service leverages open source Hector technology. In JConsole's MBeans tab, Hector statistics are available under `me.prettyprint.cassandra.service_<regionName> → hector → hector → Attributes`. Descriptions of these statistics are available in the [Health Check Attributes Available for Hector](#) section of the online Hector user guide.

The list of supported statistics is below.

- ExhaustedPoolNames
- KnownHosts
- NumActive
- NumBlockedThreads
- NumConnectionErrors
- NumExhaustedPools
- NumIdleConnections
- NumPoolExhaustedEventCount
- NumPools
- NumRenewedIdleConnections
- NumRenewedTooLongConnections
- ReadFail
- RecoverableErrorCount
- RecoverableLoadBalancedConnectErrors
- RecoverableTimedOutCount
- RecoverableTransportExceptionCount
- RecoverableUnavailableCount
- SkipHostSuccess
- StatisticsPerPool
- SuspendedCassandraHosts
- WriteFail

HTTP server thread pool stats

For serving HTTP requests, the S3 Service leverages open source Jetty technology. In JConsole's MBeans tab, Jetty thread pool statistics are available under `org.eclipse.jetty.util.thread → queuedthreadpool → 0 → Attributes`. The statistics available are:

- threads
- idleThreads
- queueSize

Admin Service JMX Statistics

Default JMX port: 19081

In JConsole's MBeans tab, timing performance statistics for Admin Service operations are available under the *metrics* MBean. Under *metrics* there is *com.cloudian.admin.stats.<operation>*, where *<operation>* is an S3 API operation such as *getUser*, *createUser*, and so on. For each operation type, under *Attributes* there is a set of timing statistics including:

- Count — The total number of executions that were timed.
- Max — The maximum value in milliseconds of the logged execution times.
- Mean — The mean execution time, in milliseconds.
- Min — The minimum value in milliseconds of the logged execution times.
- StdDev — The standard deviation, in milliseconds.

For each operation type there are also rate stats including:

- MeanRate — Average transactions-per-second (TPS) since last restart.
- FifteenMinuteRate — 15 minute exponentially weighted moving average rate for TPS.

The statistics are initialized at each restart of the S3 Service (the Admin Service stops and starts together with the S3 Service). Statistics will only be available for operations that have been performed since the last S3 Service restart — for example, if no *createUser* operations have been performed since the last restart, then no *createUser* statistics will be available.

Note These timing and rates stats are implemented with the Metrics Core library.

HyperStore Service JMX Statistics

Default JMX port: 19082

For the HyperStore Service, these categories of JMX statistics are supported:

HyperStore operation timing and rate stats

In JConsole's MBeans tab, timing performance statistics for HyperStore Service operations are available under the *metrics* MBean. Under *metrics* there is *com.cloudian.hybrid.stats.<operation>*, where *<operation>* is a HyperStore Service operation such as *put*, *getBlob*, *getDigest*, or *delete*. For each operation type, under *Attributes* there is a set of timing statistics including:

- Count — The total number of executions that were timed.
- Max — The maximum value in milliseconds of the logged execution times.
- Mean — The mean execution time, in milliseconds.
- Min — The minimum value in milliseconds of the logged execution times.
- StdDev — The standard deviation, in milliseconds.

For each operation type there are also rate stats including:

- MeanRate — Average transactions-per-second (TPS) since last restart.
- FifteenMinuteRate — 15 minute exponentially weighted moving average rate for TPS.

The statistics are initialized at each restart of the HyperStore Service. Statistics will only be available for operations that have been performed since the last HyperStore Service restart — for example, if no *delete* operations have been performed since the last restart, then no *delete* statistics will be available.

Note These timing and rates stats are implemented with the Metrics Core library.

HyperStore operation success stats

In JConsole's MBeans tab, HyperStore file system operations statistics are available under *com.gemini.cloudian.hybrid.server* → *HyperstoreOperationsStats* → *Attributes*. The statistics available are:

- *NumberOfSuccessfulDeleteOperations*
- *NumberOfSuccessfulReadOperations*
- *NumberOfSuccessfulWriteOperations*
- *TotalNumberOfDeleteOperations*
- *TotalNumberOfReadOperations*
- *TotalNumberOfWriteOperations*

Cassandra client stats

For its client interface to Cassandra, the HyperStore Service leverages open source Hector technology. In JConsole's MBeans tab, Hector statistics are available under *me.prettyprint.cassandra.service_Cloud-ian<regionName>* → *hector* → *hector* → *Attributes*.

The list of supported statistics is the same as indicated for the [S3 Service's Hector statistics](#). Descriptions of these statistics are available in the [Health Check Attributes Available](#) for Hector section of the online Hector user guide.

HTTP server thread pool stats

For serving HTTP requests, the HyperStore Service leverages open source Jetty technology. In JConsole's MBeans tab, Jetty thread pool statistics are available under *org.eclipse.jetty.util.thread* → *queuedthreadpool* → *0* → *Attributes*. The statistics available are:

- *threads*
- *idleThreads*
- *queueSize*

Cassandra Service (Metadata DB) JMX Statistics

Default JMX port: 7199

In JConsole's MBeans tab, under *org.apache.cassandra.metrics*, Cassandra supports a wide range of statistics.

The *Compaction* MBean exposes statistics including:

- Number of completed compactions.
- Number of pending compaction tasks.
- Progress of currently running compactions.

Note If the number of pending compaction tasks grows over time, this is an indicator of a need to increase cluster capacity.

The *ColumnFamily* MBean exposes statistics for active, pending, and completed tasks for each of Cassandra's thread pools.

Note A significant and sustained increase in the pending task counts for the Cassandra thread pools is an indicator of a need to increase cluster capacity.

The *ColumnFamily* MBean exposes individual column family statistics including:

- Memtable data size
- Number of live SSTables
- Average read latency
- Average write latency

Note A sustained increase in read and write latencies may indicate a need to increase cluster capacity.

11.7. Checking HTTP(S) Responsiveness

The HyperStore system supports a health check that lets administrators or applications -- such as a load balancer -- determine whether HTTP(S) interfaces are responsive on a particular HyperStore host. To perform the health submit an HTTP(S) **HEAD** request to:

`<hostname or IP address>:<port>/healthCheck`

If the service is up and running and listening on its assigned port, you will receive back an HTTP 200 OK status. If not, your request will time out.

This feature is supported for the following HTTP(S) based services and ports:

Service	Default HTTP Port	Default HTTPS Port
S3 Service	80	443
IAM Service	16080	16443
SQS Service	18090	18443
Admin Service	18081	19443
HyperStore Service	19090	not applicable

Note

* To do health checks against an HTTPS port, the client executing the check must support TLS. Note also that health checks against HTTPS ports are a more expensive operation (in terms of resource utilization) than health checks against HTTP ports.

* HTTP is disabled by default for the Admin Service, so that only HTTPS is supported. See the Introduction section of the *Cloudian HyperStore Admin API Reference*.

* The HTTP(S) health check of the Admin API service does not require Basic Authentication credentials. Basic Authentication is required for all other HTTP(S) requests to the Admin API, but is not required for health check requests.

* The SQS Service is disabled by default. For instructions to enable it see the introduction to the SQS section of the *Cloudian HyperStore AWS APIs Support Reference*.

The example below shows a health check of an S3 Service instance that is responsive.

```
HEAD http://192.168.2.16:80/.healthCheck
Status Code: 200 OK
Content-Length: 0
Date: Wed, 21 Aug 2024 12:51:50 GMT
Server: ClodianS3
```

In the case of health checks of the S3 Service, each health check request results in a special entry in the [S3 Request Log](#), such as in this example entry:

```
2024-08-16 15:01:33,757|127.0.0.1||healthCheck|||81|0|0|0|81|11820||200|
544cdd90-822f-1c98-b780-525400e89933|0|0|
```

11.7.1. Clodian Management Console (CMC)

To check the responsiveness of the CMC, submit an HTTPS **OPTIONS** request to the CMC login URL:

`https://<host>:8443/Cloudian/login.htm)`

Note Sending a GET or HEAD request to the CMC login URL will result in the CMC sending a *GET /group/list* call to the Admin Service which in turn sends a request to Cassandra. To avoid this, the more lightweight way to check CMC responsiveness is to send an OPTIONS request to the CMC login URL.

This page left intentionally blank

Chapter 12. Logging

12.1. HyperStore Logs

Subjects covered in this section:

- *Introduction (below)*
- ["Admin Service Logs" \(page 667\)](#)
- ["Cassandra \(Metadata DB\) Logs" \(page 668\)](#)
- ["CMC Logs" \(page 669\)](#)
- ["HyperStore Firewall Log" \(page 671\)](#)
- ["HyperStore Service Logs" \(page 671\)](#)
- ["HyperStore Shell \(HSH\) Log" \(page 676\)](#)
- ["IAM Service Logs" \(page 677\)](#)
- ["Monitoring Agent and Collector Logs" \(page 679\)](#)
- ["Phone Home \(Smart Support\) Log" \(page 680\)](#)
- ["Redis \(Credentials DB and QoS DB\) and Redis Monitor Logs" \(page 680\)](#)
- ["S3 Service Logs \(including Auto-Tiering, CRR, and WORM\)" \(page 682\)](#)
- ["Search Service Log" \(page 687\)](#)
- ["SQS Service Logs" \(page 687\)](#)

The major HyperStore services each generate their own application log. The S3 Service, Admin Service, and HyperStore Service, in addition to generating application logs, also generate transaction (request) logs.

The log descriptions in the table below indicate each log's default location, log entry format, logging level, and rotation and retention policy.

Note that:

- With the exception of Cassandra and Redis logs, all HyperStore logs are located in `/var/log/cloudian`.
- For information on configuring logging levels or logging rotation, see ["HyperStore Configuration Settings" \(page 700\)](#) and look at the settings for the service for which you want to adjust the logging configuration.
- For information on viewing logs from within the HyperStore Shell, see ["Using the HSH to View Logs" \(page 691\)](#).

12.1.1. Admin Service Logs

Admin Service application log (cloudian-admin.log)

Location	On every node, <code>/var/log/cloudian/cloudian-admin.log</code>
Log Entry Format	yyyy-mm-dd HH:mm:ss,SSS PriorityLevel[ThreadId]MessageCode ClassName:MESSAGE The MessageCode uniquely identifies the log message, for messages of level WARN or higher. For documentation of specific message codes including recommended operator action, see the "Log Message Codes" section of the CMC's online Help.

Log Entry Example	2024-05-04 15:48:03,158 ERROR[main]HS081003 CassandraProtectionPolicy:Caught: me.prettyprint.hector.api.exceptions. HectorException: [10.50.10.21(10.50.10.21):9160] All host pools marked down. Retry burden pushed out to client.
Default Logging Level	INFO
Default Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-admin.log.YYYY-MM-DD.i.gz</i> , where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i> . Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

Admin Service request log (*cloudian-admin-request-info.log*)

Location	On every node, <i>/var/log/cloudian/cloudian-admin-request-info.log</i>
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS ClientIpAddress HttpMethod Uri QueryParams DurationMicrosecs HttpStatus</pre> Note * Query parameters are not logged for requests that involve user credentials. * The request log records Admin API requests for which authentication fails, as well as requests for which the authentication succeeds. Success or failure is indicated by the <i>HttpStatus</i>.
Log Entry Example	2024-10-27 14:54:01,170 10.20.2.57 GET /group/list limit:100 188212 200
Logging Level	Not applicable
Default Rotation and Retention Policy	Rotation occurs if live file size reaches 100MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-admin-request-info.log.YYYY-MM-DD.i.gz</i> , where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i> . Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 2GB or if oldest rotated file age reaches 180 days.

12.1.2. Cassandra (Metadata DB) Logs

Cassandra application log (*system.log*)

Location	On every node, <i>var/log/cassandra/system.log</i>
Log Entry	PriorityLevel [ThreadId] Date(ISO8601) CallerFile:Line# - MESSAGE

Format	
Log Entry Example	INFO [FlushWriter:3] 2024-11-10 21:09:59,487 Memtable.java:237 - Writing Memtable-Migrations @445036697(12771/15963 serialized/live bytes, 1 ops)
Logging Level	INFO
Rotation and Retention Policy	Rotation occurs if live file size reaches 20MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>system.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>. Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 200MB or if oldest rotated file age reaches 30 days.

Cassandra request log (*cassandra-s3-tx.log*)

Location	On every node, <i>/var/log/cloudian/cassandra-s3-tx.log</i> . Note that these request logs are written on the client side as the S3 Service sends requests to Cassandra.
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS LogEntryType S3RequestId MESSAGE</pre> The LogEntryType is one of NORMAL, ERROR, or SLOW.
Log Entry Example	<pre>2024-11-22 13:06:27,192 c.d.d.c.Q.SLOW [cluster1] [/10.20.2.146:9042] Query too slow, took 7674 ms: alter table "UserData_20d784f480b0374559bdd71054bbb8c1"."CLOUDIAN_METADATA" with gc_grace_seconds=864000 and bloom_filter_fp_chance=0.1 and compaction = {'class': 'LeveledCompactionStrategy'};</pre>
Logging Level	Not applicable
Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cassandra-s3-tx.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>. Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

Note Currently **only a portion of request types from the S3 Service to Cassandra support this request logging feature**. These are request types that use a new DataStax Java driver (which supports the request logging) rather than the older Hector driver (which does not).

12.1.3. CMC Logs

CMC application log (*cloudian-ui.log*)

Location	On every node, <i>/var/log/cloudian/cloudian-ui.log</i>
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS PriorityLevel [ThreadId] ClassName:MESSAGE</pre> In the case of log entries for user logins to the CMC, the MESSAGE value will be format-

	ted as follows: Normal login <pre>Login <groupId> <userId> from: <ipAddress> Success</pre> <pre>Login <groupId> <userId> from: <ipAddress> Failed [<reason>]</pre> SSO login <pre>SSOLogin <groupId> <userId> from: <ipAddress> Success</pre> <pre>SSOLogin <groupId> <userId> from: <ipAddress> Failed [<reason>]</pre>
Log Entry Example	<pre>2023-05-04 11:56:48,475 INFO [localhost-startStop-1] ServiceMapUtil>Loading service map info from: clouidianservicemap.json</pre>
Default Logging Level	INFO
Default Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-ui.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>. Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

CMC user audit log (ui-action.log)

Location	On every node, <i>/var/log/cloudian/ui-action.log</i>
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS SourceIP AdminID Action CanonicalUserId UserId GroupId Result</pre> The <i>ui-action.log</i> captures CMC actions relating to login, logout, and the creation, editing, and deletion of user accounts. This log is generated on each node running the CMC, and on each node records only activity processed by that node's CMC instance. Through the CMC you can download a CSV (comma-separated value) file that aggregates all the <i>ui-action.log</i> content from across all CMC instances. For more information about downloading the CSV file, and for more information about the content of this log, see "Download User Audit Log" (page 414).
Log Entry Example	<pre>2024-01-16 09:22:13,812 172.16.6.184 0%7Cadmin CreateUser testU2 test Success</pre>
Logging Level	Not applicable
Default Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>ui-action.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>. Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 90 days.

12.1.4. HyperStore Firewall Log

HyperStore firewall log (firewall.log)

If the HyperStore firewall is enabled in your system then DROP'd connections are logged in the HyperStore firewall log. For information about the firewall see "[HyperStore Firewall](#)" (page 850).

Location	On every node, <code>/var/log/cloudian/firewall.log</code>
Log Entry Format	The log records information about dropped packets, including the timestamp, host, firewall zone (<i>cloudian-backend</i> [for the designated internal interface] or <i>cloudian-frontend</i> [for all other interfaces]), interface name and MAC address, source and destination address, protocol, TCP flags, and so on.
Log Entry Example	<pre>Apr 18 11:03:19 demo4-node1 kernel: IN_cloudian-frontend_DROP: IN=eth0 OUT= MAC=52:54:00:e3:82:d7:52:54:00:11:dd:79:08:00 SRC=10.254.254.103 DST=10.254.254.118 LEN=44 TOS=0x00 PREC=0x00 TTL=57 ID=43247 PROTO=TCP SPT=52377 DPT=74 WINDOW=1024 RES=0x00 SYN URGP=0</pre>
Logging Level	Not applicable
Rotation and Retention Policy	Rotation occurs hourly if the live file size has reached 10MB; or else daily regardless of file size (except that there is no rotation of an empty live log file). Rotated files are named as <i>firewall.log-YYYY-MM-DD.HH.gz</i>. Rotated files are compressed with <i>gzip</i>. Rotated files are retained for 180 days and then automatically deleted.

12.1.5. HyperStore Service Logs

HyperStore Service application log (cloudian-hyperstore.log)

Location	On every node, <code>/var/log/cloudian/cloudian-hyperstore.log</code>
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS PriorityLevel[S3RequestId][ThreadId]MessageCode ClassName:MESSAGE</pre> - The MessageCode uniquely identifies the log message, for messages of level WARN or higher. For documentation of specific message codes including recommended operator action, see the "Log Message Codes" section of the CMC's online Help. - The S3RequestId value is present only in messages associated with implementing S3 requests.
Log Entry Example	<pre>2017-05-04 23:58:34,634 ERROR[][main]HS220008 CloudianAbstractServer:Unable to load configuration file: storage.xml</pre>
Default Logging Level	INFO
Default Rotation and	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.

Retention Policy	Rotated files are named as <i>cloudian-hyperstore.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>. Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

HyperStore Service request log (*cloudian-hyperstore-request-info.log*)

Location	On every node, <i>/var/log/cloudian/cloudian-hyperstore-request-info.log</i>
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS IpAddressOfClientS3Server S3RequestId HttpStatus HttpOperation OriginalUri HyperStoreFilePath ContentLength DurationMicrosecs Etag ECSuffix</pre> - The <code>IpAddressOfClientS3Server</code> is the IP address of the S3 Server node that submits the request to the HyperStore Service. - The <code>HttpOperation</code> is the HyperStore Service HTTP API operation that the S3 Service invokes and will be a simple operation like "PUT" or "GET". In the case of a "secure delete", the operation will be "SECURE-DELETE" and this request won't be logged until the third and final data overwriting pass for the object is completed. By contrast a regular delete will be logged as operation "DELETE". The secure delete feature is disabled by default. For more information see "Enabling Secure Delete" (page 202). - The <code>OriginalUri</code> field shows the group ID, bucket name, and object name from the originating S3 API request, in URI-encoded form ("CloudianTest1", "buser1", and "514kbtes", respectively, in the example above). - The <code>Etag</code> field will be "0" for operations other than PUT. - The <code>ECSuffix</code> field value is "null" if not an erasure coded data request.
Log Entry Example	Erasure coded object: <pre>2024-11-03 06:23:35,090 10.112.1.79 3f1b624f-d7b7-1e2c-a6b1-0a690b692ef3 200 GET /ec/nggrp%2Fm-MDA00TI1MjcxNjA0Mzg0NTM2MzYx%2Fecbn%2Fmpu..0002.1 /cloudian1/ec/std8ZdRJDskcPvmOg4/db35e009d75fff7fce122105f6d3b877/196/037/ 90637448863864709563044774788357063423.1604384536362536381-0A70014F 5242880 15277 0 3</pre> Replicated object: <pre>2024-11-03 06:42:22,201 10.112.1.79 3f1b627a-d7b7-1e2c-a6b1-0a690b692ef3 200 PUT /file/nggrp%2Fhsfsbn%2Fhsfs cloudian1/hsfs/std8ZdRJDskcPvmOg4/9894597460837bab308e73271 44b4bcc/ 227/232/169187708830921211004707767501030750269.1604385742187742235- 0A70014F 1048576 186769 379228ccb18fb5f795aaaa17a459f0ed null</pre>
Logging Level	Not applicable
Default Rotation and Retention	Rotation occurs if live file size reaches 300MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-hyperstore-request-info.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>.

Policy	Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 3GB or if oldest rotated file age reaches 180 days.

HyperStore Service cleanup log (cloudian-hyperstore-cleanup.log)

Location	On every node, <code>/var/log/cloudian/cloudian-hyperstore-cleanup.log</code>
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS OperationType Command# ChunkName ChunkFilePath CleanupReason</pre> This log has entries when an hsstool cleanup or hsstool cleanupec operation results in files being deleted from the node. A cleanup operation that determines that no files need to be deleted from the node will not cause any entries to this log. For background information about "chunks" and chunk names see "Creating an Input File" (page 924). For information about chunk file paths see "HyperStore Service and the HSFS" (page 66). The <i>CleanupReason</i> indicates why the data was removed from the target node by the cleanup (or cleanupec) operation and will be one of the following: • BUCKET_NOT_FOUND -- Object data was cleaned because it is associated with a bucket that no longer exists in Redis (i.e., the bucket was deleted previously) • METADATA_NOT_FOUND -- Object data was cleaned because the object's metadata no longer exists in Cassandra (i.e., the object was deleted previously) • KEYS_NOT_MATCH -- Object data was cleaned because the part number in the local digest does not match the part number from Cassandra. Applicable in the context of MPU or chunked objects. • NOT_A_HYPERSTORE_BLOB -- Object data was cleaned because the local file size is 0 • TIME_NOT_MATCH -- Object data was cleaned because the local object data timestamp doesn't match the corresponding object timestamp in Cassandra (i.e. the local object data is not current) • REMOTE_STORE -- Object data was cleaned because the object has been auto-tiered to a remote S3 compatible system • REPLICATION_OUT_RANGE -- Replica object data was cleaned because its token does not fall into the token ranges currently assigned to this node • EC_OUT_RANGE -- Erasure coded object data was cleaned because its token does not fall into the token ranges currently assigned to this node. • POLICY_NOT_FOUND -- Object data was cleaned because it is associated with a storage policy that no longer exists (i.e., the storage policy was deleted previously) • OUT_RANGE_UNCHECK -- Object data was cleaned because its token does not fall into the token ranges assigned to this node, and the cleanup was run with the <code>"-c false"</code> option so that safety checks were not performed to ensure that at least one replica (or $k+m$ fragments in the case of EC data) exists on other nodes • BLOB_NOT_FOUND -- Object digest was cleaned from disk because the corresponding object data was moved to a different disk by the Data Mover tool (to remedy disk utilization imbalance)

Log Entry Example	2023-11-21 21:41:21,453 BATCHCLEANUP 1 rfbn01/sig01 /cloudian3/hsfs/aEEQRBxhmfIOhHXaInfzQ/4c15002f55b01dea3a4c7201db415340/132/054/21615439588692861858090169671595275867.1700630522489522494-0A0A01E7 METADATA_NOT_FOUND
Logging Level	Not applicable
Default Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-hyperstore-cleanup.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i> Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

HyperStore Service repair log (cloudian-hyperstore-repair.log)

Location	On every node, <i>/var/log/cloudian/cloudian-hyperstore-repair.log</i>
Log Entry Format	This log has entries when a repair operation results in an attempt to repair data on the node. A repair operation that determines that no repairs are needed on the node will not cause any entries to this log (but will result in entries to the application log <i>cloudian-hyperstore.log</i>). In <i>cloudian-hyperstore-repair.log</i> there are two different types of entries, with different fields. The first type of entry records information about a repair action being taken and has this format: <pre>yyyy-mm-dd HH:mm:ss,SSS RepairType Command# Coordinator RepairEndpoint StreamFromEndpoint ChunkName Path ChunkSize Md5Hash RepairLatencyMillisecs Suffix</pre> In most cases the RepairType field value will be one of: • RR -- regular repair for replicas • PRR -- proactive repair for replicas • REC -- regular repair for erasure coded data • PREC -- proactive repair for erasure coded data • UECD -- update of EC digest fields The Coordinator is the node to which the <i>hsstool repair</i> or <i>hsstool repairec</i> command was submitted. The StreamFromEndpoint is the node from which a replica or erasure coded fragment was streamed in order to repair missing or bad data at the RepairEndpoint node. For erasure coded data repair, the fragment is streamed from the node that performed the decoding and re-encoding of the repaired object. The Suffix field indicates the fragment suffix in the case of erasure coding repairs. For replica repairs, suffixes are not applicable and this field's value will be "-1". Note For background information about "chunks" and chunk names see "Creating an Input File" (page 924). For information about chunk file paths see "HyperStore Service and the HSFS" (page 66).

	The other type of entry in <i>cloudian-hyperstore-repair.log</i> records the outcome of successful erasure coding repair tasks. These entries have this format: <pre>Ok yyyy-mm-dd HH:mm:ss,SSS RepairType ChunkName Path TaskType</pre> The ECRepairTaskType field value indicates the type of erasure coded fragment problem that was successfully repaired -- one of "[MD5_MISMATCH]", "[MISSING]", or "[DUPLICATE]". Note Failed repair tasks are logged in <i>cloudian-hyperstore-repair-failure.log</i> (described further below).
Log Entry Examples	Example #1 (information about an EC repair operation) <pre>2024-06-10 23:00:12,265 REC 13 10.20.1.81 10.20.1.81 10.20.1.81 m-MDAzYTMymTcxNjIzMzkwODA5Mzk2/ecbn/mpmc..0002.2 /cloudian3/ec/1SZe38K1qp98VDadplVxyq/257cb007cd57283bad02763e83864bb7/075/133/27936503055090146240017134150181984843.1623390809396809455-0A140151 524288 8ad63b8f18e934333bdc2164bd7f8553 8 2</pre> Example #2 (a successful EC repair task outcome): <pre>Ok 2024-06-10 23:00:12,272 REC m-MDAzYTMymTcxNjIzMzkwODA5Mzk2/ecbn/mpmc..0002.2 /cloudian3/ec/1SZe38K1qp98VDadplVxyq/257cb007cd57283bad02763e83864bb7/075/133/27936503055090146240017134150181984843.1623390809396809455-0A140151 [MISSING]</pre>
Logging Level	Not applicable
Default Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-hyperstore-repair.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i> Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

HyperStore Service repair failure log (*cloudian-hyperstore-repair-failure.log*)

Location	On every node, <i>/var/log/cloudian/cloudian-hyperstore-repair-failure.log</i>
Log Entry Format	When <i>hsstool repair</i> or <i>hsstool repairec</i> has been run, this log records an entry for each chunk for which the repair attempt failed on the node. These entries have this format: <pre>ChunkName Path OperationId yyyy-mm-dd HH:mm:ss,SSS FailReason[TaskType]</pre> The OperationId is a system-generated unique identifier of the repair run (i.e., each time you run <i>hsstool repair</i> or <i>hsstool repairec</i> the run will be assigned its own unique ID, so that different runs can be distinguished from each other). The FailReason field value indicates the failure reason -- for example "LESS_THAN_K" in the event that fewer than K good fragments were available for an erasure coded chunk. The TaskType field may or may not be present, depending on the FailReason. When present, the TaskType field indicates the type of problem that the failed repair task was attempting to address -- either "[MD5_MISMATCH]" or "[MISSING]" or "[DUPLICATE]".

	Note The <i>cloudian-hyperstore-repair-failure.log</i> file -- or excerpts from this log file, that you save into a separate file -- can be used as input to a <i>repair -f <input-file></i> operation or a <i>repairc -f <input-file></i> operation, to try again to repair the chunks for which the prior repair attempt failed. See "hsstool repair" (page 907) or "hsstool repairc" (page 918). Note For background information about "chunks" and chunk names see the "Creating an Input File" section of "hsstool repair" (page 907) or "hsstool repairc" (page 918). For information about chunk file paths see "HyperStore Service and the HSFS" (page 66).
Log Entry Examples	Example #1: <pre>ecbn/hoge /cloudian2/ec/std8ZdRJDskcPvmOg4/96f96c866408a3fc9e42237ae94bce5d/170/045/156312679751569259476355507179662176759.1631169767765768030-0A32C89D d8cd52cc-0e72-1cd4-b03c-000c29b5a219 2024-09-08 23:47:42,654 NODE_DOWN</pre> Example #2: <pre>ecbn/sig /cloudian1/ec/1IVfPFBIIjTqx0GvLCwoi0/35418348e30905d622f692c451868e73/172/190/25879583587069690807015587486564246118.1626338080577080582-0A700108 3bad2bf2-95ce-1070-b320-0acf12cb1e97 2024-07-15 08:47:55,126 EC_DECODE_FAILED [MISSING]</pre>
Logging Level	Not applicable
Default Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-hyperstore-repair-failure.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i> Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 10000 MB or if oldest rotated file age reaches 180 days.

HyperStore Service whereis log (whereis.log)

This log is generated if and only if you execute the *hsstool whereis -a* command, to output location detail for every S3 object in the system. For information about this log see [hsstool whereis](#).

12.1.6. HyperStore Shell (HSH) Log

HyperStore shell log (hsh.log)

If the HyperStore shell (HSH) is enabled in your system then HSH user logins and commands are logged in the HyperStore shell log. An instance of the HyperStore shell log resides on each HyperStore node and records any HSH logins and commands on that node. For information about the HSH see "[HyperStore Shell \(HSH\) Feature Overview](#)" (page 105).

Location	On every node, <i>/var/log/hsh/hsh.log</i>
Log Entry Examples	HSH user login: <pre>time="2024-09-04T20:26:21-07:00" level=info msg="New Session" session=15C16CFDA4A46863 user=sa_admin</pre> HSH user running a command: <pre>time="2024-09-04T21:22:30-07:00" level=info msg="Running command." args="<s:setup>" command=hspkg session=15C16CFDA4A46863 type=interactive</pre> <pre>time="2024-09-04T21:22:30-07:00" level=info msg="Executing command." cwd=/home/sa_admin mode=User path=/opt/cloudian-staging/7.2/system_setup.sh runuser=root session=15C16CFDA4A46863 tty=true</pre> <pre>time="2024-09-04T21:24:38-07:00" level=info msg="Command complete." args="<s:setup>" command=hspkg session=15C16CFDA4A46863 status=0 type=interactive</pre>
Logging Level	INFO
Default Rotation and Retention Policy	Rotation occurs monthly or if the live file size reaches 500MB. Rotated files are named as <i>hsh.log.<n></i>, with <i>hsh.log.1</i> being the most recently rotated file. The most recently rotated file is not compressed; all other rotated files are compressed with <i>gzip</i> and the file names will have a <i>.gz</i> suffix. 12 rotated files are retained. Older files are automatically deleted.

12.1.7. IAM Service Logs

IAM application log (cloudian-iam.log)

Location	On every node in the default service region, <i>/var/log/cloudian/cloudian-iam.log</i> Note For more information on the IAM Service see the "IAM API" section of the <i>Cloudian HyperStore AWS APIs Support Reference</i>.
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS PriorityLevel[ThreadId]MessageCode ClassName:MESSAGE</pre> The MessageCode uniquely identifies the log message, for messages of level WARN or higher. For documentation of specific message codes including recommended operator action, see the "Log Message Codes" section of the CMC's online Help.
Log Entry Example	<pre>2023-02-16 10:16:36,246 INFO[qtp214187874-41] CloudianHFactory:Creating DynamicKS for: AccountInfo</pre>
Default Logging Level	INFO

Default Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-iam.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>. Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

IAM request log (*cloudian-iam-request-info.log*)

Location	On every node in the default service region, <i>/var/log/cloudian/cloudian-iam-request-info.log</i> Note This log records Security Token Service (STS) requests as well as IAM requests.
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS ClientIpAddress AccountRootUserCanonicalId RequestorUserId GroupId Protocol:Action IamUserId RoleSessionArn RoleSessionId TempCredentialsAccessKey HttpStatus ErrorCode ResponseData DurationMicrosecs</pre> - For the <i>IamUserId</i> field: - If request is by IAM user, this is the <i>UserId</i> of the IAM user - If request is by a non-SAML role session, this is the <i>UserId</i> of the IAM user who assumed the role - If request is by a SAML role session, this is the <i>Subject</i> field value from the SAML Assertion (the user identifier) - If request is by an account root user, this field is empty - If request is by a role session, the <i>RoleSessionArn</i>, <i>RoleSessionId</i>, and <i>TempCredentialsAccessKey</i> fields provide information about the role session making the request. Otherwise these fields are empty. - If the action is <i>AssumeRole</i> or <i>AssumeRoleWithSAML</i>, the <i>ResponseData</i> field provides information about the role being assumed (<i>RoleSessionArn&RoleSessionId&TempCredentialsAccessKey</i>). Otherwise this field is empty.
Log Entry Examples	<pre>2024-08-10 18:58:59,607 10.20.2.34 679d95846fb0f0047f5926ba16546552 testu159986 myGroup8732 iam:PutRolePolicy 200 6</pre> <pre>2024-08-10 18:58:59,619 10.20.2.34 679d95846fb0f0047f5926ba16546552 testu159986 myGroup8732 sts:AssumeRole aidcc54d3e60de2a74e89ad639561df0 200 arn:aws:iam::679d95846fb0f0047f5926ba16546552:role/iammypath/rolen134094& rolesn54301&asicbd8ef4bdd6e7e03c 118</pre>
Logging Level	Not applicable
Default Rotation and Retention Policy	Rotation occurs if live file size reaches 100MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-iam-request-info.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>.

	Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 2GB or if oldest rotated file age reaches 180 days.

Logging the True Originating Client IP Address

If you use a load balancer in front of your IAM Service, then by default the *ClientIpAddress* in your IAM request logs will be the IP address of a load balancer rather than that of the end client. If you want the IAM request logs to instead show the end client IP address, see the configuration setting "[com-mon.xForwardedHeaders.enabled](#)" (page 759).

12.1.8. Monitoring Agent and Collector Logs

Monitoring Agent application log (*cloudian-agent.log*)

Location	On every node, <i>/var/log/cloudian/cloudian-agent.log</i>
Log Entry Format	yyyy-mm-dd HH:mm:ss,SSS PriorityLevel [ThreadId] ClassName:MESSAGE
Log Entry Example	2017-05-04 11:57:03,698 WARN [pool-2-thread-7] LogFileTailerListener:Log file not found for service:cron
Default Logging Level	WARN
Default Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-agent.log.YYYY-MM-DD.i.gz</i> , where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i> . Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

Monitoring Data Collector application log (*cloudian-datacollector.log*)

Location	On the Monitoring Data Collector node, <i>/var/log/cloudian/cloudian-datacollector.log</i>
Log Entry Format	yyyy-mm-dd HH:mm:ss,SSS PriorityLevel[ThreadId]MessageCode ClassName:MESSAGE The MessageCode uniquely identifies the log message, for messages of level WARN or higher. For documentation of specific message codes including recommended operator action, see the "Log Message Codes" section of the CMC's online Help.
Log Entry Example	2017-05-04 00:00:05,898 WARN[main]DC040073 SmtNotification: Failed to send due to messaging error: Couldn't connect to host, port: smtp.notification.configure.me, 465; timeout 5000
Default Logging Level	WARN (configurable by the setting " monitoring.dataCollector.logging.level " (page 807))
Default Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-datacollector.log.YYYY-MM-DD.i.gz</i> , where <i>i</i> is a rota-

	tion counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>. Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

12.1.9. Phone Home (Smart Support) Log

Phone Home application log (cloudian-phonehome.log)

Location	On the Monitoring Data Collector node, <i>/var/log/cloudian/cloudian-phonehome.log</i>
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS PriorityLevel[ThreadId]MessageCode ClassName:MESSAGE</pre> The MessageCode uniquely identifies the log message, for messages of level WARN or higher. For documentation of specific message codes including recommended operator action, see the "Log Message Codes" section of the CMC's online Help.
Log Entry Example	<pre>2023-05-04 19:13:03,384 ERROR[main]HS200043 S3:All Redis read connection pools are unavailable. Redis HGETALL of key BPP_MAP fails.</pre>
Default Logging Level	WARN
Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-phonehome.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>. Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

12.1.10. Redis (Credentials DB and QoS DB) and Redis Monitor Logs

Redis Credentials application log (redis-credentials.log)

Location	On Redis Credentials nodes, <i>/var/log/redis/redis-credentials.log</i>
Log Entry Format	<pre>PID.role dd MMM hh:mm:ss.ms loglevel MESSAGE</pre> The role will usually be "M" for master or "S" for slave. The loglevel will be "*" for NOTICE or "#" for ERROR.
Log Entry Example	<pre>18290:S 28 Jul 01:23:42.416 # Connection with master lost.</pre>
Logging Level	NOTICE
Rotation Policy	Not rotated by default. You can set up rotation by using logrotate .

Redis QoS application log (redis-qos.log)

Location	On Redis QoS nodes, <i>/var/log/redis/redis-qos.log</i>

Log Entry Format	<pre>PID.role dd MMM hh:mm:ss.ms loglevel MESSAGE</pre> The role will usually be "M" for master or "S" for slave. The loglevel will be "*" for NOTICE or "#" for ERROR.
Log Entry Example	<pre>24401:M 28 Jul 01:41:46.963 * Calling fsync() on the AOF file.</pre>
Logging Level	NOTICE
Rotation Policy	Not rotated by default. You can set up rotation by using logrotate .

Redis request logs (*redis-{s3,admin,hss}-tx.log*)

Location	If enabled, these logs are written to <i>/var/log/cloudian/redis-{s3,admin,hss}-tx.log</i> on the S3 Service, Admin Service, and/or HyperStore nodes (that is, these request logs are written on the client side as the S3, Admin, and HyperStore services instances send requests to Redis).
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS S3RequestId MESSAGE</pre>
Log Entry Example	<pre>2024-11-17 23:54:01,695 CLIENT setname ACCOUNT_GROUPS_M</pre>
Logging Level	Not applicable
Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>redis-{s3,admin,hss}-tx.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>. Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

Redis Monitor application log (*cloudian-redismon.log*)

Location	On Redis Monitor nodes, <i>/var/log/cloudian/cloudian-redismon.log</i>
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS PriorityLevel[ThreadId]MessageCode ClassName:MESSAGE</pre> The MessageCode uniquely identifies the log message, for messages of level WARN or higher. For documentation of specific message codes including recommended operator action, see the "Log Message Codes" section of the CMC's online Help.
Log Entry Example	<pre>2017-05-04 00:01:13,590 INFO[pool-23532-thread-3] RedisCluster: Failed to connect to cloudian jmx service [mothra:19082]</pre>
Default Logging Level	INFO
Default Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-redismon.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>.

	Deletion of oldest rotated log file occurs if aggregate rotated files size reaches 100MB or if oldest rotated file age reaches 180 days.

12.1.11. S3 Service Logs (including Auto-Tiering, CRR, and WORM)

S3 Service application log (*cloudian-s3.log*)

Location	On every node, <i>/var/log/cloudian/cloudian-s3.log</i>
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS PriorityLevel[S3RequestId][ThreadId]MessageCode ClassName:MESSAGE</pre> The MessageCode uniquely identifies the log message, for messages of level WARN or higher. For documentation of specific message codes including recommended operator action, see the "Log Message Codes" section of the CMC's online Help.
Log Entry Example	<pre>2017-05-04 00:33:39,435 ERROR[bc6f3fca-9037-135f-a964-0026b95cedde] [qtp1718906711-94]HS204017 XmlSaxParser:Rule doesn't have AllowedMethods/AllowedOrigins.</pre>
Default Logging Level	INFO
Default Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-s3.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>. Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

Note The S3 Service proxies any S3 *SelectObjectContent* requests through to the S3 Select Service for processing. The S3 Select Service has its own dedicated application log, *s3-select.log*.

S3 Service request log (*cloudian-request-info.log*)

Location	On every node, <i>/var/log/cloudian/cloudian-request-info.log</i>
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS ClientIpAddress BucketOwnerUserId Operation BucketName RequestorUserId RequestHeaderSize RequestBodySize ResponseHeaderSize ResponseBodySize TotalRequestResponseSize DurationMicrosecs UrlEncodedObjectName HttpStatus S3RequestId Etag ErrorCode SourceBucketName/UrlEncodedSourceObjectName GroupId CanonicalUserId IamUserId RoleSessionArn RoleSessionId AccessKey UserAgent DeleteSize BucketCanonicalId</pre> The Operation field indicates the S3 API operation. Note that "getBucket" indicates GET Bucket (List Objects) Version 1 whereas "getBucketV2" indicates GET Bucket (List Objects) Version 2. (In the case of a health check request, the Operation field indicates "healthCheck". In the case of requests submitted to the S3 Service by a system cron job,

the Operation field indicates the name of the cron job action, such as "sytemBatchDelete").

- The Etag field is the Etag value from the response, if applicable to the request type. For information about Etag see for example [Common Response Headers](#) from the Amazon S3 REST API spec. This field's value will be 0 for request/response types that do not use an Etag value.
- The ErrorCode field is the Error Code in the response body, applicable only for potentially long-running requests like PUT Object. If there is no Error Code in the response body this field's value will be 0. For possible Error Code values see [Error Responses](#) from the Amazon S3 REST API spec.

Note In the case where the Operation field value is *deleteObjects*, the ErrorCode field will be formatted as *objectname1-errorcode1,objectname2-errorcode2,objectname3-errorcode3....*, and the object names will be URL-encoded. If there are no errors the field is formatted as *objectname1-0,objectname2-0,objectname3-0....*

- The SourceBucketName/UrlEncodedSourceObjectName field is populated only for copyObject and uploadPartCopy operations and is empty for other operation types.
- For the iamUserId field:
 - If request is by IAM user, this is the UserId of the IAM user
 - If request is by a non-SAML role session, this is the UserId of the IAM user who assumed the role
 - If request is by a SAML role session, this is the Subject field value from the SAML Assertion (the user identifier)
 - If request is by an account root user, this field is empty
- If request is by a role session, the RoleSessionArn and RoleSessionId fields provide information about the role session making the request. Otherwise these fields are empty.
- The AccessKey is the access key of the account root user, IAM user, or IAM role session that submitted the request. This is from the user's or role session's security credentials, which consist of an access key and a secret key.
- The UserAgent value is truncated to 35 characters if longer than that
- The DeleteSize is the number of net bytes deleted, if the operation is a deleteObject operation (this is the deleted object size, excluding any overhead from replication or erasure coding). For operations other than deleteObject this field's value will be the string 'null'.
- The BucketCanonicalId is a unique system-generated ID assigned to the bucket. (Note that if an administrator runs the Admin API operation *POST /bucketops/purge* against a bucket -- which deletes every object in the bucket -- the system assigns the bucket a new canonical ID.)

NOTE:

Cloudian HyperIQ is a solution for dynamic visualization and analysis of HyperStore system monitoring data and S3 service usage data. HyperIQ is a separate product available from Cloudian that deploys as virtual appliance on VMware or VirtualBox and integrates with your existing HyperStore system. For more information about HyperIQ contact your Cloudian representative.

Log Entry Example	2024-06-17 01:09:07,138 10.50.41.207 user1 putObject bucket2 user1 396 70 180 405 1051 6619 obj1 307 668d5360-071c-153f-becd-000c29bd5995 0 TemporaryRedirect CloudianTest1 f1a312a31d912441be2fea02f8677982 5cb6967a8f8569facb6d Boto/2.42.0 Python/2.7.5 Linux/3.10 null 620dbabd2b540109d7f945b15c5b6079
Logging Level	Not applicable
Default Rotation and Retention Policy	Rotation occurs if live file size reaches 100MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-request-info.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>. Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 2GB or if oldest rotated file age reaches 180 days.

Logging the True Originating Client IP Address

If you use a load balancer in front of your S3 Service, then by default the *ClientIpAddress* in your S3 request logs will be the IP address of a load balancer rather than that of the end client. If you want the S3 request logs to instead show the end client IP address, see the configuration settings "[s3.proxyProtocol.enabled](#)" (page 826) and "[common.xForwardedHeaders.enabled](#)" (page 759).

Auto-Tiering request log (*cloudian-tiering-request-info.log*)

Location	On every node, <i>/var/log/cloudian/cloudian-tiering-request-info.log</i> Note All nodes participate in tiering objects to their intended destinations systems. In the case of regular auto-tiering based on user-defined schedules, the tiering processing workload is randomly spread across the nodes in the service region in which the cron job host resides. In the case of Bridge Mode tiering (also known as proxy tiering), whichever node processes the S3 uploading of an object into its source bucket also processes the immediate tiering of that object to its intended destination system. For more information on auto-tiering see "Auto-Tiering Feature Overview" (page 203).
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS Command Protocol SourceBucket/Object SourceObjectVersion TargetBucket TargetObjectVersion ObjectSize TotalRequestSize Status DurationMicrosecs Mode AttemptCount</pre> - The Mode field will be one of AUTO_TIERING (for regular auto-tiering), BRIDGE_MODE (for Bridge Mode auto-tiering), or SCHEDULER (for retry attempts for Bridge Mode tiering of objects) - The AttemptCount is applicable only to Bridge Mode tiering and associated retries, and indicates how many attempts have been made to tier the object. This field's value will be "-1" if the Mode is AUTO_TIERING.
Log Entry Example	2024-04-23 05:57:35,914 MOVEOBJECT S3 tps01/abc801 null tps02 null 33 39 ERR 268776 BRIDGE_MODE 1
Logging	Not applicable

Level	
Default Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-tiering-request-info.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>. Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

Cross-Region Replication request log (cloudian-crr-request-info.log)

Location	On every node, <i>/var/log/cloudian/cloudian-crr-request-info.log</i> Note Whichever S3 Service node processes a PUT of an object into a source bucket configured for CRR will be the node that initiates the replication of the object to the destination bucket. This node will have an entry for that object in its CRR request log. In the case of retries of replication attempts that failed with a temporary error the first time, the retries will be logged in the CRR request log on the cron job node. For general information on the cross-region replication feature, see "Cross-Region Replication Feature Overview" (page 213).
Log Entry Format	yyyy-mm-dd HH:mm:ss,SSS SourceBucket/Object ObjectVersionId DestinationBucket CrrOperation Status DurationMillisecs Size The Status will be one of: • COMPLETED -- The object was successfully replicated to the destination bucket. • FAILED -- The object replication attempt resulted in an HTTP 403 or 404 response from the destination system. This is treated as a permanent error and no retry attempt will occur for replicating this object. This type of error could occur if for example the destination bucket has been deleted or if versioning has been disabled for the destination bucket. A FAILED status triggers an S3 service error Alert in the CMC's Alerts page. • PENDING -- The object replication attempt encountered an error other than an HTTP 403 or 404 response. All other errors -- such as a connection error or an HTTP 5xx response from the destination system -- are treated as temporary errors. The object replication will be retried again once every four hours until either the object is successfully replicated to the destination bucket or a permanent error is encountered. Each retry attempt results in a new log entry in <i>cloudian-crr-request-info.log</i>.
Log Entry Example	<pre>2024-03-04 15:04:58,423 prod-2024-01-16/0299-0166180000362773.pdf fe15a1de-de25-f8af-9a28-b4a9fc08ca7e prod-backup-2024-01-16 REPLICATEOBJECT COMPLETED 61 328471</pre>
Logging Level	Not applicable
Default Rotation and Retention	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-crr-request-info.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>.

Policy	Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

WORM audit log (s3-worm.log)

Location	On every node, <code>/var/log/cloudian/s3-worm.log</code> Note For information about the WORM feature see "Object Lock Feature Overview" (page 221).
Log Entry Format	DateTime Hostname S3RequestId S3Operation Headers Bucket Object ObjectVersionId CanonicalUserId CanonicalIamUserId StatusCode StatusMessage The S3Operation will be any of: • An object lock operation (<i>GET/PUT Bucket object lock configuration</i> or <i>GET/PUT Object legal hold</i> or <i>GET/PUT Object retention</i>) • A regular S3 operation that includes object lock related headers (such as a <i>PUT Bucket</i> request that includes an <code>x-amz-bucket-object-lock-enabled: true</code> header or a <i>PUT Object</i> request that includes an <code>x-amz-object-lock-mode</code> or <code>x-amz-object-lock-retain-until-date</code> or <code>x-amz-object-lock-legal-hold</code> header). For such operations the relevant header(s) will be shown in the Headers field of the log entry. Multipart uploads are logged only if completed, and the operation is indicated as <i>PUT Object</i>. • A <i>DELETE Object Version</i> request in regard to a locked object version. For a regular S3 operation such as <i>PUT Bucket</i> or <i>PUT Object</i>, the Headers field will show the object lock related headers. For <i>GET/PUT Bucket object lock configuration</i> operations the Headers field will convey information about the bucket's default object lock configuration, coded as follows: • First character: "T" for object lock is enabled on bucket • Second character: "G" for Governance mode or "C" for Compliance mode • Third character: "D" or "Y" for retention period unit of measurement (days or years) • Remaining characters: Integer for number of days or years Example 1: <i>TGD30</i> for a Governance mode configuration with 30 day retention period Example 2: <i>T</i> for a bucket with object lock enabled, but no default object lock configuration If the submitter of the request is an IAM user, the log entry shows the canonical user ID of the IAM user as well as the canonical user ID of the parent user account.
Log Entry Example	<pre>2024-10-06 09:59:30,767 arcturus a9d7e5b1-e85a-11e9-8519-52540014b047 s3:GetBucketObjectLockConfiguration TGD90 newbucket b584fb57480af5108e32d17f10c5cb7b 200 OK</pre>
Logging Level	Not applicable
Default	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regard-

Rotation and Retention Policy	less of live file size. Rotated files are named as <code>s3-worm.log.YYYY-MM-DD.i.gz</code>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>. Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

12.1.12. Search Service Log

Search Service sync log (essynctool.log)

Location	On the node that runs the HyperStore system cron jobs, <code>/var/log/cloudian/essynctool.log</code> (this log will also be generated on the Config Controller node if you use the <code>hscctl search sync</code> command, which you should not do unless directed to by Cloudian Support). Note For an overview of the optional HyperStore Search Service, see "Object Metadata and Search Feature Overview" (page 230) and "Preparing the Metadata Search Feature" (page 236)..
Log Entry Format	<code>yyyy-mm-dd HH:mm:ss,SSS PriorityLevel[ThreadId]MESSAGE</code>
Log Entry Example	2024-10-08 18:15:13,652 INFO[main] ElasticsearchSync:Processing loading metadata to elasticsearch for UserData_0c80c2335551a4c8f2e0b3b8acc6c5da Note Although log messages may reference Elasticsearch, the current HyperStore Search Service is actually built on OpenSearch.
Default Logging Level	INFO
Default Rotation	Rotation occurs if live file size reaches 10MB.

12.1.13. SQS Service Logs

SQS application log (cloudian-sqs.log)

Location	On every node, <code>/var/log/cloudian/cloudian-sqs.log</code> Note For an SQS overview (including information about how to enable the SQS Service, which is disabled by default) see the SQS API section in the <i>Cloudian HyperStore AWS APIs Support Reference</i>.

Log Entry Format	yyyy-mm-dd HH:mm:ss,SSS PriorityLevel[ThreadId]MessageCode ClassName:MESSAGE
Log Entry Example	2024-11-05 02:32:44,453 INFO[SQSRetentionCheckerThread - pool-5-thread-1] SQSMessageRetentionChecker:Another thread processing. Do nothing...
Default Logging Level	INFO
Default Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-sqs-req.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>. Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

SQS request log (*cloudian-sqs-request.log*)

Location	On every node, <i>/var/log/cloudian/cloudian-sqs-request.log</i> Note For an SQS overview (including information about how to enable the SQS Service, which is disabled by default) see the SQS section in the <i>Cloudian Hyper-Store AWS APIs Support Reference</i>.
Log Entry Format	yyyy-mm-dd HH:mm:ss,SSS PriorityLevel[ThreadId]MessageCode ClassName:MESSAGE The MESSAGE indicates information about the request including the request timestamp, the client IP address, the user ID, the request type (the SQS "action"), and the HTTP response status code.
Log Entry Example	2024-07-11 13:50:21,736 INFO[qtp1548962651-122] RequestLogger:2024-07-11 13:50:21,732 10.20.2.61 user1 CreateQueue testQueue 200 4
Logging Level	Not applicable
Default Rotation and Retention Policy	Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size. Rotated files are named as <i>cloudian-sqs-req.log.YYYY-MM-DD.i.gz</i>, where <i>i</i> is a rotation counter that resets back to 1 after each day. Rotated files are compressed with <i>gzip</i>. Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB or if oldest rotated file age reaches 180 days.

Logging the True Originating Client IP Address

If you use a load balancer in front of your SQS Service, then by default the client IP address in the MESSAGE field in your SQS request logs will be the IP address of a load balancer rather than that of the end

client. If you want the SQS request logs to instead show the end client IP address, see the configuration setting "[common.xForwardedHeaders.enabled](#)" (page 759).

12.2. Log Rolling Settings

[Note: In HyperStore 7.x and earlier, log rolling was configured in *log4j-*.xml.erb* files]

The following settings, in association with specific services' logging configuration -- for example *s3.-logging.<setting>* -- control the rolling (rotation) of log files and the retention policy for rolled log files:

rollover.maxSize -- Roll the active log if it reaches this size. Logs are rolled daily regardless of size, but if an active log file reaches this size before the regular daily roll is scheduled to occur it will be rolled immediately upon reaching this size (and a new active log file will be created). In this scenario a scheduled daily roll will also occur, for the log file that is active at the time of the scheduled daily roll.

rollover.maxAccumulatedSize -- If the aggregate size (after compression) of rolled log files for this log reaches this size, the system automatically deletes the oldest rolled log file.

rollover.maxAccumulatedFiles -- If the total number of rolled log files for this log reaches this number, the system automatically deletes the oldest rolled log file.

rollover.maxAge -- If a rolled log file reaches this age, the system automatically deletes the rolled log file. This must be specified in ISO-8601 format, for example *180d* (to indicate 180 days).

Deletion of the oldest rolled log files will be triggered by **any one** of the *rollover.maxAccumulatedSize*, *rollover.maxAccumulatedFiles*, and *rollover.maxAge* limits being reached, even if the other two limits haven't been reached yet. For example if the *rollover.maxAccumulatedSize* is reached then the oldest rolled log file will be deleted even if that rolled log file has not yet reached *rollover.maxAge*.

The table below lists the log rolling settings as associated with specific services, the default values, and the 'apply' and 'restart' targets if you edit a setting.

Note that settings named as *<service>.logging.<rollover setting>* pertain to all the logs associated with that service, unless superseded by a corresponding *<service>.logging.<log type>.<rollover setting>*. For example:

- *s3.logging.rollover.maxAge* pertains to all logs associated with the S3 service -- application logs and request logs -- since there are no *s3.logging.<log type>.rollover.maxAge* settings.
- *s3.logging.rollover.maxAccumulatedFiles* pertains only to S3 application logs since there is also a *s3.-logging.request.rollover.maxAccumulatedFiles* setting that applies specifically to S3 request logs.

Setting	Default	Apply	Restart
admin.logging.request.rollover.maxAccumulatedSize	2000mb	admin	s3
admin.logging.request.rollover.maxSize	100mb		
admin.logging.rollover.maxAccumulatedFiles	100		
admin.logging.rollover.maxAccumulatedSize	100mb		
admin.logging.rollover.maxAge	180d		
admin.logging.rollover.maxSize	10mb		

Setting	Default	Apply	Restart
cmc.logging.rollover.maxAccumulatedFiles	100	cmc	cmc
cmc.logging.rollover.maxAccumulatedSize	100mb		
cmc.logging.rollover.maxAge	180d		
cmc.logging.rollover.maxSize	10mb		
hyperstore.logging.error.rollover.maxAge	30d	hyperstore	hyperstore
hyperstore.logging.heal.rollover.maxAge	30d		
hyperstore.logging.redisTX.rollover.maxSize	100mb		
hyperstore.logging.repairFailure.rollover.maxAccumulatedFiles	1000		
hyperstore.logging.repairFailure.rollover.maxAccumulatedSize	10gb		
hyperstore.logging.request.rollover.maxAccumulatedFiles	100		
hyperstore.logging.request.rollover.maxAccumulatedSize	3gb		
hyperstore.logging.request.rollover.maxSize	300mb		
hyperstore.logging.rollover.maxAccumulatedFiles	100		
hyperstore.logging.rollover.maxAccumulatedSize	100mb		
hyperstore.logging.rollover.maxAge	180d		
hyperstore.logging.rollover.maxSize	10mb		
iam.logging.request.rollover.maxAccumulatedSize	2gb	iam	iam
iam.logging.rollover.maxAccumulatedFiles	100		
iam.logging.rollover.maxAccumulatedSize	100mb		
iam.logging.rollover.maxAge	180d		
iam.logging.rollover.maxSize	10mb		
monitoring.agent.logging.rollover.maxAccumulatedFiles	100	monitoring	no restart needed
monitoring.agent.logging.rollover.maxAccumulatedSize	10mb		
monitoring.agent.logging.rollover.maxAge	180d		
monitoring.agent.logging.rollover.maxSize	10mb		
redis.monitoring.logging.rollover.maxAccumulatedFiles	100	redis	redismon
redis.monitoring.logging.rollover.maxAccumulatedSize	100mb		
redis.monitoring.logging.rollover.maxAge	180d		
redis.monitoring.logging.rollover.maxSize	10mb		

Setting	Default	Apply	Restart
s3.logging.consistentDelete.rollover.maxAccumulatedFiles	100	s3	s3
s3.logging.crossRegionReplication.rollover.maxAccumulatedFiles	100		
s3.logging.inventoryRequest.rollover.maxAccumulatedFiles	100		
s3.logging.request.rollover.maxAccumulatedFiles	1000		
s3.logging.request.rollover.maxAccumulatedSize	2gb		
s3.logging.request.rollover.maxSize	100mb		
s3.logging.rollover.maxAccumulatedFiles	100		
s3.logging.rollover.maxAccumulatedSize	100mb		
s3.logging.rollover.maxAge	180d		
s3.logging.rollover.maxSize	10mb		
s3.logging.tiering.rollover.maxAccumulatedFiles	100	sqs	sqs
sqs.logging.rollover.maxAccumulatedFiles	100		
sqs.logging.rollover.maxAccumulatedSize	100mb		
sqs.logging.rollover.maxAge	180d		
sqs.logging.rollover.maxSize	10mb		

12.3. Forwarding Log Messages to a Central Log Server

If you wish you can have HyperStore forward application log and request log messages to a central logging server using *rsyslog*. For details see "[logging.syslog.remote.*](#)" (page 803).

12.4. Using the HSH to View Logs

If you are using the [HyperStore Shell \(HSH\)](#) to manage your HyperStore nodes, the HSH supports a command for viewing HyperStore logs -- specifically, log files under the directory */var/log/cloudian*.

To use the HSH to view HyperStore log files, first log into the Config Controller node as an HSH user. Upon successful login the HSH prompt will appear as follows:

```
<username>@<hostname>$
```

For example:

```
sa_admin@hyperstore1$
```

To view a HyperStore log:

```
$ hslog /var/log/cloudian/<logfile>
```

Note You must include the file path */var/log/cloudian/* -- not just the log file name.

For example:

```
$ hslog /var/log/cloudian/cloudian-admin.log
```

The command *hslog /var/log/cloudian/<logfile>* writes the log to *stdout*.

If you wish you can use command piping -- `hsslog /var/log/cloudian/<logfile> | <command>` -- to pipe the output to *head* or *tail* or *less* or *grep*, for example.

For details about the logs you can view, see ["HyperStore Logs"](#) (page 667).

Chapter 13. Configuration

13.1. Using *hsctl* to Control HyperStore Configuration

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Checking Current Configuration Values" \(page 694\)](#)
- ["Changing the Value of a Setting and Applying Your Change" \(page 694\)](#)
- ["Additional Options When Using *hsctl* for Configuration Control" \(page 696\)](#)

With HyperStore you can easily manage system configuration by using the HyperStore command line tool *hsctl*. With *hsctl* you can:

- Check the current value of a configuration setting or a group of related settings
- Change a configuration value, apply your change to the system, and restart the affected service

You can run *hsctl* from **any directory on your Config Controller Primary node**. If you're not sure which node is your Config Controller Primary node, check the CMC's ["Cluster Information" \(page 490\)](#) page. Note that throughout this documentation, references to the Config Controller node mean the Config Controller Primary node -- not the Config Controller Backup node.

The *hsctl* tool can be run by the *root* user or by a [HyperStore Shell user](#) with the Trusted role.

Note that in HyperStore 8.x, as compared to HyperStore 7.x and earlier versions:

- You no longer need to manually edit configuration files. Under the hood, the *hsctl* tool leverages Salt configuration management technology to automatically control configuration files throughout your HyperStore cluster.
- What used to be called the Configuration Master node in earlier HyperStore versions is now called the Config Controller node in HyperStore 8.0.
- All configuration settings have been renamed in HyperStore 8.0. The new setting naming system is hierarchical, consistent, and intuitive. The new names are all dot-segmented, with camel case used within each segment, and in most cases the first segment indicates the service to which the setting applies. To assist you with the transition from the old setting names to the new setting names:
 - In the HyperStore documentation, each setting description indicates both the new name and the old name (if the setting existed in earlier HyperStore versions). So searching on an old name will take you to the new name (if the old setting has been migrated to the new system, which is the case for most but not all of the old settings).
 - If on the command line you run *hsctl config get <old setting name>*, in most instances the error response will indicate the new setting name. For example:

```
[root@cloudian04 ~]# hsctl config get cmc_application_name
the HyperStore parameter "cmc_application_name" is deprecated. Did you mean this?
"cmc.applicationName"
```

13.1.1. Checking Current Configuration Values

To check the current value of **an individual configuration setting**, on the Config Controller node run this *hsctl* command:

```
# hsctl config get <setting>
```

For example:

```
# hsctl config get s3.sharedBucket.enabled
false
```

HyperStore configuration settings are organized into setting groups, in most cases corresponding to a particular service such as the S3 Service or the Admin Service. To check the current values of **all the settings in a setting group**, run this command:

```
# hsctl config get <settingGroup>
```

This will return, in alphabetical order, all the settings in the setting group and their current values. For example:

```
# hsctl config get cmc
HYPERSTORE OPTION          VALUE
cmc.applicationName         Cloudian
cmc.csrf.origin.allowlist   []
cmc.csrf.origin.check       false
cmc.dashboard.longRepairQueueWarning  false
etc...
```

You can also check the current values of all the settings in a setting sub-group. For example:

```
# hsctl config get s3.qos
HYPERSTORE OPTION  VALUE
s3.qos.bucketLevel  false
s3.qos.enabled      false
s3.qos.trackRate    false
```

Note Grouping relationships are reflected in the segmented setting names. Running *hsctl config get <segment>* (or *hsctl config get <segment>.<segment>*) will retrieve all settings with names that start with that segment or combination of segments.

To check the current values of **all HyperStore configuration settings**, run this command:

```
# hsctl config get all
```

This will return, in alphabetical order, all HyperStore settings and their current values.

For a list of HyperStore settings and setting groups see ["HyperStore Configuration Settings"](#) (page 700).

13.1.2. Changing the Value of a Setting and Applying Your Change

To change a HyperStore configuration setting and have your change take effect across the cluster, you will run a sequence of three *hsctl* commands on the Config Controller node:

```
# hsctl config set <setting>=<value>
# hsctl config apply <settingGroup>
# hsctl service restart <service> --nodes=ALL
```

For example:

```
# hsctl config set s3.crr.syncBucket=true
HYPERSTORE OPTION      VALUE
- s3.crr.syncBucket     false
+ s3.crr.syncBucket     true
# hsctl config apply s3
REGION      DATACENTER  NODE      ID      RESULT
ks-region1  DC1             centos79-scan  b38543a6-f222-40cf-9e80-6eb8db2ea4dc OK
# hsctl service restart s3 --nodes=ALL
REGION      DATACENTERS  NODES  SERVICE  ACTION  RESULT
ks-region1  N/A          N/A    s3       restart OK
```

Note that:

- The processing of the apply command and of the restart command may take some time, depending on how many nodes you have and whether those nodes are distributed across more than one data center (in the example above there is only one node). The command response will show the progress.
- In most cases, the `<settingGroup>` to which to apply the configuration will equal the first segment of the setting name (in the example above this is `s3`). For a small number of settings multiple setting groups need to be restarted for the change to take full effect. This can be accomplished with a single command, `hsctl config apply <settingGroup1> <settingGroup2>`.
- In most cases, the `<service>` to restart will also equal the first segment of the setting name (again in the example above this is `s3`). The exceptions are:
 - For Admin Service settings the first segment of the setting name is `admin` and the setting group to which to apply a change is also `admin`, but the service to restart is `s3`. This is because the Admin Service is restarted with the S3 Service, rather than being restarted independently.
 - For a small number of settings multiple services need to be restarted for the change to take full effect. This can be accomplished with a single command, `hsctl service restart <service1> <service2>`.

In this documentation the description of each individual setting shows the correct apply and restart commands to use for that setting. See ["HyperStore Configuration Settings" \(page 700\)](#).

If you want to **make changes to multiple settings within the same setting group**, you can do it efficiently by running several set commands (one command per setting) and then running just one apply command and one restart command, like this:

```
# hsctl config set <setting1>=<value>
# hsctl config set <setting2>=<value>
# hsctl config set <setting3>=<value>
# hsctl config apply <settingGroup>
# hsctl service restart <service> --nodes=ALL
```

If you've set multiple settings (by doing `hsctl config set` for each setting) and you want to **preview your changes** before doing `hsctl config apply`, you can do so with this command:

```
# hsctl config preview
```

This will show all settings that have been changed via `hsctl config set` but that have not yet been applied via `hsctl config apply`. For example:

```
# hsctl config preview
HYPERSTORE OPTION      VALUE
- s3.qos.enabled        false
- s3.qos.trackRate      false
```

```
+ s3.qos.enabled      true
+ s3.qos.trackRate    true
```

To get inline help for *hsctl config* and its sub-commands:

```
# hsctl config --help
```

or

```
# hsctl config <command> --help
```

13.1.3. Additional Options When Using *hsctl* for Configuration Control

- ["Specifying a Scope for a Configuration Change" \(page 696\)](#)
- ["Getting Cluster Topology Information" \(page 698\)](#)
- ["Selecting an 'hsctl config get' Output Format" \(page 699\)](#)
- ["Dumping and Importing Configuration Settings in YAML or JSON Formats" \(page 699\)](#)
- ["Getting Inline 'hsctl config' Help" \(page 699\)](#)

13.1.3.1. Specifying a Scope for a Configuration Change

Typically when you change a configuration setting, you do so for your HyperStore system as a whole. But for some settings, *hsctl* supports implementing your setting change for just a particular node or data center or service region. In general this applies to settings that relate to resource utilization or performance tuning. The data center and region scope options are of potential value only if you have a multi-DC or multi-region HyperStore system.

Although the method for implementing setting changes with a defined scope is straightforward, most likely this is something that you might do on advice from Cloudbian Support -- towards the goal of optimizing your HyperStore system's performance -- rather than on your own initiative.

To make a setting change for just one node:

```
# hsctl config set <setting>=<value> --node=<hostname>
# hsctl config apply <settingGroup> --nodes=<hostname>
# hsctl service restart <service> --nodes=<hostname>
```

To make a setting change for just the nodes in one data center:

```
# hsctl config set <setting>=<value> --datacenter=<datacenterName>
# hsctl config apply <settingGroup> --datacenters=<datacenterName>
# hsctl service restart <service> --datacenters=<datacenterName>
```

To make a setting change for just the nodes in one service region:

```
# hsctl config set <setting>=<value> --region=<regionName>
# hsctl config apply <settingGroup> --regions=<regionName>
# hsctl service restart <service> --regions=<regionName>
```

To retrieve a current setting value from just one node:

```
# hsctl config get <setting>=<value> --node=<hostname>
```

For a setting that supports specifying a scope, the setting can potentially have a value at each of four levels of granularity:

1. Global
2. Region
3. Data center
4. Node

The more granular level takes precedence over the less granular level -- so for example, a configuration value explicitly set for a particular node takes precedence over the values for the same setting at the data center, region, and global levels. If a setting's value has not been explicitly set at a more granular level, the value from the less granular level is inherited. For example, if a particular configuration setting has only been set at the global level (either by you or as an out-of-the-box system default value for that setting), then that same setting value applies across all regions, data centers, and nodes.

If you make a setting customization at more granular level and then subsequently you want to un-do that customization, you can use the `hstctl config unset` command. For example, to unset a node-specific customization:

```
# hstctl config unset <setting> --node=<hostname>
```

This results in that node using the setting value from the less granular level (the data center if the setting has been explicitly assigned a value at that level; or else the regional or global value for the setting).

Note that:

- For the `hstctl config get`, `hstctl config set`, and `hstctl config unset` commands, the scope flags are in the singular (`--node`, `--datacenter`, `--region`) while for `hstctl config apply` and `hstctl service restart` they're plural (`-nodes`, `--datacenters`, `--regions`). This is because, in a different context, the latter commands can support comma-separated lists for these scopes.
- An alternative way to perform a scope-limited `hstctl config set` is to precede the setting name with the scope like this:

```
# hstctl config set cluster.nodes.<hostname>.<setting>=<value>
# hstctl config set cluster.datacenters.<datacentername>.<setting>=<value>
# hstctl config set cluster.regions.<regionname>.<setting>=<value>
```

For example:

```
# hstctl config set cluster.datacenters.DC2.s3.maxThreads=75
```

- If you try to set a scoped value for a setting that does not support scope limitation -- a setting that only supports global implementation -- you will get an error such as in this example:

```
# hstctl config set s3.qos.enabled=true --node=vm222
the HyperStore Parameter "cluster.nodes.vm222.s3.qos.enabled" does not exist.
```

The "zone" Scope

HyperStore uses a special configuration scope called a "zone" if you install one or more [File Service](#) pools in your HyperStore system, or if you install a [Search Service](#) pool. A File Service pool consists of either one File Service node or three File Service nodes, and if you wish you can install multiple File Service pools in your HyperStore system. A Search Service pool consists of just one node, and you can install no more than one Search Service pool.

For each File Service or Search Service pool that you install, in the HyperStore configuration schema a zone is automatically created to contain settings for that service pool. The zone is automatically given the same name that you gave to the service pool when you installed it, and the settings stored under the zone are automatically derived from the information that you supply to HyperStore during the File Service pool or Search Service pool installation process.

Note These File Service pool settings have to do with how HyperStore object nodes integrate with the File Service pools. They are not the same as the [File Service settings that you can configure through the CMC](#). The latter settings are stored on the File Service nodes and are not managed by the *hscctl* tool.

If you have installed one or more File Service or Search Service pools in your HyperStore system, you can view the zone-contained settings for those service pools with this *hscctl* command:

```
# hscctl config get cluster.zones
```

Below is an example of the *hscctl config get cluster.zones* response, for a system that has two File Service pools (and no Search Service pool):

```
# hscctl config get cluster.zones
HYPERSTORE OPTION                                VALUE
cluster.zones.file-pool-1.file.auth.password      .....
cluster.zones.file-pool-1.file.auth.username      admin
cluster.zones.file-pool-1.file.enabled             true
cluster.zones.file-pool-1.file.kubernetes.apiAddress 10.130.41.36
cluster.zones.file-pool-1.file.loadBalancer.addressPool 10.130.41.32-35
cluster.zones.file-pool-1.name                     file-pool-1
cluster.zones.file-pool-2.file.auth.password      .....
cluster.zones.file-pool-2.file.auth.username      admin
cluster.zones.file-pool-2.file.enabled             true
cluster.zones.file-pool-2.file.kubernetes.apiAddress 10.130.41.41
cluster.zones.file-pool-2.file.loadBalancer.addressPool 10.130.41.37-40
cluster.zones.file-pool-2.name                     file-pool-2
```

These settings are controlled by the system and are not intended to be user-editable via *hscctl config*.

However, for a File Service pool you can change the *cluster.zones.<file-pool-name>.auth.password* value -- the password that the CMC uses to access the File Service API -- by using an *hscctl* command dedicated to that purpose. For more information see ["File Connection"](#) (page 511).

To retrieve the current API password for a File Service pool you can run this command:

```
# hscctl config get cluster.zones.<file-pool-name>.file.auth.password
```

For example:

```
# hscctl config get cluster.zones.file-pool-1.file.auth.password
```

13.1.3.2. Getting Cluster Topology Information

You can view a variety of configuration information for your HyperStore cluster topology by running this command:

```
# hscctl config get cluster
```

Of particular note among the cluster configuration settings:

- *cluster.nodes.<hostname>.ID* -- System-generated unique ID of the host. Each host in your HyperStore system has a system-generated ID. You may see these IDs in certain messages generated by the Salt configuration management layer.

- `cluster.datacenters.<datacenter-name>.ID` -- System-generated unique ID of the data center. Important because the system does not prevent data centers having the same name if they are in different service regions.
- `cluster.nodes.<hostname>.roles` -- Comma-separated list of service roles played by the host. This includes service roles common to all hosts -- such as `s3` and `cassandra` -- as well as specialized roles that only some or one host play, such as `redisCredReplica`, `ntpPrimary` or `saltMasterPrimary` (the Config Controller node), and roles that can only be played by hosts in the default service region such as `iam` and `sqfs`.

13.1.3.3. Selecting an 'hsctl config get' Output Format

By default `hsctl config get <setting>` commands return the results in columnized text format. Alternatively you can choose to have the results returned in YAML or JSON format:

```
# hsctl config get <setting or setting group> --out=yaml
```

```
# hsctl config get <setting or setting group> --out=json
```

13.1.3.4. Dumping and Importing Configuration Settings in YAML or JSON Formats

If you are experienced in working with YAML or JSON formatted configuration files and you want to implement configuration changes by working with one of those file types, you can dump the configuration of a specified setting group to either a YAML file or a JSON file:

```
# hsctl config dump <setting group> --out=<output format>
```

For example:

```
# hsctl config dump s3 --out=yaml
```

```
# hsctl config dump cmc --out=json
```

The first command would produce a file `s3.yaml` in the working directory. The second command would produce a file `cmc.json` in the working directory.

You could then edit the dumped file, and then import the edited file back in through `hsctl`:

```
# hsctl config import <filename>
```

You would then need to apply your change and restart the appropriate service:

```
# hsctl config apply <settingGroup>
```

```
# hsctl service restart <service> --nodes=ALL
```

13.1.3.5. Getting Inline 'hsctl config' Help

Inline help is available for `hsctl config`:

```
# hsctl config --help
```

Or for individual `hsctl config` commands, for example:

```
# hsctl config get --help
```

```
# hsctl config set --help
```

```
# hsctl config apply --help
```

13.2. HyperStore Configuration Settings

Listed here are HyperStore system configuration settings. You can control these settings with the `hstctl` tool (for an overview of this tool's configuration management functionality see ["Using hstctl to Control HyperStore Configuration" \(page 693\)](#)). The settings are organized into setting groups, which in most cases correspond to a particular HyperStore service. Where appropriate, settings are further organized into sub-groups. You can check the current value of settings in your system by running any of these commands on the Config Controller node:

- `hstctl config get <setting group>`
- `hstctl config get <setting sub-group>`
- `hstctl config get <setting>`

For information on how to change a particular setting and apply your change to the system, see the descriptions of individual settings.

- ["Admin Service Settings" \(page 700\)](#)
- ["Metadata DB Service Settings" \(page 708\)](#)
- ["CMC Service Settings" \(page 715\)](#)
- ["Common Settings" \(page 741\)](#)
- ["HyperStore Firewall" \(page 850\)](#)
- ["HyperStore Service Settings" \(page 761\)](#)
- ["IAM Service Settings" \(page 789\)](#)
- ["S3 Service Settings" \(page 811\)](#)
- ["SQS Service Settings" \(page 841\)](#)
- [SSH Service Settings](#)

13.2.1. Admin Service Settings

The table below lists configuration settings in the `admin` setting group, which configures the [Admin Service](#). When applicable the settings in the table are organized into sub-groups.

Setting Sub-Group	Setting	Default Value
admin.auth	admin.auth.enabled (page 701)	true
	admin.auth.password (page 701)	
	admin.auth.realm (page 702)	CloudianAdmin
	admin.auth.username (page 702)	sysadmin
admin.heap	admin.heap.auto.initPercent (page 703)	25
	admin.heap.auto.limit (page 703)	16g
	admin.heap.auto.maxPercent (page 704)	5
	admin.heap.maxSize (page 704)	
	admin.heap.minSize (page 704)	

Setting Sub-Group	Setting	Default Value
admin.logging	admin.logging.level (page 704)	INFO
	admin.logging.rollover.* settings	see "Log Rolling Settings" (page 689) for default values
admin.ports	admin.ports.loadBalancer.http (page 705)	18081
	admin.ports.loadBalancer.https (page 706)	19443
	admin.ports.main.http (page 706)	18081
	admin.ports.main.https (page 706)	19443
admin.threads	admin.threads.idleTimeout (page 707)	60000
	admin.threads.max (page 707)	50
	admin.threads.min (page 707)	10
admin.tls	admin.tls.required (page 708)	true
	admin.tls.verifyCA (page 708)	false
n/a (miscellaneous individual settings)	admin.billing.allowlist.enabled (page 702)	true
	admin.dnsName (page 703)	s3-admin.mycloudianhyperstore.com
	admin.lowRes.idleTimeout (page 705)	5000
	admin.stackSize (page 706)	

13.2.1.1. admin.* Setting Descriptions

admin.auth.enabled

[Former name in HyperStore 7.x and earlier: `admin_auth_enabled`]

Whether to have the Admin Service require HTTP(S) Basic Authentication from connecting clients. Set to *true* to have the Admin Service require Basic Authentication, or *false* to not require it.

Default = true (exception: if your original HyperStore install was older than version 6.0.2, your system's default = false)

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set admin.auth.enabled=<boolean>
# hscctl config apply all
# hscctl service restart s3 cmc iam --nodes=ALL
```

admin.auth.password

[Former name in HyperStore 7.x and earlier: `admin_auth_pass`]

If the Admin Service is configured to require HTTP(S) Basic Authentication from clients (if "**admin.auth.enabled**" (page 701) is set to *true*), this is the password for clients to use when submitting HTTP(S) requests to the Admin Service.

Do not use special characters in the password.

Default = A random password generated by the system upon installation (exception: if your original HyperStore install was older than version 7.2.2, the default in your system = public)

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set admin.auth.password=<secret>
# hsctl config apply all
# hsctl service restart s3 cmc iam --nodes=ALL
```

Note Although you supply the password as clear text when you configure this setting, the system stores the password in base64 encoding and the clear text value is only ever readable by the root user. If a HyperStore Shell user retrieves the value it will display as `*****`.

admin.auth.realm

[Former name in HyperStore 7.x and earlier: *admin_auth_realm*]

If the Admin Service is configured to require HTTP(S) Basic Authentication from clients (if "**admin.auth.enabled**" (page 701) is set to *true*), this is the realm name used by the Admin Service for Basic Authentication purposes.

Default = CloudianAdmin

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set admin.auth.realm=<string>
# hsctl config apply all
# hsctl service restart s3 cmc iam --nodes=ALL
```

admin.auth.username

[Former name in HyperStore 7.x and earlier: *admin_auth_user*]

If the Admin Service is configured to require HTTP(S) Basic Authentication from clients (if "**admin.auth.enabled**" (page 701) is set to *true*), this is the username for clients to use when submitting HTTP(S) requests to the Admin Service.

Default = sysadmin

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set admin.auth.username=<string>
# hsctl config apply all
# hsctl service restart s3 cmc iam --nodes=ALL
```

admin.billing.allowlist.enabled

[Former name in HyperStore 7.x and earlier: *admin_whitelist_enabled*]

Whether to enable the billing "allowlist" feature by which favorable billing terms can be set for a specified list of source IP addresses or subnets. If this feature is enabled, allowlist management functionality displays in the CMC (for system administrators only).

Default = true

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set admin.billing.allowlist.enabled=<boolean>
# hsctl config apply admin cmc
# hsctl service restart s3 cmc --nodes=ALL
```

admin.dnsName

[Former name in HyperStore 7.x and earlier: cmc_admin_host_ip]

Fully qualified domain name for the Admin Service. The CMC connects to this service and calls the Admin API to implement various administrative actions that you initiate through the CMC UI.

If you have multiple service regions for your HyperStore system, this FQDN must be the one for the Admin Service in your default service region.

Default = Derived from operator input during installation.

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set admin.dnsName=<string>
# hsctl config apply admin cmc
# hsctl service restart s3 cmc --nodes=ALL
```

admin.heap.auto.initPercent

[Former name in HyperStore 7.x and earlier: cloudian_admin_init_heap_percent]

Initial heap size (JAVA_OPTS -Xms value) for the Admin Service application, as a percentage of the -Xmx value.

Default = 25

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set admin.heap.auto.initPercent=<integer>
# hsctl config apply admin
# hsctl service restart s3 --nodes=ALL
```

admin.heap.auto.limit

[Former name in HyperStore 7.x and earlier: cloudian_admin_heap_limit]

Maximum heap size limit for the Admin Service application. The JAVA_OPTS -Xmx value passed to the JVM will be the **lower** of this value and the percent-of-system-RAM value set by "[admin.heap.auto.maxPercent](#)" (page 704).

Default = 16g

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hctl config set admin.heap.auto.limit=<string>
# hctl config apply admin
# hctl service restart s3 --nodes=ALL
```

admin.heap.auto.maxPercent

[Former name in HyperStore 7.x and earlier: *cloudian_admin_max_heap_percent*]

Maximum heap size for the Admin Service application, as a percentage of host system RAM. The `JAVA_OPTS -Xmx` value passed to the JVM will be the **lower** of this value and the limiting value set by "[admin.heap.auto.limit](#)" (page 703).

Default = 5

To change this setting and apply your change, run the following *hctl* commands on the Config Controller node:

```
# hctl config set admin.heap.auto.maxPercent=<integer>
# hctl config apply admin
# hctl service restart s3 --nodes=ALL
```

admin.heap.maxSize

On each node, the memory heap maximum size setting (Java *Xmx* setting) for the Admin Service application.

If this is explicitly set, it overrides the upper limit derived from the *admin.heap.auto.** settings.

Default = not set

To change this setting and apply your change, run the following *hctl* commands on the Config Controller node:

```
# hctl config set admin.heap.maxSize=<string>
# hctl config apply admin
# hctl service restart s3 --nodes=ALL
```

admin.heap.minSize

On each node, the memory heap minimum size setting (Java *Xms* setting) for the Admin Service application.

If this is explicitly set, it overrides the lower limit derived from the *admin.heap.auto.** settings.

Default = not set

To change this setting and apply your change, run the following *hctl* commands on the Config Controller node:

```
# hctl config set admin.heap.minSize=<string>
# hctl config apply admin
# hctl service restart s3 --nodes=ALL
```

admin.logging.level

This setting controls the level at which Admin Service application logging will occur. Only messages with a severity level that matches or exceeds your configured logging level will be logged.

Listed in order from the least inclusive logging to the most inclusive logging, the supported logging levels are:

- OFF -- The OFF "level" turns off logging.
- FATAL -- The FATAL level designates very severe error events that will likely lead the application to abort.
- ERROR -- The ERROR level designates error events that may still allow the application to continue running.
- WARN -- The WARN level designates potentially harmful situations.
- INFO -- The INFO level designates informational messages that indicate the progress of the application at coarse-grained level.
- DEBUG -- The DEBUG level designates fine-grained informational events that are most useful to debug an application.
- TRACE -- The TRACE level designates finer-grained informational events than the DEBUG level.

For example, if this setting is set to WARN then only messages with severity levels WARN, ERROR, or FATAL will be logged by the Admin Service.

Default = INFO

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set admin.logging.level=<string>
# hsctl config apply admin
# hsctl service restart s3 --nodes=ALL
```

admin.lowRes.idleTimeout

[Former name in HyperStore 7.x and earlier: *cloudian_s3admin_lowres_maxidletime*]

Special, "low resource" maximum idle time to apply to Admin Service HTTP(S) connections when the number of simultaneous connections to an Admin Service node exceeds 1000. Configured in milliseconds. With this setting, you can have the Admin Service be less tolerant of connection idle time during times of high concurrent usage. (For a general description of idle timer behavior, see ["admin.threads.idleTimeout"](#) (page 707).)

Default = 5000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set admin.lowRes.idleTimeout=<integer>
# hsctl config apply admin
# hsctl service restart s3 --nodes=ALL
```

admin.ports.loadBalancer.http

The Admin Service HTTP port on the load balancer.

Default = 18081

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set admin.ports.loadBalancer.http=<integer>
# hsctl config apply all
# hsctl service restart s3 cmc --nodes=ALL
```

admin.ports.loadBalancer.https

[Former name in HyperStore 7.x and earlier: *cmc_admin_secure_port*]

The Admin Service HTTPS port on the load balancer.

Default = 19443

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set admin.ports.loadBalancer.https=<integer>
# hsctl config apply all
# hsctl service restart s3 cmc --nodes=ALL
```

admin.ports.main.http

[Former name in HyperStore 7.x and earlier: *cloudian_s3_admin_port*]

On each node, the port on which the Admin Service listens for regular HTTP connections.

Default = 18081

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set admin.ports.main.http=<integer>
# hsctl config apply all
# hsctl service restart s3 firewall cmc --nodes=ALL
```

admin.ports.main.https

[Former name in HyperStore 7.x and earlier: *cmc_admin_secure_port*]

On each node, the port on which the Admin Service listens for HTTPS connections.

Default = 19443

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set admin.ports.main.https=<integer>
# hsctl config apply all
# hsctl service restart s3 firewall cmc --nodes=ALL
```

Note You also have the option of configuring this setting by using the installer's "Advanced Configuration Options", as described in ["Changing S3, Admin, or CMC Listening Ports"](#) (page 858).

admin.stackSize

Memory stack size to use for the Admin Service.

If this setting is not explicitly set, the Admin Service uses the memory stack size specified by the ["java.stackSize"](#) (page 797) setting.

Default = not set (so uses *java.stackSize* value)

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set admin.stackSize=<string>
# hsctl config apply admin
# hsctl service restart s3 --nodes=ALL
```

admin.threads.idleTimeout

[Former name in HyperStore 7.x and earlier: cloudian_s3admin_max_idletime]

When the Admin Service processes HTTP(S) requests from clients, the maximum allowed connection idle time in milliseconds. If this much idle time passes before a new request is received over an open connection with a client, or if this much idle time passes during the reading of headers and content for a request, or if this much idle time passes during the writing of headers and content of a response, the connection with the client is closed.

Default = 60000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set admin.threads.idleTimeout=<integer>
# hsctl config apply admin
# hsctl service restart s3 --nodes=ALL
```

admin.threads.max

[Former name in HyperStore 7.x and earlier: cloudian_s3admin_max_threads]

On each node, the maximum number of threads to allow in the thread pool that the Admin Service uses to process incoming HTTP(S) requests from clients. When there is a new HTTP(S) request processing task and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

Default = 50

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set admin.threads.max=<integer>
# hsctl config apply admin
# hsctl service restart s3 --nodes=ALL
```

admin.threads.min

[Former name in HyperStore 7.x and earlier: cloudian_s3admin_min_threads]

On each node, the minimum number of threads to keep in the thread pool that the Admin Service uses to process incoming HTTP(S) requests from clients.

The Admin Service uses a thread pool to process incoming HTTP(S) requests from clients. An initial pool of threads is created at server initialization time, and additional threads may be added as needed (within the limit set by "[admin.threads.max](#)" (page 707)). Idle threads in the pool are terminated if not used within a thread timeout period — unless the number of threads in the pool is down to *admin.threads.min*. If only this many threads are in the pool, then threads are kept open even if they've been idle for longer than the thread timeout.

Default = 10

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set admin.threads.min=<integer>
# hsctl config apply admin
# hsctl service restart s3 --nodes=ALL
```

admin.tls.required

[Former name in HyperStore 7.x and earlier: `admin_secure`]

Whether clients -- including other HyperStore services such as the CMC -- will be required to use HTTPS when connecting to the Admin Service. Other HyperStore services read this setting to determine whether they must use HTTPS to connect to the Admin Service.

Default = true

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set admin.tls.required=<boolean>
# hsctl config apply all
# hsctl service restart s3 cmc iam --nodes=ALL
```

admin.tls.verifyCA

[Former name in HyperStore 7.x and earlier: `cmc_admin_secure_ssl`]

Whether the Admin Service will require clients connecting over HTTPS to verify the CA that signed the Admin Service's TLS certificate -- *true* or *false*.

Note Note that the TLS certificate that is used with the Admin Service by default is self-signed and so by default this parameter is set to *false*. For information about using a CA-signed certificate for the Admin Service (or for other services), see ["Security and Privacy Features"](#) (page 177).

Default = false

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set admin.tls.verifyCA=<boolean>
# hsctl config apply all
# hsctl service restart s3 cmc iam --nodes=ALL
```

13.2.2. Metadata DB Service Settings

The table below lists configuration settings in the `cassandra` setting group, which configures the [Metadata DB Service](#). When applicable the settings in the table are organized into sub-groups.

Setting Sub-Group	Setting	Default Value
cassandra.directory	cassandra.directory.commitLog (page 710)	/var/lib/cassandra/commit
	cassandra.directory.data (page 710)	/var/lib/cassandra/data
	cassandra.directory.log (page 710)	/var/log/cassandra
	cassandra.directory.savedCaches (page 710)	/var/lib/cassandra/saved_caches
cassandra.heap	cassandra.heap.dumpOnOOM (page 711)	true
	cassandra.heap.maxSize (page 711)	empty
	cassandra.heap.minSize (page 712)	empty
cassandra.timeout	cassandra.timeout.rangeRequest (page 712)	10000
	cassandra.timeout.readRequest (page 712)	5000
	cassandra.timeout.request (page 713)	10000
	cassandra.timeout.writeRequest (page 713)	2000
cassandra.tombstone	cassandra.tombstone.gcGrace (page 713)	0
	cassandra.tombstone.threshold.cleanup (page 714)	75000
	cassandra.tombstone.threshold.fail (page 714)	100000
	cassandra.tombstone.threshold.warn (page 714)	50000
n/a (miscellaneous individual settings)	cassandra.concurrentCompactors (page 709)	2
	cassandra.enableGCLog (page 711)	true
	cassandra.stackSize (page 712)	empty
	cassandra.startTimeout	7200

13.2.2.1. cassandra.* Setting Descriptions

cassandra.concurrentCompactors

[Former name in HyperStore 7.x and earlier: `concurrent_compactors`]

Number of simultaneous [Cassandra compactions](#) to allow per node, not including validation "compactions" for anti-entropy repair. Simultaneous compactions can help preserve read performance in a mixed read/write workload, by mitigating the tendency of small SSTables to accumulate during a single long running compaction.

Default = 2

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set cassandra.concurrentCompactors=<integer>
# hscctl config apply cassandra
# hscctl service restart cassandra --nodes=ALL
```

cassandra.directory.commitLog

[Former name in HyperStore 7.x and earlier: `cassandra_commit_log_directory`]

On each node, the directory in which to store the Cassandra commit log file. All data written to a Cassandra node is written to the append-only commit log before being written to a Memtable (and then flushed to an immutable SSTable).

Default = Derived from operator input during installation, if host has multiple disks. For hosts with only one disk, default is `/var/lib/cassandra_commit`

IMPORTANT ! Changing this setting after your system has been in use can potentially result in data loss. If for some reason you are wanting to change this setting, first consult with Clouddian Support.

cassandra.directory.data

[Former name in HyperStore 7.x and earlier: `cassandra_data_directory`]

On each node, the directory in which to store Cassandra data files (SSTable files). In HyperStore, Cassandra is used for storing S3 object metadata as well as service metadata such as account information, usage data, and system monitoring data.

Default = Derived from operator input during installation, if host has multiple disks. For hosts with only one disk, default is `/var/lib/cassandra/data`

IMPORTANT ! Changing this setting after your system has been in use can potentially result in data loss. If for some reason you are wanting to change this setting, first consult with Clouddian Support.

IMPORTANT ! The Cassandra data directory should **not** be mounted on a shared file system such as a NAS device.

cassandra.directory.log

[Former name in HyperStore 7.x and earlier: `cassandra_log_directory`]

On each node, the directory in which to store Cassandra application log files.

Default = `/var/log/cassandra`

IMPORTANT ! If for some reason you are wanting to change this setting on a system that is already in use, first consult with Clouddian Support.

cassandra.directory.savedCaches

[Former name in HyperStore 7.x and earlier: `cassandra_saved_cache_directory`]

On each node, the directory in which to store the Cassandra key cache files.

Default = `/var/lib/cassandra/saved_caches`

IMPORTANT ! If for some reason you are wanting to change this setting on a system that is already in use, first consult with Cloudbian Support.

cassandra.enableGCLog

[Former name in HyperStore 7.x and earlier: *cassandra_enable_gc_logging*]

Whether to enable Java garbage collection (GC) logging for Cassandra. On each node the log is written to */var/log/cassandra/gc.log*.

Default = true

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cassandra.enableGCLog=<boolean>
# hsctl config apply cassandra
# hsctl service restart cassandra --nodes=ALL
```

Note Separately, Java GC logging is enabled for the S3 Service, Admin Service, and HyperStore Service. On each node these GC logs are in the */var/log/cloudian* directory and are named *s3-gc.log*, *admin-gc.log*, and *hss-gc.log*, respectively.

cassandra.heap.dumpOnOOM

[Former name in HyperStore 7.x and earlier: *cassandra_heapdump_on_out_of_memory*]

Whether to enable Java heap dumping for Cassandra out-of-memory (OOM) errors. On each node the dump path is */var/log/cassandra/cassandra.hprof*. The dump file is in HPROF binary format.

Default = true

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cassandra.heap.dumpOnOOM=<boolean>
# hsctl config apply cassandra
# hsctl service restart cassandra --nodes=ALL
```

cassandra.heap.maxSize

[Former name in HyperStore 7.x and earlier: *cassandra_max_heap_size*]

On each node, the memory heap maximum size setting (Java *Xmx* setting) for the Cassandra Service application.

If this is explicitly set, it overrides the maximum that is automatically derived from host characteristics.

Default = not set

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cassandra.heap.maxSize=<string>
# hsctl config apply cassandra
# hsctl service restart cassandra --nodes=ALL
```

cassandra.heap.minSize

On each node, the memory heap minimum size setting (Java *Xms* setting) for the Cassandra Service application.

If this is explicitly set, it overrides the minimum that is automatically derived from host characteristics.

Default = not set

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cassandra.heap.minSize=<string>
# hsctl config apply cassandra
# hsctl service restart cassandra --nodes=ALL
```

cassandra.stackSize

Memory stack size to use for Cassandra (the Metadata DB).

If this setting is not explicitly set, Cassandra uses the memory stack size specified by the ["java.stackSize" \(page 797\)](#) setting.

Default = not set (so uses *java.stackSize* value)

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cassandra.stackSize=<string>
# hsctl config apply cassandra
# hsctl service restart cassandra --nodes=ALL
```

cassandra.timeout.rangeRequest

The number of milliseconds that the Metadata DB (Cassandra) coordinator node waits for sequential or index scans to complete before timing out. In the case of a timeout, a timeout exception is returned to the client. The client will always be a HyperStore service, such as the S3 Service or the Admin Service. End user client applications do not directly interface with the Metadata DB.

Default = 10000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cassandra.timeout.rangeRequest=<integer>
# hsctl config apply cassandra
# hsctl service restart cassandra --nodes=ALL
```

cassandra.timeout.readRequest

The number of milliseconds that the Metadata DB (Cassandra) coordinator node waits for a read operation to complete before timing out. In the case of a timeout, a timeout exception is returned to the client. The client will always be a HyperStore service, such as the S3 Service or the Admin Service. End user client applications do not directly interface with the Metadata DB.

Default = 5000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cassandra.timeout.readRequest=<integer>
# hsctl config apply cassandra
# hsctl service restart cassandra --nodes=ALL
```

cassandra.timeout.request

The request completion timeout (in number of milliseconds) that the Metadata DB (Cassandra) coordinator node applies to miscellaneous operation types for which an operation-specific timeout is not configured by a different setting (such as the "[cassandra.timeout.readRequest](#)" (page 712) setting which configures a timeout for Cassandra read requests).

In the case of a timeout, a timeout exception is returned to the client. The client will always be a HyperStore service, such as the S3 Service or the Admin Service. End user client applications do not directly interface with the Metadata DB.

Default = 10000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cassandra.timeout.request=<integer>
# hsctl config apply cassandra
# hsctl service restart cassandra --nodes=ALL
```

cassandra.timeout.writeRequest

The number of milliseconds that the Metadata DB (Cassandra) coordinator node waits for a write operation to complete before timing out. In the case of a timeout, a timeout exception is returned to the client. The client will always be a HyperStore service, such as the S3 Service or the Admin Service. End user client applications do not directly interface with the Metadata DB.

Default = 2000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cassandra.timeout.writeRequest=<integer>
# hsctl config apply cassandra
# hsctl service restart cassandra --nodes=ALL
```

cassandra.tombstone.gcGrace

[Former name in HyperStore 7.x and earlier: `cassandra_tombstone_gcgrace`]

While doing a [cleanup of tombstones](#) (markers indicating deleted data), the system will not purge tombstones that are fewer than this many seconds old.

If this is set to 0, then no tombstones are exempted from the purge.

Default = 0

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cassandra.tombstone.gcGrace=<integer>
# hsctl config apply cassandra
# hsctl service restart cassandra --nodes=ALL
```

cassandra.tombstone.threshold.cleanup

[Former name in HyperStore 7.x and earlier: `cassandra_tombstone_cleanup_threshold`]

If a Cassandra query encounters more than this many tombstones (markers indicating deleted data), a tombstone purge process is automatically triggered for that column family.

An example of query that can potentially encounter a high number of tombstones is a metadata query triggered by an S3 *ListObjects* operation (which returns a list of all the objects in a bucket).

Default = 75000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cassandra.tombstone.threshold.cleanup=<integer>
# hsctl config apply cassandra
# hsctl service restart cassandra --nodes=ALL
```

Note You can also manually trigger a tombstone purge for a specific bucket, as described in "[System cron Jobs](#)" (page 632).

cassandra.tombstone.threshold.fail

[Former name in HyperStore 7.x and earlier: `cassandra_tombstone_failure_threshold`]

If a Cassandra query encounters more than this many tombstones (markers indicating deleted data), the query fails and a tombstone error is logged in the [Cassandra application log](#).

An example of query that can potentially encounter a high number of tombstones is a metadata query triggered by an S3 *ListObjects* operation (which returns a list of all the objects in a bucket).

Default = 100000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cassandra.tombstone.threshold.fail=<integer>
# hsctl config apply cassandra
# hsctl service restart cassandra --nodes=ALL
```

cassandra.tombstone.threshold.warn

[Former name in HyperStore 7.x and earlier: `cassandra_tombstone_warn_threshold`]

If while processing a Cassandra query it is found that a single row within a column family has more than this many tombstones (markers indicating deleted data), a tombstone warning is logged in the [Cassandra application log](#).

An example of query that can potentially encounter a high number of tombstones is a metadata query triggered by an S3 *ListObjects* operation (which returns a list of all the objects in a bucket).

Default = 50000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cassandra.tombstone.threshold.warn=<integer>
# hsctl config apply cassandra
# hsctl service restart cassandra --nodes=ALL
```

13.2.3. CMC Service Settings

The table below lists configuration settings in the *cmc* setting group, which configures the Clouddian Management Console Service. When applicable the settings in the table are organized into sub-groups.

Setting Sub-Group	Setting	Default Value
cmc.csrf	cmc.csrf.origin.allowlist (page 717)	[]
	cmc.csrf.origin.check (page 718)	false
cmc.groupList	cmc.groupList.enabled (page 719)	true
	cmc.groupList.maxSize (page 720)	100
cmc.logging	cmc.logging.level (page 720)	INFO
	cmc.logging.rollover.* settings	see " Log Rolling Settings " (page 689) for default values
cmc.login	cmc.login.adminCheckbox.enabled (page 721)	true
	cmc.login.gate.buttonConfirm (page 721)	
	cmc.login.gate.message (page 722)	
	cmc.login.gate.size (page 722)	1
	cmc.login.gate.title (page 722)	
	cmc.login.groupList.enabled (page 722)	true
	cmc.login.langSelection.enabled (page 723)	true
cmc.ports	cmc.ports.loadBalancer.http (page 725)	8888
	cmc.ports.loadBalancer.https (page 725)	8443
	cmc.ports.main.http (page 725)	8888
	cmc.ports.main.https (page 726)	8443
cmc.sso	cmc.sso.enabled (page 727)	false
	cmc.sso.sharedKey (page 727)	
	cmc.sso.tolerance (page 727)	3600000
cmc.threads	cmc.threads.idleTimeout (page 728)	20000
	cmc.threads.max (page 728)	150
	cmc.threads.min (page 728)	25
cmc.tiering	cmc.tiering.allowCustomEndpoint (page 729)	false
	cmc.tiering.allowPerBucketCredentials (page 729)	true
	cmc.tiering.defaultEndpointURL (page 730)	https://s3.amazonaws.com
	cmc.tiering.enabled (page 731)	false
cmc.ui	cmc.ui.account.credentials.accessKeyGrantees (page 732)	SystemAdmin,GroupAdmin,User

Setting Sub-Group	Setting	Default Value
	cmc.ui.account.credentials.grantees (page 732)	SystemAdmin,GroupAdmin,User
	cmc.ui.account.credentials.signinGrantees (page 732)s	SystemAdmin,GroupAdmin,User
	cmc.ui.account.profile.editGrantees (page 733)	SystemAdmin,GroupAdmin,User
	cmc.ui.account.view.activityGrantees (page 733)	SystemAdmin
	cmc.ui.account.view.usageGrantees (page 734)s	SystemAdmin,GroupAdmin,User
	cmc.ui.admin.manageGroups.createGrantees (page 734)	SystemAdmin
	cmc.ui.admin.manageGroups.deleteGrantees (page 734)	SystemAdmin
	cmc.ui.admin.manageGroups.edit.defaultQOSGrantees (page 735)	SystemAdmin,GroupAdmin
	cmc.ui.admin.manageGroups.edit.grantees (page 735)	SystemAdmin,GroupAdmin
	cmc.ui.admin.manageGroups.grantees (page 736)	SystemAdmin,GroupAdmin
	cmc.ui.admin.manageUsers.createGrantees (page 736)	SystemAdmin,GroupAdmin
	cmc.ui.admin.manageUsers.deleteGrantees (page 737)	SystemAdmin,GroupAdmin
	cmc.ui.admin.manageUsers.edit.credentialsGrantees (page 737)	SystemAdmin,GroupAdmin
	cmc.ui.admin.manageUsers.edit.grantees (page 738)	SystemAdmin,GroupAdmin
	cmc.ui.admin.manageUsers.edit.qosGrantees (page 738)	SystemAdmin,GroupAdmin
	cmc.ui.admin.manageUsers.grantees (page 739)	SystemAdmin,GroupAdmin
	cmc.ui.admin.manageUsers.view.objectDataGrantees (page 739)	[]
	cmc.ui.bucket.view.storagePolicyGrantees (page 740)	SystemAdmin,GroupAdmin,User
	cmc.ui.page.defaultSize (page 740)	10
	cmc.ui.page.maxSize (page 740)	100
	cmc.ui.security.serviceInfo (page 740)	true
n/a (miscellaneous individual settings)	cmc.applicationName (page 717)	Cloudian
	cmc.dashboard.longRepairQueueWarning (page 718)	false
	cmc.dnsName (page 718)	cmc.mycloudianhyperstore.com
	cmc.enabled (page 719)	true
	cmc.externalCRR.enabled (page 719)	false
	cmc.mfa.enforced (page 723)	false

Setting Sub-Group	Setting	Default Value
	cmc.pathStyleAccess.enabled (page 724)	true
	cmc.purgeBucket.enabled (page 726)	false
	cmc.session.timeout (page 727)	30
	cmc.tls.required (page 731)	true

13.2.3.1. cmc.* Setting Descriptions

cmc.applicationName

[Former name in HyperStore 7.x and earlier: *cmc_application_name*]

Name of the CMC web application, to be displayed in the URL paths for the various CMC UI pages. The CMC's URL paths are in the form *https://<host>:<port>/<application_name>/<page>.htm*.

Use only alphanumeric characters. Do not use spaces, dashes, underscores, or other special characters.

Default = Cloudian (and so URL paths are in form *https://<host>:<port>/Cloudian/<page>.htm*. For example *https://enterprise2.com:8443/Cloudian/dashboard.htm*).

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.applicationName=<string>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.csrf.origin.allowlist

[Former name in HyperStore 7.x and earlier: *cmc_csrf_origin_allowlist*]

If you set "[cmc.csrf.origin.check](#)" (page 718) to *true* and you have the CMC behind a proxy or a load balancer, use the *cmc.csrf.origin.allowlist* setting to specify the CMC domain(s) associated with the proxy or load balancer (the CMC FQDN[s] mapped to the proxy or load balancer in your DNS configuration). This is necessary because in this environment, when the CMC application applies the origin check to guard against CSRF, the origin domain in incoming HTTP requests should match the CMC domain associated with the proxy or load balancer -- rather than matching the internal CMC application domain to which the proxy or load balancer forwards the requests.

When configuring *cmc.csrf.origin.allowlist*, include the transfer protocol along with the FQDN -- for example *https://cmc.domain.com*. If there are multiple CMC domains associated with your proxies or load balancers, configure *cmc.csrf.origin.allowlist* as a vertical bar separated list -- for example *https://cmc.domain1.com|https://cmc.domain2.com*.

Default = empty

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.csrf.origin.allowlist=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.csrf.origin.check

[Former name in HyperStore 7.x and earlier: *cmc_csrf_origin_check_enabled*]

Set this to *true* if you want the CMC to implement request origin checks as a safeguard against Cross-Site Request Forgery (CSRF). CSRF exploits users who, while logged in to a secure web application such as the CMC, are concurrently doing things like checking their email or surfing web sites in other browser tabs. CSRF tricks such users into clicking on seemingly harmless links or images that are designed to submit malicious requests to the targeted web application without the user's knowledge. The malicious requests, sent from the user's browser, can reach the secure web application because the user has already successfully logged in.

If you set *cmc.csrf.origin.check* to *true* then the CMC will check standard HTTP request headers to confirm that, apart from login requests, all other requests to the CMC domain are originating from the CMC domain (as would be expected for legitimate requests, such as if the user is clicking links within the CMC or taking actions within a CMC page). If an HTTP request's origin domain doesn't match the target domain (the CMC domain), the CMC will reject the request.

If you set *cmc.csrf.origin.check* to *true* and you have the CMC behind a proxy or a load balancer, then you must also configure the "**cmc.csrf.origin.allowlist**" (page 717) setting.

Default = false

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.csrf.origin.check=<boolean>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.dashboard.longRepairQueueWarning

[Former name in HyperStore 7.x and earlier: *proactive.repair.queue.warning.enabled*]

The CMC [Dashboard](#) has a feature whereby it displays a "Long proactive repair queue" warning if the [proactive repair queue](#) for a particular node has more than 10,000 objects in it. This feature requires the Dashboard to retrieve proactive repair queue length data from the Metadata DB, each time the Dashboard page is loaded in your browser. This can sometimes result in slower loading times for the Dashboard.

The *cmc.dashboard.longRepairQueueWarning* setting enables and disables this Dashboard feature. If this setting is set to "false", then the Dashboard does not retrieve proactive repair queue length data from the Metadata DB and does not display any warnings in regard to proactive repair queue length.

Default = false

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.dashboard.longRepairQueueWarning=<boolean>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.dnsName

[Former name in HyperStore 7.x and earlier: *cmc_domain*]

Service endpoint (fully qualified domain name) for the CMC. This endpoint must be resolvable for CMC clients.

Default = Derived from operator input during installation.

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.dnsName=<string>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

Note You also have the option of configuring this setting by using the installer's "Advanced Configuration Options", as described in ["Changing S3, Admin, CMC, or IAM/STS Service Endpoints"](#) (page 859).

cmc.enabled

This setting determines whether the CMC Service is enabled, *true* or *false*. This also determines whether the CMC Service will start automatically when a host reboots.

Default = true

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.enabled=<boolean>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.externalCRR.enabled

[Former name in HyperStore 7.x and earlier: *cmc_crr_external_enabled*]

This controls whether settings for replicating objects from a local source bucket to an external S3 system -- a system other than the HyperStore system in which the source bucket resides -- will appear in the **Cross Region Replication** tab of the CMC's **Bucket Properties** screen. The default is *false*, so that such settings do not appear in the tab, and the tab can only be used to configure replication from a source bucket to a destination bucket in the same HyperStore system.

If you are considering setting *cmc.externalCRR.enabled* to *true* you should first read ["Cross-System Replication"](#) (page 216) including the limitations and caveats noted in that section.

Default = false

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.externalCRR.enabled=<boolean>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.groupList.enabled

[Former name in HyperStore 7.x and earlier: *cmc_grouplist_enabled*]

This setting controls whether the CMC will show a drop-down list of group names when selection of a group is necessary in interior parts of the CMC UI (parts other than the login page). This is relevant only for system administrators, since only system administrators have the opportunity to choose among groups for certain features (such as user management, group management, or usage reporting). Set this to "false" to have the UI

instead present a text box in which a system administrator can type the name of the group they want to work with.

Default = true

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.groupList.enabled=<boolean>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

Note If the number of groups in your system exceeds the value of the "**cmc.groupList.maxSize**" (page 720) setting (100 by default), then group drop-down lists are not supported and the UI will display a text input box for group name regardless of how you've set *cmc.groupList.enabled*.

cmc.groupList.maxSize

[Former name in HyperStore 7.x and earlier: *cmc_grouplist_size_max*]

Maximize number of groups that can be displayed in a CMC drop-down list.

If you have more than this many groups in your HyperStore system, then in parts of the CMC interface that require the user to select a group the interface will display a text input box rather than a drop-down list of groups to select from. The CMC will do this regardless of your settings for "**cmc.login.groupList.enabled**" (page 722) and "**cmc.groupList.enabled**" (page 719).

For example, if *cmc.login.groupList.enabled* and *cmc.groupList.enabled* are both set to *true* and *cmc.groupList.maxSize* is set to 100 (the default values), then the CMC will display drop-down lists for group selection if you have up to 100 groups in your system, or text input boxes for group name entry if you have more than 100 groups in your system.

Default = 100

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.groupList.maxSize=<integer>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.logging.level

This setting controls the level at which CMC Service application logging will occur. Only messages with a severity level that matches or exceeds your configured logging level will be logged.

Listed in order from the least inclusive logging to the most inclusive logging, the supported logging levels are:

- OFF -- The OFF "level" turns off logging.
- FATAL -- The FATAL level designates very severe error events that will likely lead the application to abort.
- ERROR -- The ERROR level designates error events that may still allow the application to continue running.
- WARN -- The WARN level designates potentially harmful situations.

- INFO -- The INFO level designates informational messages that indicate the progress of the application at coarse-grained level.
- DEBUG -- The DEBUG level designates fine-grained informational events that are most useful to debug an application.
- TRACE -- The TRACE level designates finer-grained informational events than the DEBUG level.

For example, if this setting is set to WARN then only messages with severity levels WARN, ERROR, or FATAL will be logged by the CMC Service.

Default = INFO

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set cmc.logging.level=<string>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.login.adminCheckbox.enabled

[Former name in HyperStore 7.x and earlier: `cmc_login_grouplist_admincheckbox_enabled`]

This setting is relevant only if the group name drop-down list is not being displayed in the CMC login page (because "[cmc.login.groupList.enabled](#)" (page 722) is set to *false* or "[cmc.groupList.maxSize](#)" (page 720) is exceeded). With no group name drop-down list in the login page, then:

- If `cmc.login.adminCheckbox.enabled` is set to *true* (the default), then underneath the Group Name text input field an "Admin Group" checkbox displays and system admin users can check this box rather than typing anything in the Group Name field.
- If `cmc.login.adminCheckbox.enabled` is set to *false*, then an "Admin Group" checkbox does not display and instead system admin users must enter the system admin group ID in the Group Name text input field when they log into the CMC. **The system admin group ID is 0.** With this configuration, system admin users must enter **0** in the Group Name field to log in.

Default = true

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set cmc.login.adminCheckbox.enabled=<boolean>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.login.gate.buttonConfirm

[Former name in HyperStore 7.x and earlier: `cmc_login_banner_button_confirm`]

When you are "[Configuring a Login Page Acknowledgment Gate](#)" (page 270), this setting specifies the text of the "confirm" button in the acknowledgment dialog. For example, you could use *Confirm* or *Acknowledge* or *Agree* or *OK* as the button text.

Default = empty

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hstctl config cmc.login.gate.buttonConfirm=<string>
# hstctl config apply cmc
# hstctl service restart cmc --nodes=ALL
```

cmc.login.gate.message

[Former name in HyperStore 7.x and earlier: *cmc_login_banner_message*]

When you are "[Configuring a Login Page Acknowledgment Gate](#)" (page 270), this setting specifies the body text of the acknowledgment dialog. If you want to apply any formatting to this text, use HTML format tags within the text. For example, `<br><br>` to start a second paragraph with a line break between paragraphs; or `<b>text</b>` to create bold text.

Default = empty

To change this setting and apply your change, run the following *hstctl* commands on the Config Controller node:

```
# hstctl config cmc.login.gate.message=<string>
# hstctl config apply cmc
# hstctl service restart cmc --nodes=ALL
```

cmc.login.gate.size

[Former name in HyperStore 7.x and earlier: *cmc_login_banner_size*]

When you are "[Configuring a Login Page Acknowledgment Gate](#)" (page 270), this setting controls the width of the acknowledgment dialog. The valid values are 0, 1, 2, or 3, with 0 being the narrowest and 3 the widest.

Default = 1

To change this setting and apply your change, run the following *hstctl* commands on the Config Controller node:

```
# hstctl config cmc.login.gate.size=<integer>
# hstctl config apply cmc
# hstctl service restart cmc --nodes=ALL
```

cmc.login.gate.title

[Former name in HyperStore 7.x and earlier: *cmc_login_banner_title*]

When you are "[Configuring a Login Page Acknowledgment Gate](#)" (page 270), this setting specifies the title text for the acknowledgment dialog.

Default = empty

To change this setting and apply your change, run the following *hstctl* commands on the Config Controller node:

```
# hstctl config cmc.login.gate.title=<string>
# hstctl config apply cmc
# hstctl service restart cmc --nodes=ALL
```

cmc.login.groupList.enabled

[Former name in HyperStore 7.x and earlier: *cmc_login_grouplist_enabled*]

When users log in to the CMC they must indicate the group to which they belong. This setting controls whether to enable a **Group** selection drop-down list on the CMC's login page:

- If this is set to *true*, in the CMC login page a **Group** drop-down list will show all groups registered in the HyperStore system, and when users log in they can choose their group from the list.
- If this is set to *false*, in the CMC login page a **Group** drop-down list will not display and instead users will need to enter their group name in a Group Name text input box when logging into the CMC. Set this to *false* if you don't want users to see the names of other groups.

Default = true

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set cmc.login.groupList.enabled=<boolean>
# hscctl config apply cmc
# hscctl service restart cmc --nodes=ALL
```

Note If the number of groups in your system exceeds the value of the "[cmc.groupList.maxSize](#)" (page 720) setting (100 by default), then group drop-down lists are not supported and the CMC login page will display a text input box for group name regardless of how you've set *cmc.login.groupList.enabled*.

cmc.login.langSelection.enabled

[Former name in HyperStore 7.x and earlier: *cmc_login_languageselection_enabled*]

This setting controls whether to display at the top of the CMC's login page a selection of languages from which the user can choose, for rendering the CMC's text (such as screen names, button labels, and so on). The supported languages are English, Japanese, Spanish, German, and Portuguese. With this set to *true*, the CMC language will initially be based on the user's browser language setting, but in the login page the user will be able to select a different supported language if they wish.

If you set this to *false*, then the language selection will not display at the top of the CMC's login page, and instead the CMC text language will be exclusively based on the user's browser language setting. If the user's browser language setting matches one of the supported CMC languages, then that language will be used for the CMC text. If the user's browser language setting does not match any of the CMC's supported languages, the CMC text will display in English.

Default = true

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set cmc.login.langSelection.enabled=<boolean>
# hscctl config apply cmc
# hscctl service restart cmc --nodes=ALL
```

cmc.mfa.enforced

[Former name in HyperStore 7.x and earlier: *mfa_enforced*]

This setting controls whether or not Multi-Factor Authentication (MFA) is mandatory for CMC users:

- If `cmc.mfa.enforced` is set to `true`, then when a user tries to log into the CMC, the CMC will prompt them to set up MFA and will present them an interface for doing so. Before using the CMC interface to set up MFA the user must install a virtual MFA device -- a software application that generates MFA codes -- for the browser from which they will be accessing the CMC. This can be any MFA application that supports the open TOTP (time-based one-time password) standard, also known as RFC 6238. Examples of free MFA applications include Google Authenticator and Microsoft Authenticator. Once the user has installed a virtual MFA device for their browser, they can use the CMC's interface for setting up MFA. When the MFA set-up is completed the CMC will automatically log the user out. Then when they log back in they will need to provide an MFA authentication code. Subsequently, whenever they log into the CMC they will need to provide MFA authentication codes. For more information on MFA set-up see ["Enable or Disable Multi-Factor Authentication \(MFA\)" \(page 574\)](#).
 - For already existing CMC users, setting up MFA will be required the next time they try to log into the CMC, after you've set `cmc.mfa.enforced` to `true` and applied the change.
 - For new users created after you've applied the change, setting up MFA will be required the first time they try to log into the CMC.
 - If `cmc.mfa.enforced` is set to `true`, the requirement to use MFA for CMC logins applies to system admin users (including the default system admin user named 'admin') as well as the group admins and regular users.
- If `cmc.mfa.enforced` is set to `false`, then MFA is optional for CMC logins. CMC users can optionally set up MFA for their CMC accounts, as described in ["Enable or Disable Multi-Factor Authentication \(MFA\)" \(page 574\)](#). But users will not be required to set up and use MFA.

Default = false

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set cmc.mfa.enforced=<boolean>
# hscctl config apply cmc
# hscctl service restart cmc --nodes=ALL
```

cmc.pathStyleAccess.enabled

[Former name in HyperStore 7.x and earlier: `path_style_access`]

Whether the CMC (and also the installer's basic validation test script) should use "path style" request formatting when submitting S3 requests to the HyperStore S3 Service. In path style S3 requests, the bucket name is part of the Request-URI rather than being part of the Host header value.

Options are:

- `true` — The CMC will use path style HTTP request formatting when submitting S3 requests to the HyperStore S3 Service. The bucket name associated with the request will be in the Request-URI. For example:

```
PUT /bucket1/objectname HTTP/1.1
Host: s3-region1.mycompany.com
```

- `false` — The CMC will not use path style HTTP request formatting when submitting S3 requests to the HyperStore S3 Service. Instead it will use "virtual host" style access. The bucket name associated with the request will be in the HTTP Host header. For example:

```
PUT /objectname HTTP/1.1
Host: bucket1.s3-region1.mycompany.com
```


Note that this setting affects only the behavior of the CMC and the behavior of the installer's basic validation test script, in their role as S3 clients. Meanwhile the HyperStore S3 Service always supports both path style access and virtual host style access.

IMPORTANT ! If the CMC or any other S3 client applications use virtual host style access to the HyperStore S3 Service, then your DNS environment must be configured to resolve this type of Host value. See ["DNS Set-Up" \(page 53\)](#).

Default = true

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.pathStyleAccess.enabled=<boolean>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.ports.loadBalancer.http

The CMC HTTP port on the load balancer.

Default = 8888

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ports.loadBalancer.http=<integer>
# hsctl config apply all
# hsctl service restart cmc --nodes=ALL
```

cmc.ports.loadBalancer.https

The CMC HTTPS port on the load balancer.

Default = 8443

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ports.loadBalancer.https=<integer>
# hsctl config apply all
# hsctl service restart cmc --nodes=ALL
```

cmc.ports.main.http

[Former name in HyperStore 7.x and earlier: cmc_http_port]

On each node, the port on which the CMC Service listens for regular HTTP requests.

Default = 8888

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ports.main.http=<integer>
# hsctl config apply all
# hsctl service restart cmc firewall --nodes=ALL
```

Note You also have the option of configuring this setting by using the installer's "Advanced Configuration Options", as described in "[Changing S3, Admin, or CMC Listening Ports](#)" (page 858).

cmc.ports.main.https

[Former name in HyperStore 7.x and earlier: *cmc_https_port*]

On each node, the port on which the CMC Service listens for HTTPS requests.

Default = 8443

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ports.main.https=<integer>
# hsctl config apply all
# hsctl service restart cmc firewall --nodes=ALL
```

Note You also have the option of configuring this setting by using the installer's "Advanced Configuration Options", as described in "[Changing S3, Admin, or CMC Listening Ports](#)" (page 858).

cmc.purgeBucket.enabled

[Former name in HyperStore 7.x and earlier: *cmc_purgebucket_enabled*]

For CMC users who want to delete a bucket that has objects in it, this setting controls whether or not users will be given the option to purge all the content in a bucket in a single operation, as a precursor to deleting the bucket:

- If this setting is *true*, and in the CMC's **Bucket** page a user attempts to delete a bucket that has objects in it, the CMC will inform the user that there are objects in the bucket and will present the user an option to purge the entire contents of the bucket by typing 'PURGE' into a confirmation text box. If the user confirms that they want to purge the bucket contents, the system will initiate a purge of the bucket contents as a background operation and inform the user that they can now delete the bucket.

Note The CMC will not allow users to purge contents from a bucket that has Object Lock enabled, even if you have set *cmc.purgeBucket.enabled* to *true*.

- If this setting is *false*, and in the CMC's **Bucket** page a user attempts to delete a bucket that has objects in it, the CMC will inform the user that they must delete the objects in the bucket before they can delete the bucket. The user can then go and delete each object, either through the CMC's **Objects** page, or through a third party S3 application.

Default = false

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.purgeBucket.enabled=<boolean>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.session.timeout

[Former name in HyperStore 7.x and earlier: *cmc_session_timeout*]

Session timeout for a user logged in to the CMC, as a number of minutes. After a logged-in user has been inactive for this many minutes, the CMC will terminate the user's session.

Default = 30

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.session.timeout=<integer>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.sso.enabled

[Former name in HyperStore 7.x and earlier: *cmc_sso_enabled*]

Whether to enable CMC support for single sign-on (SSO) functionality, *true* or *false*. When this is set to *false*, if a user attempts to access the CMC via SSO, the access will be denied and an error will be written to *cloudian-ui.log*.

For more information on CMC SSO, see ["Implementing Single Sign-On for the CMC"](#) (page 271).

Default = false

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.sso.enabled=<boolean>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

IMPORTANT ! If you enable CMC SSO functionality, then for security reasons you should set a custom value for **"cmc.sso.sharedKey"** (page 727). Do not leave this setting at its default value.

cmc.sso.sharedKey

[Former name in HyperStore 7.x and earlier: *cmc_sso_shared_key*]

Shared security key used for hash creation, when using the "auto-login with one way hash" method of single sign-on access to the CMC. (Applicable only if **"cmc.sso.enabled"** (page 727) is set to *true*.)

Default = ss0sh5r3dk3y

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.sso.sharedKey=<secret>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.sso.tolerance

[Former name in HyperStore 7.x and earlier: *sso.tolerance.millis*]

Maximum allowed variance between the CMC server time and the timestamp submitted in a client request invoking the "auto-login with one way hash" method of single sign-on access to the CMC, in milliseconds. If the variance is greater than this, the request is rejected. In effect this serves as a request expiry mechanism.

Default = 3600000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.sso.tolerance=<integer>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.threads.idleTimeout

When the CMC processes HTTP(S) requests from clients, the maximum allowed connection idle time in milliseconds. If this much idle time passes before a new request is received over an open connection with a client, or if this much idle time passes during the reading of headers and content for a request, or if this much idle time passes during the writing of headers and content of a response, the connection with the client is closed.

Default = 20000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.threads.idleTimeout=<integer>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.threads.max

On each node, the maximum number of threads to allow in the thread pool that the CMC uses to process incoming HTTP(S) requests from clients. When there is a new HTTP(S) request processing task and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

Default = 150

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.threads.max=<integer>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.threads.min

On each node, the minimum number of threads to keep in the thread pool that the CMC uses to process incoming HTTP(S) requests from clients.

The CMC uses a thread pool to process incoming HTTP(S) requests from clients. An initial pool of threads is created at server initialization time, and additional threads may be added as needed (within the limit set by "[cmc.threads.max](#)" (page 728)). Idle threads in the pool are terminated if not used within a thread timeout

period — unless the number of threads in the pool is down to *cmc.threads.min*. If only this many threads are in the pool, then threads are kept open even if they've been idle for longer than the thread timeout.

Default = 25

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.threads.min=<integer>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.tiering.allowCustomEndpoint

[Former name in HyperStore 7.x and earlier: *tiering_customendpoint_enabled*]

Applicable only if "**cmc.tiering.enabled**" (page 731) and "**cmc.tiering.allowPerBucketCredentials**" (page 729) are both set to *true* (by default *cmc.tiering.enabled* is *false* and *cmc.tiering.allowPerBucketCredentials* is *true*)

- If you set *cmc.tiering.allowCustomEndpoint* to *true*, then users configuring bucket lifecycle auto-tiering rules through the CMC will be able to choose from among the several tiering destinations described for "**cmc.tiering.allowPerBucketCredentials**" (page 729) and also they will have the option to enter an endpoint URL for an S3-compliant destination of their own choosing. If they do so they will also enter their own security credentials for that destination, in a separate field that will display in the interface. HyperStore will attempt to connect to the specified custom domain using the user's supplied credentials, and if the connection is successful the user will be able to auto-tier to that domain.
- If *cmc.tiering.allowCustomEndpoint* is set to *false*, then users configuring bucket lifecycle auto-tiering rules through the CMC will not be able to specify a custom tiering destination endpoint. Instead they will choose among the several system-configured destinations (the default list is in the description of "**cmc.tiering.allowPerBucketCredentials**" (page 729)).

Default = false

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.tiering.allowCustomEndpoint=<boolean>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

Note You also have the option of configuring this setting in the **Auto-Tiering** section of the CMC's "**Configuration Settings**" (page 498) page. In the CMC UI this setting is called "Enable Custom Endpoint".

cmc.tiering.allowPerBucketCredentials

[Former name in HyperStore 7.x and earlier: *tiering_perbucketcredentials_enabled*]

Applicable only if "**cmc.tiering.enabled**" (page 731) is set to *true* (by default it's set to *false*).

- If *cmc.tiering.allowPerBucketCredentials* is set to *true*, then users configuring bucket lifecycle auto-tiering rules through the CMC will be able to choose from several different options for tiering destination, and they will provide (through the CMC interface) their own security credentials for their selected tiering destination. By default the tiering destinations that will be available to users are:

- AWS S3 (default endpoint = <https://s3.amazonaws.com>)
- AWS Glacier (default endpoint = <https://s3.amazonaws.com>)
- Google Cloud Storage (default endpoint = <https://storage.googleapis.com>)
- Azure (default endpoint = <https://blob.core.windows.net>)
- Spectra BlackPearl (default endpoint = <https://bplab.spectrallogic.com>) (note that Spectra BlackPearl is in the default list of tiering destinations only if your original installed HyperStore version was older than 7.1.4).

Note You can change this list of destinations and/or their endpoints if you wish. For details see "[Configure Tiering Destinations](#)" (page 210).

- If `cmc.tiering.allowPerBucketCredentials` is set to `false`, then all users configuring bucket lifecycle auto-tiering rules through the CMC will be presented with just one, system-default tiering destination, and all tiering to that destination will use a default set of security credentials that you supply to the system. You set the one default tiering destination with the "[cmc.tiering.defaultEndpointURL](#)" (page 730) setting.

Default = true

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set cmc.tiering.allowPerBucketCredentials=<boolean>
# hscctl config apply cmc
# hscctl service restart cmc --nodes=ALL
```

Note You also have the option of configuring this setting in the **Auto-Tiering** section of the CMC's "[Configuration Settings](#)" (page 498) page. In the CMC UI this setting is called "Enable Per Bucket Credentials".

cmc.tiering.defaultEndpointURL

[Former name in HyperStore 7.x and earlier: `bucket_transition_url`]

Applicable only if "[cmc.tiering.enabled](#)" (page 731) is set to `true` (by default it's set to `false`).

If you want all users setting bucket lifecycle auto-tiering rules through the CMC to tier to the same tiering destination, using the same account credentials, set "[cmc.tiering.allowPerBucketCredentials](#)" (page 729) to `false` (it's `true` by default) and use the `cmc.tiering.defaultEndpointURL` setting to specify the tiering destination URL. For example, if you want all users to tier to the same corporate account with Amazon S3 you could set the `cmc.tiering.defaultEndpointURL` to <https://s3.amazonaws.com> -- or if your organization is based in California, USA, <https://s3.us-west-1.amazonaws.com>. Be sure to include the `https://` part when configuring your default tiering URL.

If you have a multi-region HyperStore system, note that the `cmc.tiering.defaultEndpointURL` setting will apply to all of your service regions.

Default = <https://s3.amazonaws.com>

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set cmc.tiering.defaultEndpointURL=<string>
# hscctl config apply cmc
# hscctl service restart cmc --nodes=ALL
```

IMPORTANT ! If you want CMC users to use a default tiering URL ("[cmc.tiering.allowPerBucketCredentials](#)" (page 729) is set to *false* and *cmc.tiering.defaultEndpointURL* is set to your desired default tiering URL), then you **must** set default security credentials to be used with the default tiering URL. You can set these default security credentials (access key and secret key), either through the CMC's **Configuration Settings** page (in the "Auto-Tiering" section) or through the Admin API call *POST /tiering/credentials*. Note that the CMC's **Configuration Settings** page "Auto-Tiering" section also lets you configure the default tiering URL itself, if you'd prefer to do it there rather than on the command line with *hscctl config set*.

cmc.tiering.enabled

[Former name in HyperStore 7.x and earlier: *bucket_transition_enabled*]

This setting controls whether or not the CMC will support configuring auto-tiering rules for buckets:

- If you set *cmc.tiering.enabled* to *true*, then the CMC interface for configuring a bucket lifecycle (**Buckets & Objects** -> **Buckets** -> **bucket Properties**) will include an option for configuring an auto-tiering rule.
- If *cmc.tiering.enabled* is set to *false*, then the bucket lifecycle configuration interface will not display an option for configuring an auto-tiering rule.

Note that this setting only controls whether bucket lifecycle rules for auto-tiering can be configured **through the CMC**. Even if you have this setting set to *false*, bucket lifecycle rules for auto-tiering can still be configured through HyperStore extensions to the S3 API method *PutBucketLifecycle*, by other S3 client applications that call that API method.

For general information about auto-tiering see "[Auto-Tiering Feature Overview](#)" (page 203).

Default = false

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set cmc.tiering.enabled=<boolean>
# hscctl config apply cmc
# hscctl service restart cmc --nodes=ALL
```

Note You also have the option of configuring this setting in the **Auto-Tiering** section of the CMC's "[Configuration Settings](#)" (page 498) page. In the CMC UI this setting is called "Enable Auto-Tiering".

cmc.tls.required

[Former name in HyperStore 7.x and earlier: *cmc_web_secure*]

Whether the CMC will require connecting clients to use TLS (HTTPS), *true* or *false*.

Default = true

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set cmc.tls.required=<boolean>
# hscctl config apply all
# hscctl service restart cmc --nodes=ALL
```

cmc.ui.account.credentials.accessKeyGrantees

[Former name in HyperStore 7.x and earlier: *account.credentials.access.enabled*]

This setting specifies the CMC user role(s) that can view and change their own S3 storage access keys in the CMC's **Security Credentials** interface. Configure as a comma-separated list if specifying more than one role. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin, GroupAdmin, and/or User (regular non-administrative users); or to hide this functionality from all user roles set this setting to [] (brackets with no space between, this denotes an empty list of user roles).

Default = SystemAdmin,GroupAdmin,User

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set cmc.ui.account.credentials.accessKeyGrantees=<list>
# hscctl config apply cmc
# hscctl service restart cmc --nodes=ALL
```

cmc.ui.account.credentials.grantees

[Former name in HyperStore 7.x and earlier: *account.credentials.enabled*]

This setting specifies the CMC user role(s) that can access the CMC's **Security Credentials** interface (through a drop-down list under the user's login name in the upper right of the GUI). Configure as a comma-separated list if specifying more than one role. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin, GroupAdmin, and/or User (regular non-administrative users); or to hide this functionality from all user roles set this setting to [] (brackets with no space between, this denotes an empty list of user roles).

Default = SystemAdmin,GroupAdmin,User

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set cmc.ui.account.credentials.grantees=<list>
# hscctl config apply cmc
# hscctl service restart cmc --nodes=ALL
```

Note If you want to enable some aspects of the **Security Credentials** interface and not others, you can have *cmc.ui.account.credentials.grantees* set so that the interface is enabled for your desired user types, and then use the granular *cmc.ui.account.credentials.<*>Grantees* settings to enable/disable specific capabilities within the interface.

cmc.ui.account.credentials.signinGrantees

[Former name in HyperStore 7.x and earlier: *account.credentials.signin.enabled*]

This setting specifies the CMC user role(s) that can view and change their own CMC login password in the CMC's **Security Credentials** interface. Configure as a comma-separated list if specifying more than one role. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin, GroupAdmin, and/or User (regular non-administrative users); or to hide this functionality from all user roles set this setting to [] (brackets with no space between, this denotes an empty list of user roles).

Default = SystemAdmin,GroupAdmin,User

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.account.credentials.signinGrantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.ui.account.profile.editGrantees

[Former name in HyperStore 7.x and earlier: account.profile.writeable.enabled]

This setting specifies the CMC user role(s) that can view and change their own account profile information in the CMC's **Account Profile** interface (accessible through a drop-down list under the user's login name in the upper right of the GUI). Configure as a comma-separated list if specifying more than one role. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin, GroupAdmin, and/or User (regular non-administrative users); or to hide this functionality from all user roles set this setting to [] (brackets with no space between, this denotes an empty list of user roles).

Default = SystemAdmin,GroupAdmin,User

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.account.profile.editGrantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.ui.account.view.activityGrantees

[Former name in HyperStore 7.x and earlier: account.activity.enabled]

This setting specifies the CMC user role that can access the CMC's **Account Activity** interface (**Users & Groups -> Account Activity**). For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin; or to hide this functionality from all user roles set this setting to [] (brackets with no space between, this denotes an empty list of user roles).

Default = SystemAdmin

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.account.view.activityGrantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.ui.account.view.usageGrantees

[Former name in HyperStore 7.x and earlier: usage.enabled]

This setting specifies the CMC user role(s) that can access the CMC's **Usage By Users and Groups** interface (**Analytics -> Usage By User & Group**). Configure as a comma-separated list if specifying more than one role. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin, GroupAdmin, and/or User (regular non-administrative users); or to hide this functionality from all user roles set this setting to `[ ]` (brackets with no space between, this denotes an empty list of user roles).

Default = SystemAdmin,GroupAdmin,User

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set cmc.ui.account.view.usageGrantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.ui.admin.manageGroups.createGrantees

[Former name in HyperStore 7.x and earlier: admin.manage_groups.create.enabled]

This setting specifies the CMC user role that can create new groups in the CMC's **Manage Groups** interface. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin; or to hide this functionality from all user roles set this setting to `[ ]` (brackets with no space between, this denotes an empty list of user roles).

Default = SystemAdmin

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set cmc.ui.admin.manageGroups.createGrantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.ui.admin.manageGroups.deleteGrantees

[Former name in HyperStore 7.x and earlier: admin.manage_groups.delete.enabled]

This setting specifies the CMC user role that can delete groups in the CMC's **Manage Groups** interface. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin; or to hide this functionality from all user roles set this setting to `[ ]` (brackets with no space between, this denotes an empty list of user roles).

Default = SystemAdmin

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set cmc.ui.admin.manageGroups.deleteGrantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.ui.admin.manageGroups.edit.defaultQOSGrantees

[Former name in HyperStore 7.x and earlier: *admin.manage_groups.user_qos_groups_default.enabled*]

This setting specifies the CMC user role(s) that can set default Quality of Service (QoS) controls for users within a specific group in the CMC's **Manage Groups** interface. Configure as a comma-separated list if specifying more than one role. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin and/or GroupAdmin; or to hide this functionality from all user roles set this setting to `[ ]` (brackets with no space between, this denotes an empty list of user roles). If enabled for group admins, the capability is confined to their own group.

Default = SystemAdmin,GroupAdmin

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.admin.manageGroups.edit.defaultQOSGrantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

Note To have access to this functionality a user role must be specified by this setting and also by the parent settings "[cmc.ui.admin.manageGroups.grantees](#)" (page 736) and "[cmc.ui.admin.manageGroups.edit.grantees](#)" (page 735).

cmc.ui.admin.manageGroups.edit.grantees

[Former name in HyperStore 7.x and earlier: *admin.manage_groups.edit.enabled*]

This setting specifies the CMC user role(s) that can edit an existing group's profile and service attributes in the CMC's **Manage Groups** interface. Configure as a comma-separated list if specifying more than one role. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin and/or GroupAdmin; or to hide this functionality from all user roles set this setting to `[ ]` (brackets with no space between, this denotes an empty list of user roles). If enabled for group admins, the capability is confined to their own group.

Default = SystemAdmin,GroupAdmin

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.admin.manageGroups.edit.grantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

Note To have access to this functionality a user role must be specified by this setting and also by the parent setting "[cmc.ui.admin.manageGroups.grantees](#)" (page 736).

Note Even when this editing capability is enabled for group admins, they will not be able to perform certain group-related actions that are reserved for system admins, such as setting QoS controls for the group as a whole or assigning a default rating plan for the group. Group admins' privileges will be limited to changing their group description and (possibly) changing the default user QoS settings for the

group. The latter capability is controlled by the more granular configuration setting "[cmc.ui.admin.manageGroups.edit.defaultQOSGrantees](#)" (page 735).

cmc.ui.admin.manageGroups.grantees

[Former name in HyperStore 7.x and earlier: *admin.manage_groups.enabled*]

This setting specifies the CMC user role(s) that can access the CMC's **Manage Groups** interface (**Users & Groups -> Manage Groups**). Configure as a comma-separated list if specifying more than one role. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin and/or GroupAdmin; or to hide this functionality from all user roles set this setting to [] (brackets with no space between, this denotes an empty list of user roles). If enabled for group admins, the capability is confined to their own group.

Default = SystemAdmin,GroupAdmin

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.admin.manageGroups.grantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

Note If you want to enable some aspects of the **Manage Groups** interface and not others, you can have *cmc.ui.admin.manageGroups.grantees* set so that the function is enabled for your desired user roles, and then use the granular *cmc.ui.admin.manageGroups.<*>Grantees* settings to enable/disable specific capabilities.

cmc.ui.admin.manageUsers.createGrantees

This setting specifies the CMC user role(s) that can create new users in the CMC's **Manage Users** interface. Configure as a comma-separated list if specifying more than one role. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin and/or GroupAdmin; or to hide this functionality from all user roles set this setting to [] (brackets with no space between, this denotes an empty list of user roles). If enabled for group admins, the capability is confined to their own group.

Default = SystemAdmin,GroupAdmin

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.admin.manageUsers.createGrantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

Note To have access to this functionality a user role must be specified by this setting and also by the parent setting "[cmc.ui.admin.manageUsers.grantees](#)" (page 739).

cmc.ui.admin.manageUsers.deleteGrantees

[Former name in HyperStore 7.x and earlier: *admin.manage_users.delete.enabled*]

This setting specifies the CMC user role(s) that can delete users in the CMC's **Manage Users** interface. Configure as a comma-separated list if specifying more than one role. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin and/or GroupAdmin; or to hide this functionality from all user roles set this setting to `[ ]` (brackets with no space between, this denotes an empty list of user roles). If enabled for group admins, the capability is confined to their own group.

Default = SystemAdmin,GroupAdmin

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.admin.manageUsers.deleteGrantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

Note To have access to this functionality a user role must be specified by this setting and also by the parent setting "[cmc.ui.admin.manageUsers.grantees](#)" (page 739).

cmc.ui.admin.manageUsers.edit.credentialsGrantees

[Former name in HyperStore 7.x and earlier: *admin.manage_users.edit.user_credentials.enabled*]

This setting specifies the CMC user role(s) that can change users' CMC login passwords and to view and manage user's S3 access credentials in the CMC's **Manage Users** interface. Configure as a comma-separated list if specifying more than one role. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin and/or GroupAdmin; or to hide this functionality from all user roles set this setting to `[ ]` (brackets with no space between, this denotes an empty list of user roles). If enabled for group admins, the capability is confined to their own group.

Default = SystemAdmin,GroupAdmin

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.admin.manageUsers.edit.credentialsGrantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

Note To have access to this functionality a user role must be specified by this setting and also by the parent settings "[cmc.ui.admin.manageUsers.grantees](#)" (page 739) and "[cmc.ui.admin.manageUsers.edit.grantees](#)" (page 738).

Note Regardless of how you configure the *cmc.ui.admin.manageUsers.edit.credentialsGrantees* setting, regular users can manage their own CMC login password and S3 access credentials, in the **Security Credentials** page of the CMC.

cmc.ui.admin.manageUsers.edit.grantees

[Former name in HyperStore 7.x and earlier: *admin.manage_users.edit.enabled*]

This setting specifies the CMC user role(s) that can edit existing users' profiles and service attributes in the CMC's **Manage Users** interface. Configure as a comma-separated list if specifying more than one role. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin and/or GroupAdmin; or to hide this functionality from all user roles set this setting to `[ ]` (brackets with no space between, this denotes an empty list of user roles). If enabled for group admins, the capability is confined to their own group.

Default = SystemAdmin,GroupAdmin

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.admin.manageUsers.edit.grantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

Note To have access to this functionality a user role must be specified by this setting and also by the parent setting "[cmc.ui.admin.manageUsers.grantees](#)" (page 739).

Note Regardless of how you configure the *cmc.ui.admin.manageUsers.edit.grantees* setting:

- * Nobody but a system administrator can ever change a user's rating plan assignment.
- * Regular users can edit their own profile information, in the **Profile** page of the CMC.

cmc.ui.admin.manageUsers.edit.qosGrantees

[Former name in HyperStore 7.x and earlier: *admin.manage_users.edit.user_qos.enabled*]

This setting specifies the CMC user role(s) that can set Quality of Service (QoS) controls for specific users in the CMC's **Manage Users** interface. Configure as a comma-separated list if specifying more than one role. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin and/or GroupAdmin; or to hide this functionality from all user roles set this setting to `[ ]` (brackets with no space between, this denotes an empty list of user roles). If enabled for group admins, the capability is confined to their own group.

Default = SystemAdmin,GroupAdmin

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.admin.manageUsers.edit.qosGrantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

Note To have access to this functionality a user role must be specified by this setting and also by the parent settings "[cmc.ui.admin.manageUsers.grantees](#)" (page 739) and "[cmc.ui.admin.manageUsers.edit.grantees](#)" (page 738).

cmc.ui.admin.manageUsers.grantees

[Former name in HyperStore 7.x and earlier: *admin.manage_users.enabled*]

This setting specifies the CMC user role(s) that can access the CMC's **Manage Users** interface (**Users & Groups -> Manage Users**). Configure as a comma-separated list if specifying more than one role. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin and/or GroupAdmin; or to hide this functionality from all user roles set this setting to `[ ]` (brackets with no space between, this denotes an empty list of user roles). If enabled for group admins, the capability is confined to their own group.

Default = SystemAdmin,GroupAdmin

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.admin.manageUsers.grantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

Note If you want to enable some aspects of the **Manage Users** interface and not others, you can have *cmc.ui.admin.manageUsers.grantees* set so that the function is enabled for your desired user roles, and then use the granular *cmc.ui.admin.manageUsers.<*>Grantees* settings to enable/disable specific capabilities.

cmc.ui.admin.manageUsers.view.objectDataGrantees

[Former name in HyperStore 7.x and earlier: *cmc_view_user_data*]

This setting specifies the CMC user role(s) that can access regular users' storage buckets in the CMC's **Manage Users** interface. Configure as a comma-separated list if specifying more than one role. For user roles not specified by this setting, this CMC functionality will be hidden.

When allowed this access, admin users can view regular users' data, add data to users' buckets, and delete users' data. Also when allowed this access, admin users can change the properties of regular users' buckets and objects.

Options are SystemAdmin and/or GroupAdmin; or to hide this functionality from all user roles set this setting to `[ ]` (brackets with no space between, this denotes an empty list of user roles). If enabled for group admins, the capability is confined to their own group.

Default = empty (so no user roles have this access)

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.admin.manageUsers.view.objectDataGrantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

Note The *cmc.ui.admin.manageUsers.view.objectDataGrantees* setting does not apply to file shares owned by users for whom you have enabled [File Services](#). Regardless of how the *cmc.ui.admin.manageUsers.view.objectDataGrantees* setting is configured, administrators cannot add, view, edit, or delete users' file shares.

cmc.ui.bucket.view.storagePolicyGrantees

[Former name in HyperStore 7.x and earlier: bucket.storagepolicy.showdetail.enabled]

This setting specifies the CMC user role(s) that can access the "Storage Policy" tab in the CMC's **Bucket Properties** interface. The "Storage Policy" tab provides information about the storage policy being used by the bucket, including the storage policy's replication factor or erasure coding k+m configuration.

Configure as a comma-separated list if specifying more than one role. For user roles not specified by this setting, this CMC functionality will be hidden.

Options are SystemAdmin, GroupAdmin, and/or User (regular non-administrative users); or to hide this functionality from all user roles set this setting to [] (brackets with no space between, this denotes an empty list of user roles).

Default = SystemAdmin,GroupAdmin,User

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.bucket.view.storagePolicyGrantees=<list>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.ui.page.defaultSize

[Former name in HyperStore 7.x and earlier: page.size.default]

In the CMC's **Usage By Users & Groups** page, the default number of table rows to display on each page of a tabular report.

Default = 10

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.page.defaultSize=<integer>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.ui.page.maxSize

[Former name in HyperStore 7.x and earlier: page.size.max]

In the CMC's **Usage By Users & Groups** page, the maximum number of table rows that users can select to display on each page of a tabular report.

Default = 100

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.page.maxSize=<integer>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

cmc.ui.security.serviceInfo

[Former name in HyperStore 7.x and earlier: security.serviceinfo.enabled]

This setting controls whether HyperStore's S3 service endpoints will display for group admins and regular users in the CMC's **Security Credentials** page (accessible through a drop-down list under the user's login name in the upper right of the GUI).

Options are:

- *true* — S3 service endpoints will display for group admins and regular users in the CMC's **Security Credentials** page.
- *false* — S3 service endpoints will not display for group admins or regular users in the CMC's **Security Credentials** page.

Default = true

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set cmc.ui.security.serviceInfo=<boolean>
# hsctl config apply cmc
# hsctl service restart cmc --nodes=ALL
```

Note For HyperStore system admins, the S3 service endpoint information displays in the [Cluster Information](#) page. The *cmc.ui.security.serviceInfo* setting has no effect on this display.

13.2.4. Common Settings

The table below lists configuration settings in the *common* setting group, which consists of settings that are used by multiple services. When applicable the settings in the table are organized into sub-groups.

Setting Sub-Group	Setting	Default Value
common.appliance	common.appliance.blinkInterval.chassis (page 743)	180
	common.appliance.blinkInterval.disk (page 743)	180
common.phoneHome	common.phoneHome.auth.accessKey (page 744)	empty
	common.phoneHome.auth.secretKey (page 744)	empty
	common.phoneHome.bucket (page 744)	empty
	common.phoneHome.concealPrivateData.enabled (page 745)	false
	common.phoneHome.concealPrivateData.bucketObjectNames (page 745)	false
	common.phoneHome.enabled (page 745)	true
	common.phoneHome.logging.level (page 746)	WARN
	common.phoneHome.logging.* rollover settings	see settings description for default values
	common.phoneHome.proxy.auth.password (page 747)	empty
	common.phoneHome.proxy.auth.username (page 747)	empty

Setting Sub-Group	Setting	Default Value
	common.phoneHome.proxy.enabled (page 747)	true
	common.phoneHome.proxy.host (page 748)	empty
	common.phoneHome.proxy.port (page 748)	3128
	common.phoneHome.url (page 749)	https://s3-sup-port.cloudian.com:443
	common.phoneHome.verifyCA	true
common.services	common.services.autoRestart (page 749)	onFailure
	common.services.enabled (page 750)	true
	common.services.heap.dumpOnOOM (page 750)	true
common.sysinfo	common.sysInfo.auth.accessKey (page 751)	empty
	common.sysInfo.auth.secretKey (page 751)	empty
	common.sysInfo.bucket (page 751)	empty
	common.sysInfo.proxy.auth.password (page 752)	empty
	common.sysInfo.proxy.auth.username (page 752)	empty
	common.sysInfo.proxy.enabled (page 752)	true
	common.sysInfo.proxy.host (page 753)	empty
	common.sysInfo.proxy.port (page 753)	empty
	common.sysInfo.url (page 753)	empty
common.user	common.user.allowDeleteWithBuckets (page 754)	false
	common.user.allowDeleteWithIAMResources (page 754)	false
	common.user.name.maxLength (page 755)	64
	common.user.password.dupCharRatioLimit (page 755)	-1
	common.user.password.lock.durationSec (page 756)	1800
	common.user.password.lock.enabled (page 756)	true
	common.user.password.lock.maxFailedAttempts (page 757)	6
	common.user.password.maxLength (page 757)	64
	common.user.password.minLength (page 757)	9
	common.user.password.rotation.expiration (page 758)	0
	common.user.password.rotation.gracePeriod (page 758)	0
	common.user.password.uniqueGenerations (page 758)	0
common.xForwardedHeaders	common.xForwardedHeaders.enabled (page 759)	false
	common.xForwardedHeaders.trusted (page 760)	false
n/a (miscellaneous)	common.configController.failover	manual

Setting Sub-Group	Setting	Default Value
individual settings)	common.fips.enabled (page 743)	false
	common.licenseFileName (page 744)	cloudian_license.lic
	common.tls.ciphers.profile	default

13.2.4.1. common.* Setting Descriptions

common.appliance.blinkInterval.chassis

[Former name in HyperStore 7.x and earlier: *blink_appliance_intervalsec*]

Applicable only if you have HyperStore Appliances.

If in the Appliance section of the CMC's **Node Status** page you click the **Blink Chassis** function for one of your appliances, the light on the appliance chassis will blink for this many seconds.

Default = 180 (three minutes)

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set common.appliance.blinkInterval.chassis=<integer>
# hsctl config apply common
# hsctl service restart cmc --nodes=ALL
```

common.appliance.blinkInterval.disk

[Former name in HyperStore 7.x and earlier: *blink_disk_intervalsec*]

Applicable only if you have HyperStore Appliances.

If in the Appliance section of the CMC's **Node Status** page you click the **Blink** function for one of the disk drives on one of your appliances, the light on the drive will blink for this many seconds.

Default = 180 (three minutes)

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set common.appliance.blinkInterval.disk=<integer>
# hsctl config apply common
# hsctl service restart cmc --nodes=ALL
```

common.fips.enabled

[Former name in HyperStore 7.x and earlier: *fips_enabled*]

For information about this setting see ["FIPS Support" \(page 203\)](#).

Default = false

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set common.fips.enabled=<boolean>
# hsctl config apply all
# hsctl service restart all --nodes=ALL
```

common.licenseFileName

[Former name in HyperStore 7.x and earlier: cloudian_license_file_name]

Name of your Clouidian license file.

Default = Derived from operator input during installation.

Do not edit manually. To apply an updated license file, use the CMC's Update License function (**Cluster** → **Cluster Config** → **Cluster Information** → **Update License**). That function will automatically update this configuration setting as appropriate and dynamically apply the change to your live system. No service restart is necessary.

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set common.licenseFileName=<string>
# hsctl config apply common
# hsctl service restart s3 cmc --nodes=ALL
```

Note You also have the option of configuring this setting by using the CMC's Update License function (**Cluster** → **Cluster Config** → **Cluster Information** → **Update License**). That CMC function will automatically update this configuration setting as appropriate and dynamically apply the change to your live system, without requiring any service restart.

common.phoneHome.auth.accessKey

[Former name in HyperStore 7.x and earlier: phonehome_access_key]

For description of this setting see "[common.phoneHome.bucket](#)" (page 744).

Default = empty

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set common.phoneHome.auth.accessKey=<string>
# hsctl config apply common
```

No service restart is required.

common.phoneHome.auth.secretKey

[Former name in HyperStore 7.x and earlier: phonehome_secret_key]

For description of this setting see "[common.phoneHome.bucket](#)" (page 744).

Default = empty

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set common.phoneHome.auth.secretKey=<secret>
# hsctl config apply common
```

No service restart is required.

common.phoneHome.bucket

[Former name in HyperStore 7.x and earlier: phonehome_bucket]

- If you leave "**common.phoneHome.url**" (page 749) at its default value -- which is the Clodian Support S3 URL -- you can leave the `common.phoneHome.bucket`, "**common.phoneHome.auth.accessKey**" (page 744), and "**common.phoneHome.auth.secretKey**" (page 744) settings empty (as they are by default). The Smart Support feature will automatically extract the Clodian Support S3 bucket name and security credentials from your encrypted HyperStore license file.
- If you set "**common.phoneHome.url**" (page 749) to an S3 URL other than the Clodian Support URI, set the `common.phoneHome.bucket`, "**common.phoneHome.auth.accessKey**" (page 744), and "**common.phoneHome.auth.secretKey**" (page 744) settings to the destination bucket name and the S3 access key and secret key for accessing that bucket.

Default = empty

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set common.phoneHome.bucket=<string>
# hsctl config apply common
```

No service restart is required.

common.phoneHome.concealPrivateData.bucketObjectNames

[Former name in HyperStore 7.x and earlier: `phonehome_gdpr_bucket`]

For description of how to use this setting, see "**Maintaining Personal Data Privacy While Using Smart Support and Node Diagnostics**" (page 121).

Default = false

common.phoneHome.concealPrivateData.enabled

[Former name in HyperStore 7.x and earlier: `phonehome_gdpr`]

For description of how to use this setting, see "**Maintaining Personal Data Privacy While Using Smart Support and Node Diagnostics**" (page 121).

Default = false

common.phoneHome.enabled

[Former name in HyperStore 7.x and earlier: `phonehome.enabled`]

The HyperStore Data Collector collects and stores system-wide diagnostic data for your HyperStore system on an ongoing basis. By default this diagnostics data is automatically uploaded to Clodian Support via the S3 protocol once a day, as part of the Smart Support feature (also known as the "Phone Home" feature). For more information about this feature -- including what data gets sent to Clodian Support and how they use it for your benefit -- see "**Smart Support and Diagnostics Feature Overview**" (page 117).

- **If you want diagnostics data to be automatically uploaded each day to Clodian Support via S3**, there's nothing you need to do -- just leave `common.phoneHome.enabled` set to `true` and leave the setting "**common.phoneHome.url**" (page 749) set to the Clodian Support S3 URL. This is the default behavior and is the recommended behavior.
- **If you want diagnostics data to be automatically uploaded each day to an S3 destination other than Clodian Support**, leave `phonehome.enabled` set to `true` and set the "**common.phoneHome.url**" (page 749) setting to the desired S3 URL and also set the "**common.phoneHome.bucket**" (page 744),

"[common.phoneHome.auth.accessKey](#)" (page 744), and "[common.phoneHome.auth.secretKey](#)" (page 744) settings to the desired destination bucket and access credentials.

- If you do not want diagnostics data automatically uploaded to an S3 destination each day, set `common.phoneHome.enabled` to `false`. This is not recommended.

Even if you choose not to automatically upload the daily diagnostic data to an S3 destination -- that is, even if you set `common.phoneHome.enabled` to `false` -- a diagnostics data file is still generated locally and stored under `/var/log/cloudian` on the node on which the HyperStore Monitoring Data Collector runs. (To see which of your nodes is the Data Collector node, go to the CMC's [Cluster Information](#) page and check for the identity of the "System Monitoring/Cronjob Primary Host".) The "live" diagnostics log -- which is recording the current day's performance statistics -- is named `diagnostics.csv`. The rolled up daily diagnostic packages from previous days -- which include prior days' `diagnostics.csv` files and also various application and transaction logs -- are named `diagnostics_<date/time>_<version>_<region>.tgz`. Diagnostics packages are automatically deleted 15 days after they are created.

Default = true

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set common.phoneHome.enabled=<boolean>
# hscctl config apply common
```

No service restart is required.

Note HyperStore also supports a set of `hyperIQ.phoneHome.*` settings. These settings are for **future** use, when the [Cloudian HyperIQ application](#) will support the ability to parse and analyze phone home files. At that point in time you will be able to use the `hyperIQ.phoneHome.*` settings to specify a local, on-premise bucket to which to write daily phone home files for HyperIQ consumption **in addition to** having the system upload daily phone home files to Cloudian Support. Currently HyperIQ does not yet support the ability to process phone home files and so in HyperStore 8.2 the `hyperIQ.phoneHome.*` settings are empty and do not yet have any utility. When HyperIQ does start supporting this ability, this will be reflected in your HyperIQ documentation and in your HyperStore documentation.

common.phoneHome.logging.level

This setting controls the level at which Phone Home (Smart Support) application logging will occur. Only messages with a severity level that matches or exceeds your configured logging level will be logged.

Listed in order from the least inclusive logging to the most inclusive logging, the supported logging levels are:

- OFF -- The OFF "level" turns off logging.
- FATAL -- The FATAL level designates very severe error events that will likely lead the application to abort.
- ERROR -- The ERROR level designates error events that may still allow the application to continue running.
- WARN -- The WARN level designates potentially harmful situations.
- INFO -- The INFO level designates informational messages that indicate the progress of the application at coarse-grained level.
- DEBUG -- The DEBUG level designates fine-grained informational events that are most useful to debug

an application.

- **TRACE** -- The TRACE level designates finer-grained informational events than the DEBUG level.

For example, if this setting is set to WARN then only messages with severity levels WARN, ERROR, or FATAL will be logged by the Phone Home service.

Default = WARN

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set common.phoneHome.logging.level=<string>
# hscctl config apply common
```

No service restart is required.

common.phoneHome.proxy.auth.password

[Former name in HyperStore 7.x and earlier: phonehome_proxy_password]

If you want HyperStore to use a local forward proxy when the Smart Support feature sends daily system diagnostics packages to Clouddian Support, use this setting to specify the password that HyperStore should use when connecting to the proxy (if a username and password are required by the proxy).

Default = empty

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set common.phoneHome.proxy.auth.password=<secret>
# hscctl config apply common
```

No service restart is required.

common.phoneHome.proxy.auth.username

[Former name in HyperStore 7.x and earlier: phonehome_proxy_username]

If you want HyperStore to use a local forward proxy when the Smart Support feature sends daily system diagnostics packages to Clouddian Support, use this setting to specify the username that HyperStore should use when connecting to the proxy (if a username and password are required by the proxy).

Default = empty

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set common.phoneHome.proxy.auth.username=<string>
# hscctl config apply common
```

No service restart is required.

common.phoneHome.proxy.enabled

For information on this functionality see ["common.phoneHome.proxy.host" \(page 748\)](#).

Although *common.phoneHome.proxy.enabled* defaults to *true*, this functionality is not actually enabled unless you set a value for *common.phoneHome.proxy.host*.

If you configure *common.phoneHome.proxy.host* and potentially the other settings for this proxy-using functionality (such as the proxy authentication settings), and then subsequently you want to stop using the proxy

without having to delete this configuration information that you've set up, you can set `common.phoneHome.proxy.enabled` to `false`.

Default = true

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set common.phoneHome.proxy.enabled=<boolean>
# hsctl config apply common
```

No service restart is required.

common.phoneHome.proxy.host

[Former name in HyperStore 7.x and earlier: `phonehome_proxy_host`]

If you want HyperStore to use a local forward proxy when the Smart Support feature sends daily system diagnostics packages to Cloudian Support, use this setting to specify the hostname or IP address of the proxy.

Default = empty

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set common.phoneHome.proxy.host=<string>
# hsctl config apply common
```

No service restart is required.

Note For this feature you should configure your forward proxy to support access to `*.s3-support.cloudian.com` (that is, to any sub-domain of `s3-support.cloudian.com`).

Note By default any proxy settings that you configure for the daily Smart Support uploads will also apply to the sending of on-demand Node Diagnostics packages (triggered by your using the CMC's [Collect Diagnostics](#) function). If you want to use a different proxy for Node Diagnostics sending than you do for the daily Smart Support upload, use the [common.sysinfo.proxy.*](#) settings to separately configure proxy information for Node Diagnostics sending.

For more background information on these features see "[Smart Support and Diagnostics Feature Overview](#)" (page 117).

common.phoneHome.proxy.port

[Former name in HyperStore 7.x and earlier: `phonehome_proxy_port`]

If you want HyperStore to use a local forward proxy when the Smart Support feature sends daily system diagnostics packages to Cloudian Support, use this setting to specify the proxy's port number.

Default = 3128

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set common.phoneHome.proxy.port=<integer>
# hsctl config apply common
```


No service restart is required.

common.phoneHome.url

[Former name in HyperStore 7.x and earlier: *phonehome_url*]

S3 URL to which to upload system-wide diagnostics data each day. By default this is the S3 URL for Cloudbian Support.

If you set this to a different S3 destination, include the HTTP or HTTPS protocol part of the URL (*http://* or *https://*), as well as the port number. Also, if you set this to a different S3 destination you must set the "**common.phoneHome.bucket**" (page 744), "**common.phoneHome.auth.accessKey**" (page 744), and "**common.phoneHome.auth.secretKey**" (page 744).

Default = `https://s3-support.cloudbian.com:443`

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set common.phoneHome.url=<string>
# hscctl config apply common
```

No service restart is required.

Note If you set *common.phoneHome.url* to a URI for your own HyperStore S3 Service (rather than the Cloudbian Support URI), and if your S3 Service is using HTTPS, then your S3 Service's SSL certificate must be a CA-verified certificate — not a self-signed certificate. By default the phone home function cannot upload to an HTTPS URI that's using a self-signed certificate. If you require that the upload go to an HTTPS URI that's using a self-signed certificate, contact Cloudbian Support for guidance on modifying the phone home launch script. For information on HTTPS set-up for the S3 Service, see "**Security and Privacy Features**" (page 177).

common.rdma.enabled

This setting controls whether NVIDIA® GPUDirect® Storage (GDS) is enabled for the HyperStore system, *true* or *false*. When enabled, this functionality uses Remote Direct Memory Access (RDMA) for direct transfer of object data between HyperStore and GPU memory on client hosts.

Note that:

- This functionality is supported only for new installs of HyperStore version 8.2 and later.
- Setting *common.rdma.enabled* to *true* is just one part of the requirements for using this functionality.

For more information see "**GPUDirect Storage Feature Overview**" (page 143) and "**GDS Setup**" (page 146).

Default = `false`

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set common.rdma.enabled=<boolean>
# hscctl config apply common
# hscctl service restart s3 hyperstore cmc --nodes=ALL
```

common.services.autoRestart

[Former name in HyperStore 7.x and earlier: *service_restart_on_failure*]

When to automatically restart services:

- If this is set to *onFailure*, the system will automatically restart a service if the service crashes (stops with a non-zero exit code -- a failure code).
- If this is set to *never*, the system will not automatically restart a service that crashes.

When set to *onFailure*, this feature is implemented for each node individually. For example, if the S3 Service crashes on a particular node, then the S3 Service on will be automatically restarted on that node.

The first automatic attempt to restart a crashed service will occur 30 seconds after the crash. If four attempts (automatic or manual) to restart a service on a node fail within a 10 minute interval, no more restart attempts of that service on that node are allowed until the end of that 10 minute interval, when restart attempts can resume. (If you wish you can release this constraint by running *systemctl reset-failed <service>*.)

This setting applies only to the S3 Service, the Admin Service, the HyperStore Service, and the Cassandra Service (Metadata DB). Automatic restart is not supported for other services.

Default = *onFailure*

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set common.services.autoRestart=<string>
# hsctl config apply common
# hsctl service restart s3 hyperstore cassandra --nodes=ALL
```

common.services.enabled

[Former name in HyperStore 7.x and earlier: *service_starts_on_boof*]

Whether to have HyperStore services start on host reboot, *true* or *false*.

Optionally this setting can be overridden on a per-service basis, by the *<service>.enabled* settings.

Default = *true*

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set common.services.enabled=<boolean>
# hsctl config apply common
# hsctl service restart all --nodes=ALL
```

common.services.heap.dumpOnOOM

[Former name in HyperStore 7.x and earlier: *cloudian_heapdump_on_out_of_memory*]

Whether to enable Java heap dumps, for when out-of-memory errors occur for HyperStore's Java-based service applications. This applies to the Admin Service, Cassandra Service, HyperStore Service, IAM Service, Monitoring Data Collector Service, S3 Service, and SQS Service. Dump file paths are */var/log/cloudian/<service>.hprof*. The dump file is in HPROF binary format.

Default = *true*

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set common.services.heap.dumpOnOOM=<boolean>
# hsctl config apply common
# hsctl service restart all --nodes=ALL
```

Note that:

- The `common.services.heap.dumpOnOOM` setting can be overridden for individual services with the setting `<service>.heap.dumpOnOOM`
- The dump path for individual services can be customized with the setting `<service>.heap.dumpPath`
- For these service-specific settings the supported `<service>` values are:
 - admin
 - cassandra
 - hyperstore
 - iam
 - monitoring.dataCollector
 - s3
 - sqs

common.sysInfo.auth.accessKey

[Former name in HyperStore 7.x and earlier: `sysinfo.accessKey`]

This setting gives you the ability to set a different destination S3 access key for uploading on-demand Node Diagnostics packages than is used for uploading daily Smart Support packages, if you wish. If this setting is empty (as it is by default) then the corresponding setting from the Smart Support configuration -- "[com-mon.phoneHome.auth.accessKey](#)" (page 744) -- is used for Node Diagnostics package processing as well.

Default = empty

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set common.sysInfo.auth.accessKey=<string>
# hsctl config apply common
```

No service restart is required.

common.sysInfo.auth.secretKey

[Former name in HyperStore 7.x and earlier: `sysinfo.secretKey`]

This setting gives you the ability to set a different destination S3 secret key for uploading on-demand Node Diagnostics packages than is used for uploading daily Smart Support packages, if you wish. If this setting is empty (as it is by default) then the corresponding setting from the Smart Support configuration -- "[com-mon.phoneHome.auth.secretKey](#)" (page 744) -- is used for Node Diagnostics package processing as well.

Default = empty

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set common.sysInfo.auth.secretKey=<secret>
# hsctl config apply common
```

No service restart is required.

common.sysInfo.bucket

[Former name in HyperStore 7.x and earlier: `sysinfo.bucket`]

This setting gives you the ability to set a different destination S3 bucket for uploading on-demand Node Diagnostics packages than is used for uploading daily Smart Support packages, if you wish. If this setting is empty (as it is by default) then the corresponding setting from the Smart Support configuration -- "[common.phoneHome.bucket](#)" (page 744) -- is used for Node Diagnostics package processing as well.

Default = empty

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set common.sysInfo.bucket=<string>
# hsctl config apply common
```

No service restart is required.

common.sysInfo.proxy.auth.password

[Former name in HyperStore 7.x and earlier: `sysinfo.proxy.password`]

This setting gives you the ability to set a different forward proxy password for uploading on-demand Node Diagnostics packages than is used for uploading daily Smart Support packages, if you wish. If this setting is empty (as it is by default) then the corresponding setting from the Smart Support configuration -- "[common.phoneHome.proxy.auth.password](#)" (page 747) -- is used for Node Diagnostics package processing as well.

Default = empty

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set common.sysInfo.proxy.auth.password=<secret>
# hsctl config apply common
```

No service restart is required.

common.sysInfo.proxy.auth.username

[Former name in HyperStore 7.x and earlier: `sysinfo.proxy.username`]

This setting gives you the ability to set a different forward proxy username for uploading on-demand Node Diagnostics packages than is used for uploading daily Smart Support packages, if you wish. If this setting is empty (as it is by default) then the corresponding setting from the Smart Support configuration -- "[common.phoneHome.proxy.auth.username](#)" (page 747) -- is used for Node Diagnostics package processing as well.

Default = empty

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set common.sysInfo.proxy.auth.username=<string>
# hsctl config apply common
```

No service restart is required.

common.sysInfo.proxy.enabled

For information on this functionality see "[common.sysInfo.proxy.host](#)" (page 753).

Although `common.sysInfo.proxy.enabled` defaults to `true`, this functionality is not actually enabled unless you set a value for `common.sysInfo.proxy.host`.

If you configure `common.sysInfo.proxy.host` and potentially the other settings for this proxy-using functionality (such as the proxy authentication settings), and then subsequently you want to stop using the proxy without having to delete this configuration information that you've set up, you can set `common.sysInfo.proxy.enabled` to `false`.

Default = true

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set common.sysInfo.proxy.enabled=<boolean>
# hscctl config apply common
```

No service restart is required.

common.sysInfo.proxy.host

[Former name in HyperStore 7.x and earlier: `sysinfo.proxy.host`]

This setting gives you the ability to set a different forward proxy host for uploading on-demand Node Diagnostics packages than is used for uploading daily Smart Support packages, if you wish. If this setting is empty (as it is by default) then the corresponding setting from the Smart Support configuration -- "[common.phoneHome.proxy.host](#)" (page 748) -- is used for Node Diagnostics package processing as well.

Default = empty

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set common.sysInfo.proxy.host=<string>
# hscctl config apply common
```

No service restart is required.

common.sysInfo.proxy.port

[Former name in HyperStore 7.x and earlier: `sysinfo.proxy.port`]

This setting gives you the ability to set a different forward proxy port for uploading on-demand Node Diagnostics packages than is used for uploading daily Smart Support packages, if you wish. If this setting is empty (as it is by default) then the corresponding setting from the Smart Support configuration -- "[common.phoneHome.proxy.port](#)" (page 748) -- is used for Node Diagnostics package processing as well.

Default = empty

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set common.sysInfo.proxy.port=<integer>
# hscctl config apply common
```

No service restart is required.

common.sysInfo.url

[Former name in HyperStore 7.x and earlier: `sysinfo.uri`]

This setting gives you the ability to set a different destination S3 URL for uploading on-demand Node Diagnostics packages than is used for uploading daily Smart Support packages, if you wish. If this setting is empty (as it is by default) then the corresponding setting from the Smart Support configuration -- "[common.phoneHome.url](#)" (page 749) -- is used for Node Diagnostics package processing as well.

For an overview of this feature set see "[Smart Support and Diagnostics Feature Overview](#)" (page 117).

Default = empty

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set common.sysInfo.url=<string>
# hsctl config apply common
```

No service restart is required.

common.user.allowDeleteWithBuckets

[Former name in HyperStore 7.x and earlier: `allow_delete_users_with_buckets`]

This setting controls system behavior regarding the deletion of user accounts that still own buckets:

- If `common.user.allowDeleteWithBuckets` is set to `true`, then a user who owns buckets can be deleted via the CMC or the Admin API, and the system will not only delete the user but will also **automatically delete the user's buckets and all the data in those buckets**. This includes buckets created by IAM users under the user account root, if any. The deleted data will not be recoverable.
- If `common.user.allowDeleteWithBuckets` is set to `false`, then the system will not allow a user who owns buckets to be deleted via the CMC or the Admin API. Instead, the user or an administrator must first delete all buckets owned by the user -- via the CMC or a different S3 client application -- including buckets created by IAM users under the user account root, if any. Only after all such buckets are deleted can the user then be deleted via the CMC or the Admin API.

Default = `false` (exception: if your HyperStore system was originally installed as version 7.2.x or earlier, the default in your system is `true`).

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set common.user.allowDeleteWithBuckets=<boolean>
# hsctl config apply common
# hsctl service restart s3 --nodes=ALL
```

Note Regarding an administrator's ability to delete a user's buckets or resource through the CMC, see the configuration setting "[cmc.ui.admin.manageUsers.view.objectDataGrantees](#)" (page 739).

common.user.allowDeleteWithIAMResources

This setting controls system behavior regarding the deletion of user accounts that still own IAM groups, users, roles, and or policies:

- If `common.user.allowDeleteWithIAMResources` is set to `true`, then a user who owns IAM resources can be deleted via the CMC or the Admin API, and the system will not only delete the user but will also **automatically delete the user's IAM resources**. The deleted IAM resources will not be recoverable.
- If `common.user.allowDeleteWithIAMResources` is set to `false`, then the system will not allow a user who owns IAM resources to be deleted via the CMC or the Admin API. Instead, the user or an administrator

must first delete all IAM resources owned by the user -- via the CMC or a different S3 client application. Only after all such IAM resources are deleted can the user then be deleted via the CMC or the Admin API.

Default = false

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set common.user.allowDeleteWithIAMResources=<boolean>
# hsctl config apply common
# hsctl service restart s3 iam cmc --nodes=ALL
```

Note Regarding an administrator's ability to delete a user's buckets or resources through the CMC, see the configuration setting "[cmc.ui.admin.manageUsers.view.objectDataGrantees](#)" (page 739).

common.user.name.maxLength

[Former name in HyperStore 7.x and earlier: *cloudian_userid_length*]

Maximum number of characters allowed in a HyperStore user ID. The highest value you can set this to is 256.

Default = 64

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set common.user.name.maxLength=<integer>
# hsctl config apply common
# hsctl service restart s3 cmc --nodes=ALL
```

Note This maximum applies also to the user "full name" (for example "John Q. Public"). For example if this is set to 64, then when you are creating a user through the CMC or the Admin API the user ID can be a maximum of 64 characters long, and also the user full name can be a maximum of 64 characters long including spaces.

common.user.password.dupCharRatioLimit

[Former name in HyperStore 7.x and earlier: *user_password_dup_char_ratio_limit*]

When a CMC user creates a new password, no more than this percentage of characters in the new password can be characters that are in the user's current password. For example, if this is set to 50, then no more than 50% of the characters in the new password can be characters that are in the user's current password. The system will reject a user's attempt to set a new password that does not meet this requirement.

Note that:

- If this is set to 0, then none of the characters in a user's new password can be characters that are in the user's current password.
- If this is set to -1, then this password requirement is disabled.

Default = -1

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hscctl config set common.user.password.dupCharRatioLimit=<integer>
# hscctl config apply common
# hscctl service restart s3 cmc --nodes=ALL
```

common.user.password.lock.durationSec

[Former name in HyperStore 7.x and earlier: `user_password_lock_durationsec`]

Applicable only if "[common.user.password.lock.enabled](#)" (page 756) is set to `true`.

If within an interval of `common.user.password.lock.durationSec` seconds, a user tries "[common.user.password.lock.maxFailedAttempts](#)" (page 757) number of times to log in to the CMC with an incorrect password, then the system locks out that user for an interval of `common.user.password.lock.durationSec` seconds. During this lock-out interval the system will not allow the user to log into the CMC even if the user supplies the correct password. If during the lock-out interval the user again attempts to log in with an incorrect password, the lock-out interval starts over. After the lock-out interval expires the user will be allowed to log into the CMC if the user supplies the correct password.

For example, if `common.user.password.lock.durationSec` is set to 1800 and "[common.user.password.lock.maxFailedAttempts](#)" (page 757) is set to 6, then the system will lock out a user who tries 6 times within an 1800 second (30 minute) period to log into the CMC with an incorrect password. The lock-out will last 30 minutes, starting from the 6th invalid login attempt. During the 30 minute lock-out period:

- If the user tries again to log in with an incorrect password, this restarts the 30 minute lock-out period.
- If the user does not try again to log in with an incorrect password, the lock-out period automatically expires at the end of the 30 minutes, and the user is then allowed to log in if they supply the correct password.
- Optionally, the system administrator can intervene to release the lock on a user -- before the lock reaches its automatic expiration -- by using the CMC (see "[Unlock a User Who's Locked Out of the CMC](#)" (page 411)) or the Admin API (see [POST /user/unlock](#) [Unlock a locked-out user](#)).

Default = 1800

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set common.user.password.lock.durationSec=<integer>
# hscctl config apply common
# hscctl service restart s3 cmc --nodes=ALL
```

Note If a system administrator becomes locked out of the CMC -- due to too many login attempts within an incorrect password -- the lock on that administrator can be released by a different system administrator using the CMC, or by the Admin API method mentioned above. Alternatively, as with any type of locked out user the lock-out will release automatically after the configurable time interval.

common.user.password.lock.enabled

[Former name in HyperStore 7.x and earlier: `user_password_lock_enabled`]

This setting controls whether or not the CMC password lock feature is enabled:

- If `common.user.password.lock.enabled` is set to `true`, the CMC password lock feature is enabled and its behavior is configured by the "[common.user.password.lock.durationSec](#)" (page 756) and "[common.user.password.lock.maxFailedAttempts](#)" (page 757) settings.

Note The CMC password lock feature applies only to users for which local authentication (based on a user ID and password defined within HyperStore) is performed, not users whose CMC login attempts are authenticated by reference to an LDAP system. For users authenticated by an LDAP system you can configure a password lockout policy within the LDAP system if desired.

- If `common.user.password.lock.enabled` is set to `false`, the CMC password lock feature is disabled.

Default = `true` (exception: if your HyperStore system was originally installed as version 7.2.x or earlier, the default in your system is `false`).

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set common.user.password.lock.enabled=<boolean>
# hsctl config apply common
# hsctl service restart s3 cmc --nodes=ALL
```

common.user.password.lock.maxFailedAttempts

[Former name in HyperStore 7.x and earlier: `user_password_lock_maxfailedattempts`]

Applicable only if "**common.user.password.lock.enabled**" (page 756) is set to `true`.

For a description of the `common.user.password.lock.maxFailedAttempts` setting see "**common.user.password.lock.durationSec**" (page 756). These two settings work in combination.

Default = 6

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set common.user.password.lock.maxFailedAttempts=<integer>
# hsctl config apply common
# hsctl service restart s3 cmc --nodes=ALL
```

common.user.password.maxLength

[Former name in HyperStore 7.x and earlier: `admin.user.password.length`]

Maximum allowed character length for users' Cloudian Management Console login passwords.

Default = 64

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set common.user.password.maxLength=<integer>
# hsctl config apply common
# hsctl service restart s3 cmc --nodes=ALL
```

common.user.password.minLength

[Former name in HyperStore 7.x and earlier: `user_password_min_length`]

For users' CMC passwords, the minimum required length in number of characters. The system will reject a user's attempt to set a new password that does not meet this requirement.

Default = 9

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set common.user.password.minLength=<integer>
# hsctl config apply common
# hsctl service restart s3 cmc --nodes=ALL
```

common.user.password.rotation.expiration

[Former name in HyperStore 7.x and earlier: user_password_rotation_expiration]

After a CMC user has had the same password for this many days, the password expires and the CMC will require the user to create a new password before they can log in again. This will be enforced by the CMC's login function.

If this is set to 0, then this password requirement is disabled.

Note that if you change this setting, the lifespan of each user's **existing** password is considered to have started **when the password was created**. For example, if you change this setting's value from 0 to 90, then users whose existing passwords are already older than 90 days old will be required to change their passwords the next time that they log into the CMC; and users whose existing passwords are 60 days old will need to change their passwords in 30 days.

Default = 0

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set common.user.password.rotation.expiration=<integer>
# hsctl config apply common
# hsctl service restart s3 cmc --nodes=ALL
```

common.user.password.rotation.gracePeriod

[Former name in HyperStore 7.x and earlier: user_password_rotation_graceperiod]

When a CMC user creates a new password, they must wait at least this many days before replacing that password with another new password. For example if you set *common.user.password.rotation.gracePeriod* to 7, a user who successfully creates a new CMC password would not be able to create their next new CMC password until 7 days have passed.

If this is set to 0, then this password requirement is disabled.

Default = 0

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set common.user.password.rotation.gracePeriod=<integer>
# hsctl config apply common
# hsctl service restart s3 cmc --nodes=ALL
```

common.user.password.uniqueGenerations

[Former name in HyperStore 7.x and earlier: user_password_unique_generations]

When a CMC user creates a new password, the new password cannot be the same as one of the user's past passwords, going back this many passwords into the user's password history. The system will reject a user's attempt to set a new password that does not meet this requirement.

For example if you set this to 10, then a user's new password cannot match against any of the user's past 10 passwords.

If this is set to 0, then this password requirement is disabled.

Default = 0

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set common.user.password.uniqueGenerations=<integer>
# hscctl config apply common
# hscctl service restart s3 cmc --nodes=ALL
```

common.xForwardedHeaders.enabled

If you are using reverse proxies or load balancers in front of the HyperStore S3 Service, then by default the "source" of requests incoming to the S3 Service is the proxies / load balancers, and the source IP address is the IP address of a proxy / load balancer. If you want HyperStore's S3 request logging to capture the true source (origin) IP address of incoming requests -- or if you want to configure S3 bucket policies that use the true source IP address of requests as a condition -- you have these options:

- **If you are using pass-through proxies or pass-through Layer 7 load balancers** -- i.e. they do not terminate incoming TLS connections but instead pass through the TLS-encrypted request data -- then for the S3 Service (request logging and bucket policies) to have access to the true source IP of incoming requests, you must use PROXY Protocol between the proxies / load balancers and the S3 Service. This protocol allows the proxies / load balancers to pass the IP addresses of originating clients to the S3 Service along with the S3 requests. For more information about enabling PROXY Protocol support on the S3 Service side, see the HyperStore configuration setting "[s3.proxyProtocol.enabled](#)" (page 826). For guidance on configuring the proxies / load balancers consult with Cloudbian Sales Engineering or Support.
- **If you are using load balancers that operate at a lower layer than Layer 7** -- then again your only option is to use the PROXY Protocol as described above.
- **If are using proxies or Layer 7 load balancers that terminate incoming TLS connections**, then you can **either** use PROXY Protocol as described above, **or** you can make use of HTTP "X-Forwarded-For" headers in incoming requests by configuring these HyperStore settings:
 - To use HTTP "X-Forwarded-For" headers to capture true source IP addresses in S3 request logging, set `common.xForwardedHeaders.enabled` to `true` (it is `false` by default).
 - To use HTTP "X-Forwarded-For" headers to capture true source IP addresses in S3 request logging and to **also** allow HyperStore to use HTTP "X-Forwarded-For" headers when enforcing bucket policies (in particular, policy conditions based on source IP address), set `common.xForwardedHeaders.enabled` to `true` and set "[common.xForwardedHeaders.trusted](#)" (page 760) to `true` (both are `false` by default).

These "X-Forwarded-For" related settings also apply to the IAM Service (and the SQS Service [if you have enabled it](#)). See the Note below for further information.

Default = false

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node (if you are using the SQS Service then also include `sqs` in the list of services to restart):

```
# hscctl config set common.xForwardedHeaders.enabled=<boolean>
# hscctl config apply common
# hscctl service restart s3 iam --nodes=ALL
```

Note If you want to have different behavior for the S3 Service, the IAM Service, and the SQS Service request logs in regard to use of the HTTP *X-Forwarded-For* header, you can use any of these service-specific settings (which override the *common.xForwardedHeaders.enabled* setting on a per-service basis):

```
s3.xForwardedHeaders.enabled
iam.xForwardedHeaders.enabled
sqs.xForwardedHeaders.enabled
```

For example, if you want to utilize the HTTP *X-Forwarded-For* header only for S3 request logs but not for IAM or SQS request logs, then rather than setting *common.xForwardedHeaders.enabled* to *true* you can leave *common.xForwardedHeaders.enabled* at its default value of *false* and do the following:

```
# hscctl config set s3.xForwardedHeaders.enabled=true
# hscctl config apply s3
# hscctl service restart s3 --nodes=ALL
```

common.xForwardedHeaders.trusted

This setting is applicable only if you have set *common.xForwardedHeaders.enabled* to *true*.

For a description of the purpose of the *common.xForwardedHeaders.trusted* setting see ["common.xForwardedHeaders.enabled"](#) (page 759).

Default = false

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set common.xForwardedHeaders.trusted=<boolean>
# hscctl config apply ALL
# hscctl service restart s3 iam --nodes=ALL
```

Note If you want to have different behavior for the S3 Service and the IAM Service, you can use these service-specific settings (which override the *common.xForwardedHeaders.trusted* setting on a per-service basis):

```
s3.xForwardedHeaders.trusted
iam.xForwardedHeaders.trusted
```

13.2.5. HyperStore Shell Settings

The table below lists configuration settings in the *hsh* setting group, which configures the HyperStore Shell.

Setting	Default Value
hsh.enabled (page 761)	false
hsh.idleTimeout (page 761)	30m
hsh.ldap.bindDN (page 761)	
hsh.ldap.bindPassword (page 761)	

13.2.5.1. hsh.* Setting Descriptions

hsh.enabled

For information about this setting see ["Enabling the HSH" \(page 106\)](#).

Default = false

hsh.idleTimeout

For information about this setting see ["Configurable HSH Idle Timeout" \(page 110\)](#).

Default = 30m

hsh.lldap.bindDN

For information about this setting see ["LDAP Authentication of System Administrators and HSH Users" \(page 281\)](#).

Default = empty

hsh.lldap.bindPassword

For information about this setting see ["LDAP Authentication of System Administrators and HSH Users" \(page 281\)](#).

Default = empty

13.2.6. HyperStore Service Settings

The table below lists configuration settings in the *hyperstore* setting group, which configures the [HyperStore Service](#). When applicable the settings in the table are organized into sub-groups.

Setting Sub-Group	Setting	Default Value
hyperstore.cassandra	hyperstore.cassandra.maxActive (page 763)	specific to your system
	hyperstore.cassandra.maxActiveWait (page 764)	9000
	hyperstore.cassandra.maxIdle	20
	hyperstore.cassandra.thrift.timeout (page 764)	15000
hyperstore.cleanup	hyperstore.cleanup.objectDelete.gracePeriod (page 765)	86400
	hyperstore.cleanup.objectDelete.queue.maxLength (page 765)	2000
	hyperstore.cleanup.operations.maxPerDC (page 766)	1
hyperstore.disk	hyperstore.disk.check.interval (page 767)	4320
	hyperstore.disk.fail.action (page 767)	disable
	hyperstore.disk.threshold.stopRepair (page 768)	95
hyperstore.heap	hyperstore.heap.auto.initPercent (page 769)	25
	hyperstore.heap.auto.limit (page 769)	30g

Setting Sub-Group	Setting	Default Value
	hyperstore.heap.auto.maxPercent (page 769)	15
	hyperstore.heap.maxSize (page 770)	not set
	hyperstore.heap.minSize (page 770)	not set
hyperstore.logging	hyperstore.logging.level (page 770)	WARN
	hyperstore.logging.rollover.* settings	see "Log Rolling Settings" (page 689) for default values
hyper-store.messagebus	hyperstore.messagebus.connections.max (page 771)	specific to your system
	hyperstore.messagebus.connections.timeout (page 772)	300
hyperstore.repair	hyperstore.repair.EC.queue.size (page 772)	2000
	hyperstore.repair.merkleTreeRequest.timeout (page 772)	360
	hyperstore.repair.objectQueue.size (page 772)	1000
	hyperstore.repair.objectQueue.timeout (page 773)	120
	hyperstore.repair.proactive.interval (page 773)	60
	hyperstore.repair.proactive.retryQueue.timeout (page 774)	240
	hyperstore.repair.scheduled.EC.interval (page 777)	41760
	hyperstore.repair.scheduled.RF.interval (page 777)	43200
	hyperstore.repair.scheduled.cassandra.interval (page 775)	10080
	hyper-store.repair.scheduled.cassandra.range.enabled (page 775)	true
	hyperstore.repair.scheduled.cassandra.timeout (page 775)	7200
	hyper-store.repair.scheduled.computeDigest.frequency (page 776)	0
hyperstore.threadPool	hyperstore.threadPool.autoRepair (page 779)	specific to your system
	hyperstore.threadPool.cleanup.jobs (page 780)	10
	hyperstore.threadPool.cleanup.session (page 780)	specific to your system
	hyperstore.threadPool.decommission (page 781)	specific to your system
	hyperstore.threadPool.messaging (page 781)	specific to your system
	hyperstore.threadPool.rangeRepair (page 782)	specific to your system
	hyperstore.threadPool.repair.digestIndex (page 782)	specific to your system
	hyperstore.threadPool.repair.session (page 783)	specific to your system
	hyperstore.threadPool.repairEC.digestRequest	30

Setting Sub-Group	Setting	Default Value
	(page 783)	
	hyperstore.threadPool.repairEC.rocksDBScan (page 783)	30
	hyperstore.threadPool.repairEC.sessionScan (page 784)	50
	hyperstore.threadPool.repairEC.task (page 784)	60
	hyperstore.threadPool.session.rangeDownload (page 785)	specific to your system
	hyperstore.threadPool.session.rangeUpload (page 785)	specific to your system
	hyperstore.threadPool.streamOutbound (page 786)	specific to your system
hyperstore.threads	hyperstore.threads.idleTimeout (page 786)	30000
	hyperstore.threads.max (page 786)	specific to your system
	hyperstore.threads.maxRead (page 787)	specific to your system
	hyperstore.threads.maxRepair (page 787)	50
	hyperstore.threads.maxWrite (page 788)	specific to your system
	hyperstore.threads.min (page 788)	100
n/a (miscellaneous individual settings)	hyperstore.directory.data (page 766)	/var/lib/cloud-ian/hyperstore/data
	hyperstore.enabled (page 769)	true
	hyperstore.lowRes.idleTimeout (page 771)	10000
	hyperstore.secureDelete (page 778)	false
	hyperstore.stackSize (page 778)	not set
	hyperstore.startTimeout	300
	hyperstore.storagePolicy.max (page 779)	25

13.2.6.1. hyperstore.* Setting Descriptions

hyperstore.cassandra.maxActive

[Former name in HyperStore 7.x and earlier: `cassandra_max_active`]

The maximum allowed number of simultaneously active connections in a HyperStore node's Cassandra (Metadata DB) connection pool. If this limit has been reached and a thread requires a new connection to Cassandra, the thread will wait for a period configured by "[hyperstore.cassandra.maxActiveWait](#)" (page 764) (default 9 seconds) before returning an error to the client.

If this is set to a negative value (e.g. -1) this disables the limit on active connections.

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set hyperstore.cassandra.maxActive=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore s3 --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use *hsctl config set* to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

hyperstore.cassandra.maxActiveWait

[Former name in HyperStore 7.x and earlier: *cassandra.cluster.MaxWaitTimeWhenExhausted*]

If "**hyperstore.cassandra.maxActive**" (page 763) has been reached for a HyperStore node's Cassandra connection pool and a thread requires a new connection to Cassandra, the thread will wait this many milliseconds for a connection to become available before returning an error to the client. (Connection pool queue timeout.)

Default = 9000

For a diagram showing the place of this timeout within the S3 request processing flow, see "**hyperstore.cassandra.thrift.timeout**" (page 764).

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set hyperstore.cassandra.maxActiveWait=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore s3 --nodes=ALL
```

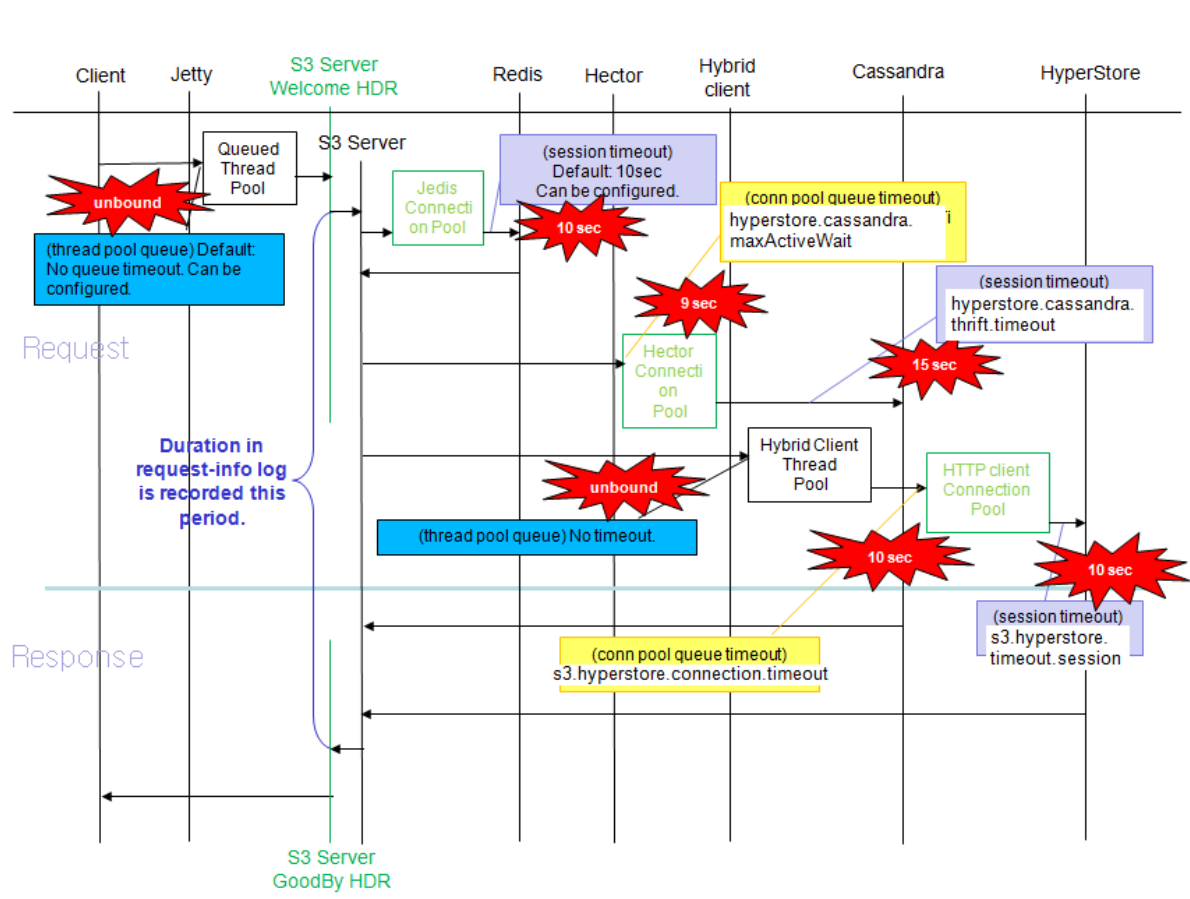
hyperstore.cassandra.thrift.timeout

[Former name in HyperStore 7.x and earlier: *cassandra.cluster.CassandraThriftSocketTimeout*]

After a HyperStore node submits a request to a Cassandra instance via an established Thrift socket, the amount of time in milliseconds to wait for a response from Cassandra before failing the operation. (Session timeout.)

Default = 15000

The diagram below shows the place of *hyperstore.cassandra.thrift.timeout* and other timeouts within the S3 Service's request processing flow.



To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set hyperstore.cassandra.thrift.timeout=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore s3 --nodes=ALL
```

hyperstore.cleanup.objectDelete.gracePeriod

[Former name in HyperStore 7.x and earlier: `cleanup.session.delete.graceperiod`]

During [hsstool cleanup](#) or [hsstool cleanuppec](#) operations, the system will only consider an object for deletion if at least this many seconds have passed since the object's Last Modified timestamp. Objects with a Last Modified timestamp more recent than this are excluded from the evaluation that determines whether or not an object's data should be on a particular node.

Default = 86400 (24 hours)

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set hyperstore.cleanup.objectDelete.gracePeriod=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

hyperstore.cleanup.objectDelete.queue.maxLength

[Former name in HyperStore 7.x and earlier: `cleanup.session.deleteobjs.queue.maxlength`]

During [hsstool cleanup](#) or [hsstool cleanupec](#) operations, the target maximum number of object delete jobs to queue for processing. Object delete jobs are placed in queue by a cleanup job that detects discrepancies between object metadata in Cassandra versus object metadata on the local node.

Default = 2000

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set hyperstore.cleanup.objectDelete.queue.maxLength=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore s3 --nodes=ALL
```

hyperstore.cleanup.operations.maxPerDC

[Former name in HyperStore 7.x and earlier: `max.cleanup.operations.perdc`]

Maximum number of [hsstool cleanup](#) or [hsstool cleanupec](#) operations that the system will allow to run concurrently, per data center.

The limit is applied separately to `cleanup` and `cleanupec` operations. For example, if `hyperstore.cleanup.operations.maxPerDC` is set to 1 (as it is by default), then within a DC you can run one `cleanup` operation at a time and one `cleanupec` operation at a time. The limit does not prevent you from running one `cleanup` operation and one `cleanupec` operation concurrently.

If you have multiple DCs in your HyperStore system, the limit is applied separately to each DC. For example if you have two data centers named DC1 and DC2, and if `hyperstore.cleanup.operations.maxPerDC` is set to 1, you can run one `cleanup` operation and one `cleanupec` operation in DC1 at the same time as you are running one `cleanup` operation and one `cleanupec` operation in DC2.

If you are considering raising this limit from its default of 1, first consult with Cloudian Support.

Default = 1

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set hyperstore.cleanup.operations.maxPerDC=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

hyperstore.directory.data

[Former name in HyperStore 7.x and earlier: `hyperstore_data_directory`]

A quote-enclosed, comma-separated list of mount points to use for object data storage. In a production environment, use dedicated disks for object data storage. Do not use the same disk(s) that are storing the OS and Cassandra.

Do not change the `hyperstore.directory.data` setting once your cluster is operational.

Do not use symbolic links when specifying your mount points for the `hyperstore.directory.data` setting. The HyperStore system does not support symbolic links for these directories.

Example of a multiple mount point configuration = `"/cloudian1,/cloudian2,/cloudian3,/cloudian4"`

Default = Set during installation based on operator input, if multiple mount points (or if only one mount point, `/var/lib/cloudian/hyperstore/data`)

IMPORTANT ! Changing this setting after your system has been in use can potentially result in data loss. If for some reason you are wanting to change this setting, first consult with Clouddian Support.

hyperstore.disk.check.interval

[Former name in HyperStore 7.x and earlier: *hyperstore_disk_check_interval*]

The interval (in minutes) at which each HyperStore Service instance will run a check to see if there is significant imbalance in disk usage on its host node. If an imbalance is found, then one or more tokens are automatically moved from the over-used disk(s) to one or more less-used disk(s) on the same host.

Specifically, when the disk balance check is run at the interval set by *hyperstore.disk.check.interval*, token migration is triggered if a disk's utilization percentage differs from the average disk utilization percentage on the node by more than 10. For example:

- If the average disk space utilization on a node is 35%, and the disk space utilization for Disk4 is 55%, then one or more tokens will be migrated away from Disk4 to other disks on the node (since the actual delta of 20% exceeds the maximum allowed delta of 10%).
- If the average disk utilization on a node is 40%, and the disk utilization for Disk7 is 25%, then one or more tokens will be migrated to Disk7 from the other disks on the node (since the actual delta of 15% exceeds the maximum allowed delta of 10%).

For more information see ["Automated Disk Usage Management Feature Overview"](#) (page 124).

Default = 4320 (72 hours)

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set hyperstore.disk.check.interval=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

Note This feature applies only to HyperStore data disks (on which are stored object data). It does not apply to disks that are storing only the OS and Cassandra (the Metadata DB).

hyperstore.disk.fail.action

[Former name in HyperStore 7.x and earlier: *disk.fail.action*]

The automated action for the system to take in the event that read or write errors occur for a particular HyperStore data disk. Supported automated actions are:

- **disable** — With this setting, if disk errors are detected the system will automatically unmount the disk and mark it as disabled. The system will also automatically transfer all of the disabled disk's storage tokens to the remaining disks on the host, in an approximately balanced way. The data from the disabled disk will **not** be recreated on the other disks (repair operations will not be automatically triggered, and even if you triggered repair operations manually the data from the disabled disk would not be recreated on the other disks because with this disk disabling approach the moved tokens are assigned updated timestamps).

When a disk is disabled in this way, writes of new or updated S3 object data that would have gone to the disabled disk will go to the other disks on the host instead. Meanwhile existing S3 object data will be

unreadable on the host. Whether the system as a whole can still provide S3 clients with read access to the affected S3 objects depends on the storage policies with which the objects are associated, and on the availability of other replicas or erasure coded fragments within the cluster.

When a disk is disabled in this way, an alert is generated and the disk will show as disabled in the Disk Detail Info section of the CMC's **Node Status** page.

Note When you subsequently perform the ["Enabling a HyperStore Data Disk" \(page 640\)](#) operation or the ["Replacing a HyperStore Data Disk" \(page 642\)](#) operation, the tokens will automatically be moved back to the re-enabled or replacement disk. Data written in association with the tokens when they were on the other disks will remain on those other disks and does not need to be moved to the re-enabled or replacement disk. For information on how HyperStore tracks token location over time so that objects can be written to and read from the correct disks, see ["Dynamic Object Routing" \(page 127\)](#).

- **none** — With this option the system will not automatically disable a disk for which errors have occurred. Each disk error will trigger an alert, however.

By default, your configured automatic disk failure action will be triggered if 10 "HSDISKERROR" error messages for the disk occur in the HyperStore Service application log within a 5 minute span.

Your configured automatic disk failure action will also be triggered if the HyperStore drive audit feature detects that a disk is in a read-only condition.

Note The automatic disk disabling feature works only if you have multiple HyperStore data disks on the host. If there is only one HyperStore data disk on the host, the system will not automatically disable the disk even if errors are detected.

Also, the automatic disk failure handling feature does not work correctly in Xen, Logical Volume Manager (LVM), or Amazon EC2 environments. Contact Cloudbian Support if you are using any of these technologies.

Default = disable

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set hyperstore.disk.fail.action=<string>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

Note You also have the option of configuring this setting in the **System Settings** section of the CMC's ["Configuration Settings" \(page 498\)](#) page. In the CMC UI this setting is called "HyperStore Disk Failure Action".

hyperstore.disk.threshold.stopRepair

[Former name in HyperStore 7.x and earlier: *max.diskusage.percentage*]

A [hsstool repair](#) operation will fail and stop if any HyperStore data disk encompassed by the repair operation is more than this percent full.

Default = 95

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set hyperstore.disk.threshold.stopRepair=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

hyperstore.enabled

This setting determines whether the HyperStore Service is enabled, *true* or *false*. This also determines whether the HyperStore Service will start automatically when a host reboots.

Default = true

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set hyperstore.enabled=<boolean>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

hyperstore.heap.auto.initPercent

[Former name in HyperStore 7.x and earlier: cloudian_hss_init_heap_percent]

On each node, this setting determines the initial heap size (JAVA_OPTS `-Xms` value) for the HyperStore Service application, as a percentage of the `-Xmx` value.

Default = 25

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set hyperstore.heap.auto.initPercent=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

hyperstore.heap.auto.limit

[Former name in HyperStore 7.x and earlier: cloudian_hss_heap_limit]

On each node, this setting determines the maximum heap size limit for the HyperStore Service application. The JAVA_OPTS `-Xmx` value passed to the JVM will be the **lower** of this setting's value and the percent-of-system-RAM value set by the ["hyperstore.heap.auto.maxPercent"](#) (page 769) setting.

Default = 30g

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set hyperstore.heap.auto.limit=<string>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

hyperstore.heap.auto.maxPercent

[Former name in HyperStore 7.x and earlier: cloudian_hss_max_heap_percent]

On each node, this setting determines the maximum heap size for the HyperStore Service application, as a percentage of host system RAM. The `JAVA_OPTS -Xmx` value passed to the JVM will be the **lower** of this percentage of host RAM based value and the absolute size value set by the "[hyperstore.heap.auto.limit](#)" (page 769) setting.

Default = 15

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set hyperstore.heap.auto.maxPercent=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

hyperstore.heap.maxSize

On each node, the memory heap maximum size setting (Java `Xmx` setting) for the HyperStore Service application.

If this is explicitly set, it overrides the upper limit derived from the `hyperstore.heap.auto.*` settings.

Default = not set

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set hyperstore.heap.maxSize=<string>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

hyperstore.heap.minSize

On each node, the memory heap minimum size setting (Java `Xms` setting) for the HyperStore Service application.

If this is explicitly set, it overrides the lower limit derived from the `hyperstore.heap.auto.*` settings.

Default = not set

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set hyperstore.heap.minSize=<string>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

hyperstore.logging.level

This setting controls the level at which HyperStore Service application logging will occur. Only messages with a severity level that matches or exceeds your configured logging level will be logged.

Listed in order from the least inclusive logging to the most inclusive logging, the supported logging levels are:

- OFF -- The OFF "level" turns off logging.
- FATAL -- The FATAL level designates very severe error events that will likely lead the application to abort.
- ERROR -- The ERROR level designates error events that may still allow the application to continue running.

- WARN -- The WARN level designates potentially harmful situations.
- INFO -- The INFO level designates informational messages that indicate the progress of the application at coarse-grained level.
- DEBUG -- The DEBUG level designates fine-grained informational events that are most useful to debug an application.
- TRACE -- The TRACE level designates finer-grained informational events than the DEBUG level.

For example, if this setting is set to WARN then only messages with severity levels WARN, ERROR, or FATAL will be logged by the HyperStore Service.

Default = WARN

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set hyperstore.logging.level=<string>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

hyperstore.lowRes.idleTimeout

Special, "low resource" maximum idle time to apply to HyperStore Service HTTP(S) connections when the number of simultaneous connections to a HyperStore Service node exceeds 1000. Configured in milliseconds. With this setting, you can have the HyperStore Service be less tolerant of connection idle time during times of high concurrent usage. (For a general description of idle timer behavior, see ["hyperstore.threads.idleTimeout"](#) (page 786).)

Default = 10000

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set hyperstore.lowRes.idleTimeout=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

hyperstore.messagebus.connections.max

[Former name in HyperStore 7.x and earlier: `messaging.service.maxconnections`]

On each node, the maximum simultaneous number of incoming connections that the HyperStore messaging service will accept.

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set hyperstore.messagebus.connections.max=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use `hsctl config set` to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

hyperstore.messagebus.connections.timeout

[Former name in HyperStore 7.x and earlier: *messaging.service.connection.timeout*]

Maximum time in seconds that the HyperStore Service on a node will allow for establishing a connection to the HyperStore Service on another node, for conducting inter-node operations.

Default = 300

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set hyperstore.messagebus.connections.timeout=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

hyperstore.repair.EC.queue.size

[Former name in HyperStore 7.x and earlier: *repair.session.queue.maxlength*]

During "[hsstool repair](#)" (page 918) operations, the target maximum number of object update jobs to queue for processing. Object update jobs are placed in queue by a differencer mechanism that detects discrepancies between an object metadata's on the local node versus the object's metadata on other nodes in the cluster.

Default = 2000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set hyperstore.repair.EC.queue.size=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

hyperstore.repair.merkleTreeRequest.timeout

[Former name in HyperStore 7.x and earlier: *repair.merkletree.response.waittime*]

During a [hsstool repair](#) operation, the repair coordinator node retrieves Merkle Trees from HyperStore endpoint nodes. When contacted by the coordinator node, the endpoint nodes first must construct the Merkle Trees before returning them to the coordinator node. Constructing the trees can take some time if a very large number of objects is involved.

The *hyperstore.repair.merkleTreeRequest.timeout* setting sets the maximum amount of time in minutes that the coordinator node will wait for Merkle Trees to be returned by all endpoint nodes. If not all Merkle Trees have been returned in this time, the repair operation will fail and stop.

Default = 360

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set hyperstore.repair.merkleTreeRequest.timeout=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

hyperstore.repair.objectQueue.size

[Former name in HyperStore 7.x and earlier: *repair.session.updateobjs.queue.maxlength*]

During an [hsstool repair](#) operation, the target maximum number of object update jobs to queue for processing. Object update jobs are placed in queue by a differencer mechanism that detects discrepancies between object metadata on remote replicas versus object metadata on the local node.

This target maximum may be exceeded in certain circumstances as indicated in the description of the "[hyperstore.repair.objectQueue.timeout](#)" (page 773) setting.

Default = 1000

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set hyperstore.repair.objectQueue.size=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

hyperstore.repair.objectQueue.timeout

[Former name in HyperStore 7.x and earlier: repair.session.updateobjs.queue.maxwaittime]

During an [hsstool repair](#) operation, the maximum total number of seconds for the differencer to wait for the object update job queue to fall below its target maximum size. After this interval, the differencer goes ahead and writes its current batch of update requests to the queue. In this scenario, the queue can grow beyond the target maximum size. The next time that the differencer has object update requests, it again checks the queue size, and if the queue size larger than the target maximum size, the wait time procedure starts over again.

Default = 120

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set hyperstore.repair.objectQueue.timeout=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

hyperstore.repair.proactive.interval

[Former name in HyperStore 7.x and earlier: hyperstore.proactiverepair.poll_time]

At this recurring interval each HyperStore node checks to see whether any proactive repair jobs are queued for itself, and executes those jobs if there are any. Configured as a number of minutes.

This check is also automatically performed when a HyperStore Service node starts up. Subsequently the recurring check occurs at this configured interval.

For more information about the proactive repair feature, see "[Automated Data Repair Feature Overview](#)" (page 130) .

Default = 60

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set hyperstore.repair.proactive.interval=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

Note If you want to trigger proactive repair on a particular node immediately, you can do so by running the `"hsstool proactiverepairq"` (page 899) command with the `"-start"` option.

Note For information about temporarily disabling the proactive repair feature, see ["Disabling or Stopping Data Repairs"](#) (page 134).

hyperstore.repair.proactive.retryQueue.timeout

[Former name in HyperStore 7.x and earlier: `hyperstore.proactiverepair.queue.max.time`]

When eventual consistency for writes is used in the system -- that is, if you have storage policies for which you have configured the write consistency level to be something less strict than ALL -- S3 PUTs of objects may succeed in the system even in circumstances when one or more write endpoints is unavailable. When this happens the system's proactive repair feature queues information about the failed endpoint writes, and automatically executes those writes later -- on an hourly interval (by default), without operator intervention. For more information about proactive repair see ["Proactive Repair"](#) (page 131).

The proactive repair feature's queueing mechanism entails writing metadata to the Metadata DB, and that metadata is subsequently deleted when the endpoint writes are executed by proactive repair. To avoid overburdening the Metadata DB with proactive queueing metadata, a limit is placed on how long the queueing can continue for an unavailable write endpoint node. The `hyperstore.repair.proactive.retryQueue.timeout` setting determines this time limit, in minutes.

If a node has been unavailable for more than `hyperstore.repair.proactive.retryQueue.timeout` minutes, the system stops writing to the proactive repair queue for that node. The next time the data center receives an incoming S3 PUT request for which the downed node is a write endpoint, an error is logged in the S3 application log, and an alert is generated in the CMC. As indicated by the alert, in this circumstance -- where the node has been down for more than `hyperstore.repair.proactive.retryQueue.timeout` minutes -- then after the node comes back online you first need to wait for proactive repair to complete on the node (you can monitor this in the CMC's **Repair Status** page) and then you **must manually initiate a full repair on the node** (see [hsstool repair](#) and [hsstool repairec](#)).

Note that once the node is back up, the `hyperstore.repair.proactive.retryQueue.timeout` timer is reset to 0. So if that node subsequently goes down again, proactive repair queueing would again occur for that node for up to `hyperstore.repair.proactive.retryQueue.timeout` minutes.

Default = 240

Note To disable this limit -- so that there is no limit on the time for which proactive repair queueing metadata can build up for a node that's unavailable -- set `hyperstore.repair.proactive.retryQueue.timeout` to 0. If you are considering doing so, first consult with Cloudian Support about the potential impact on the Metadata DB.

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set hyperstore.repair.proactive.retryQueue.timeout=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

hyperstore.repair.scheduled.cassandra.interval

For each HyperStore node, the interval (in minutes) between automatically running a full repair of the node's Cassandra data (Metadata DB) . It is essential to run a full Cassandra data repair on each node at an interval **less than 10 days** (less than the Cassandra `gc_grace_seconds` interval, which by default is 10 days). For each node the repair is for the node's primary token range only, and the system allows auto-repairs to run on multiple nodes concurrently.

Note that when a Cassandra repair operation is running on a target node, the scope of repair activity will extend to other nodes as well. In particular, repair of a target node will also make sure that for metadata row keys that fall within the target node's primary token range, the metadata's replicas also are present on the other nodes where they are supposed to be.

Default = 10080 (every seven days)

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set hyperstore.repair.scheduled.cassandra.interval=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

Note You also have the option of configuring this setting in the **Auto Repair Schedule Settings** section of the CMC's **"Configuration Settings"** (page 498) page. In the CMC UI this setting is called "Cassandra Full Repair Interval (Minutes)".

hyperstore.repair.scheduled.cassandra.range.enabled

[Former name in HyperStore 7.x and earlier: `enable.cassandra.rangerepair`]

If this is set to true, HyperStore uses a "range repair" approach when executing Cassandra (Metadata DB) auto-repairs. Each impacted token range is repaired one range at a time, sequentially. This approach improves the performance for Cassandra auto-repairs.

For background information on the scheduled auto-repair feature see **"Automated Data Repair Feature Overview"** (page 130).

Default = true

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set hyperstore.repair.scheduled.cassandra.range.enabled=<boolean>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

hyperstore.repair.scheduled.cassandra.timeout

[Former name in HyperStore 7.x and earlier: `cassandra.range_repair.max.waiting.time.in_sec`]

During a Cassandra auto-repair operation each of a node's token ranges are repaired one at a time, sequentially (if **"hyperstore.repair.scheduled.cassandra.range.enabled"** (page 775) is `true`, as it is by default). The system will wait for a maximum of this many seconds for repair of a range to complete. If a range repair times out by not being completed within this many seconds, the system moves on to repair the next range in sequence. Subsequently, after other ranges are repaired, repair of the range that timed out will be retried a maximum of 3 times. If it still cannot be repaired, the Cassandra repair as a whole will return a Failed status.

Default = 7200

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set hyperstore.repair.scheduled.cassandra.timeout=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

hyperstore.repair.scheduled.computeDigest.frequency

[Former name in HyperStore 7.x and earlier: *auto_repair_computedigest_run_number*]

This setting determines how often the [scheduled auto-repair feature](#) will use the "-computedigest" option in order to detect and repair any data corruption on disk ("bit rot"). Specifically, this setting sets an *N* such that every *N*th auto-repair run of [hssstool repair](#) (for replicated object data) and [hssstool repairrec](#) (for erasure coded object data) will use the "-computedigest" option.

For example, if *hyperstore.repair.scheduled.computeDigest.frequency* is set to 3, then on each node every 3rd run of *repair* will use the "-computedigest" option and for each data center every 3rd run of *repairrec* will use the "-computedigest" option.

To consider how this works in practice, note that by default the auto-repair interval for *repair* is 30 days, and each individual node has its own every-30-days repair schedule. So if for example you set *hyperstore.repair.scheduled.computeDigest.frequency* to 3, then on a given node the automatically triggered *repair* runs would be implemented like this:

- Day 0: *repair* without "-computedigest"
- Day 30: *repair* without "-computedigest"
- Day 60: *repair* with "-computedigest"
- Day 90: *repair* without "-computedigest"
- Day 120: *repair* without "-computedigest"
- Day 150: *repair* with "-computedigest"
- etc

By default the auto-repair interval for *repairrec* is 29 days. With erasure coded data repair, running *hssstool repairrec* on any one node repairs all the erasure coded data in the local data center. Consequently the auto-repair feature runs the command on just one randomly selected node in each data center every 29 days. So if for example you set *hyperstore.repair.scheduled.computeDigest.frequency* to 3, then for a given data center the automatically triggered *repairrec* runs would be implemented like this:

- Day 0: *repairrec* without "-computedigest"
- Day 29: *repairrec* without "-computedigest"
- Day 58: *repairrec* with "-computedigest"
- Day 87: *repairrec* without "-computedigest"
- Day 116: *repairrec* without "-computedigest"
- Day 145: *repairrec* with "-computedigest"
- etc

Setting *hyperstore.repair.scheduled.computeDigest.frequency* to 1 would result in all auto-repair runs using "-computedigest".

By default `hyperstore.repair.scheduled.computeDigest.frequency` is set to 0, which disables using "-computedigest" for auto-repair runs. **So by default no auto-repair runs will use the "-computedigest" option.**

Default = 0

Note Because it entails recalculating a fresh MD5 hash of each replica or erasure coded fragment on the target node, using "-computedigest" on repair runs is an expensive operation in terms of resource utilization, so this should be taken into account when you consider how often to use this option.

Note The `hyperstore.repair.scheduled.computeDigest.frequency` setting has no impact on `hsstool repair` or `hsstool repairec` runs that you manually execute on a node. The setting only impacts the auto-repair feature.

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set hyperstore.repair.scheduled.computeDigest.frequency=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

hyperstore.repair.scheduled.EC.interval

For each HyperStore node, the interval (in minutes) between scheduled auto-repairs of erasure coded object data. The auto-repair feature automatically executes the [hsstool repairec](#) operation to repair each node at the configured interval.

Note that for erasure coded object data repair, for single data center storage policies and for multi- data center replicated EC storage policies, repair of a target node has the effect of assessing and repairing all erasure coded data on all nodes within the data center where the target node resides. And for multi- data center distributed EC storage policies, repair of a target node has the effect of assessing and repairing all erasure coded data in all of the participating data centers.

Auto-repair of the erasure coded object data in your cluster is a resource-intensive operation and should be configured to run infrequently.

Default = 41760 (every 29 days)

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set hyperstore.repair.scheduled.EC.interval=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

Note You also have the option of configuring this setting in the **Auto Repair Schedule Settings** section of the CMC's **"Configuration Settings"** (page 498) page. In the CMC UI this setting is called "EC Repair Interval (Minutes)".

hyperstore.repair.scheduled.RF.interval

For each HyperStore node, the interval (in minutes) between scheduled auto-repairs of replicated object data. The auto-repair feature automatically executes the [hsstool repair](#) operation to repair each node at the

configured interval. For each node the repair is for the node's primary token range only, and the system allows auto-repairs to run on multiple nodes concurrently.

Note that when a replica repair operation is running on a target node, the scope of repair activity will extend to other nodes as well. In particular, repair of a target node will also make sure that for objects that fall within the target node's primary token range, the objects' replicas also are present on the other nodes where they are supposed to be.

Auto-repair of the replicated object data in your cluster is a resource-intensive operation and should be configured to run infrequently.

Default = 43200 (every 30 days)

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set hyperstore.repair.scheduled.RF.interval=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

Note You also have the option of configuring this setting in the **Auto Repair Schedule Settings** section of the CMC's **"Configuration Settings"** (page 498) page. In the CMC UI this setting is called "Replicas Repair Interval (Minutes)".

hyperstore.secureDelete

[Former name in HyperStore 7.x and earlier: *secure.delete*]

Set this to true if you want HyperStore to use **"secure delete"** methodology whenever implementing the deletion of an object from a bucket.

For information about how secure delete works, see **"Enabling Secure Delete"** (page 202)

IMPORTANT ! Using secure delete has substantial impact on **system performance for delete operations**. Consult with your Cloudbian representative if you are considering using secure delete.

Default = false

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set hyperstore.secureDelete=<boolean>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

hyperstore.stackSize

Memory stack size to use for the HyperStore Service.

If this setting is not explicitly set, the HyperStore Service uses the memory stack size specified by the **"java.stackSize"** (page 797) setting.

Default = not set (so uses *java.stackSize* value)

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hscctl config set hyperstore.stackSize=<string>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

hyperstore.storagePolicy.max

[Former name in HyperStore 7.x and earlier: *cloudian.protection.policy.max*]

Maximum number of storage policies that the system will support. Policies with status "Active", "Pending", or "Disabled" all count toward this system limit.

If the storage policies maximum has been reached, you will not be able to create new storage policies until you either delete existing policies or increase the value of *hyperstore.storagePolicy.max*.

For more information about storage policies, see ["Storage Policies Feature Overview" \(page 148\)](#).

Default = 25

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set hyperstore.storagePolicy.max=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

hyperstore.threadPool.autoRepair

[Former name in HyperStore 7.x and earlier: *hyperstore_auto_repair_threadpool*]

On each node, the maximum number of threads to allow in the thread pool used by the HyperStore auto-repair feature. When there is an auto-repair task to be performed and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set hyperstore.threadPool.autoRepair=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use *hscctl config set* to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

Note For more information about the auto-repair feature see ["Automated Data Repair Feature Overview" \(page 130\)](#).

hyperstore.threadPool.cleanup.jobs

[Former name in HyperStore 7.x and earlier: `cleanupjobs.threadpool.corepoolsizes`]

During [hsstool cleanup](#) or [hsstool cleanupec](#) operations, this setting controls how many cleanup "jobs" can run in parallel on a single HyperStore node. For each HyperStore data mount point on a node, the object data directory structure is as follows:

```
<mountpoint>/<hsfs|ec>/<base62-encoded-vNode-token>/<policyid>/<000-255>/<000-255>/<filename>
```

Under the *hsfs* (for replica data) or *ec* (for erasure coded data) directory level, there are sub-directories for each of the mount point's vNodes (identified by token), and under those, sub-directories for each storage policy configured in your system (identified by system-generated policy ID). For more information on this directory structure, see ["HyperStore Service and the HSFS" \(page 66\)](#).

When a physical HyperStore node is cleaned, there is a separate cleanup "job" for each `<policyid>` sub-directory on the physical node. The `hyperstore.threadPool.cleanup.jobs` setting controls how many such jobs can run in parallel on a given physical node. Each concurrent job will run on a different HyperStore data disk on the node.

Within each job, the separate setting ["hyperstore.threadPool.cleanup.session" \(page 780\)](#) controls how many blobs (object replicas or erasure coded fragments) can be processed in parallel. Processing a blob entails checking the blob's corresponding object metadata to determine whether the blob is supposed to be where it is or rather should be deleted.

So for example with `hyperstore.threadPool.cleanup.jobs = 10` and `hyperstore.threadPool.cleanup.session = 10`, then on a given physical node being cleaned a maximum of 10 cleanup jobs would run in parallel (with each job working on a different disk), with a maximum of 10 blobs being processed in parallel within each of the 10 jobs.

Default = 10

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set hyperstore.threadPool.cleanup.jobs=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

hyperstore.threadPool.cleanup.session

[Former name in HyperStore 7.x and earlier: `hyperstore_cleanup_session_threadpool`]

This thread pool size limit determines the maximum number of blobs (object replicas or erasure coded fragments) to process in parallel within each cleanup "job" taking place on a node. Processing a blob entails checking the blob's corresponding object metadata to determine whether the blob is supposed to be where it is or if it should be cleaned (deleted) from its location. When there is a new blob processing task to be performed and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:


```
# hscctl config set hyperstore.threadPool.cleanup.session=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use *hscctl config set* to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

hyperstore.threadPool.decommission

[Former name in HyperStore 7.x and earlier: *hyperstore_decommission_threadpool*]

Maximum number of threads to allow in of the thread pool used for migrating object data away from a node that is being decommissioned. When there is a new decommission task to be performed and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set hyperstore.threadPool.decommission=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use *hscctl config set* to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

hyperstore.threadPool.messaging

[Former name in HyperStore 7.x and earlier: *hyperstore_messaging_service_threadpool*]

On each node, the maximum number of threads to allow in the thread pool used by the HyperStore Service's inter-node messaging mechanism. When there is a new messaging task to be performed and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set hyperstore.threadPool.messaging=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use `hctl config set` to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

hyperstore.threadPool.rangeRepair

[Former name in HyperStore 7.x and earlier: `hyperstore_rangerepair_threadpool`]

On each node, the maximum number of threads to allow in the thread pool used for running multiple range repair tasks in parallel during [hsstool repair](#). When there is a new range repair task to be performed and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

To change this setting and apply your change, run the following `hctl` commands on the Config Controller node:

```
# hctl config set hyperstore.threadPool.rangeRepair=<integer>
# hctl config apply hyperstore
# hctl service restart hyperstore --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use `hctl config set` to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

hyperstore.threadPool.repair.digestIndex

[Former name in HyperStore 7.x and earlier: `hyperstore_repair_digest_index_threadpool`]

On each node, the maximum number of threads to allow in the thread pool used for reading file digests on a node in order to build the index required by Merkle Tree based repair (the default [hsstool repair](#) type). When there is a new digest reading task to be performed and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

To change this setting and apply your change, run the following `hctl` commands on the Config Controller node:

```
# hctl config set hyperstore.threadPool.repair.digestIndex=<integer>
# hctl config apply hyperstore
# hctl service restart hyperstore --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use `hctl config set` to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

hyperstore.threadPool.repair.session

[Former name in HyperStore 7.x and earlier: *hyperstore_repair_session_threadpool*]

On each node, the maximum number of threads to allow in the thread pool used for [hsstool repair](#) or [hsstool repairec](#) sessions (at a high level, repair work is organized into sessions). When there is a new session task to be performed and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set hyperstore.threadPool.repair.session=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use *hsctl config set* to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

hyperstore.threadPool.repairEC.digestRequest

[Former name in HyperStore 7.x and earlier: *hyperstore_repairec_digestrequest_threadpool*]

During an [hsstool repairec](#) operation, the threads in this thread pool are tasked with reading the digests associated with batches of erasure coded objects, to determine whether any of those objects need repair. This setting sets the maximum number of threads to allow in the thread pool per node. When there is a new digest reading task to be performed and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

Default = 30

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set hyperstore.threadPool.repairEC.digestRequest=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

hyperstore.threadPool.repairEC.rocksDBScan

[Former name in HyperStore 7.x and earlier: *hyperstore_repairec_rocksdbscan_threadpool*]

During an [hsstool repairec](#) operation, the threads in this thread pool enable concurrent reads of digest data in each [RocksDB](#) instance, as digest read requests come in from multiple digest request threads on multiple nodes. This parameter sets the maximum number of threads to allow in the thread pool per RocksDB instance.

When there is a new digest reading task to be performed and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

Default = 30

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set hyperstore.threadPool.repairEC.rocksDBScan=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

hyperstore.threadPool.repairEC.sessionScan

[Former name in HyperStore 7.x and earlier: `hyperstore_repairec_sessionscan_threadpool`]

During an [hsstool repairec](#) operation, the threads in this thread pool are tasked with identifying erasure coded objects in the system and batching them for evaluation. This parameter sets the maximum number of threads to allow in the thread pool per node. When there is a new scanning task to be performed and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

Default = 50

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set hyperstore.threadPool.repairEC.sessionScan=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

hyperstore.threadPool.repairEC.task

[Former name in HyperStore 7.x and earlier: `hyperstore_repairec_task_threadpool`]

During an [hsstool repairec](#) operation, the threads in this thread pool are tasked with repairing erasure coded objects that have been determined to need repair. This setting sets the maximum number of threads to allow in the thread pool per node. When there is a new repairing task to be performed and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

Default = 60

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set hyperstore.threadPool.repairEC.task=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

hyperstore.threadPool.session.rangeDownload

[Former name in HyperStore 7.x and earlier: *hyperstore_downloadrange_session_threadpool*]

During [hsstool repair](#), the maximum number of threads to allow in the thread pool used for range download sessions conducted by a node. When there is a new range download session task to perform and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set hyperstore.threadPool.session.rangeDownload=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use *hscctl config set* to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

hyperstore.threadPool.session.rangeUpload

[Former name in HyperStore 7.x and earlier: *hyperstore_uploadrange_session_threadpool*]

During [hsstool repair](#), the maximum number of threads to allow in the thread pool used for range upload sessions conducted by a node. When there is a new range upload session task to perform and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set hyperstore.threadPool.session.rangeUpload=<integer>
# hscctl config apply hyperstore
# hscctl service restart hyperstore --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use *hscctl config set* to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

hyperstore.threadPool.streamOutbound

[Former name in HyperStore 7.x and earlier: `hyperstore_stream_outbound_threadpool`]

Maximum number of threads to allow in the thread pool used for streaming files from one HyperStore node to another during Merkle Tree based [hsstool repair](#). When there is a new outbound streaming task to perform and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set hyperstore.threadPool.streamOutbound=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use `hsctl config set` to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

hyperstore.threads.idleTimeout

When the HyperStore Service processes HTTP(S) requests from clients, the maximum allowed connection idle time in milliseconds. If this much idle time passes before a new request is received over an open connection with a client, or if this much idle time passes during the reading of headers and content for a request, or if this much idle time passes during the writing of headers and content of a response, the connection with the client is closed.

Default = 30000

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set hyperstore.threads.idleTimeout=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

hyperstore.threads.max

[Former name in HyperStore 7.x and earlier: `hyperstore_jetty_maxThreads`]

On each node, the maximum number of threads to allow in the thread pool that the HyperStore Service uses to process incoming HTTP requests from clients. When there is a new HTTP request processing task to be performed and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set hyperstore.threads.max=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use `hsctl config set` to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

hyperstore.threads.maxRead

[Former name in HyperStore 7.x and earlier: `hyperstore.maxthreads.read`]

On each node, the maximum number of threads to allow in the thread pool that the HyperStore Service uses to process reads of the HyperStore File System. When there is a new read processing task to be performed and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set hyperstore.threads.maxRead=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use `hsctl config set` to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

hyperstore.threads.maxRepair

[Former name in HyperStore 7.x and earlier: `hyperstore.maxthreads.repair`]

On each node, the maximum number of threads to allow in the thread pool that the HyperStore Service uses to process HyperStore File System data repairs automatically performed during read operations ("repair on read"). When there is a new repair on read processing task to be performed and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

For more information on the "repair on read" mechanism see ["Automated Data Repair Feature Overview" \(page 130\)](#).

Default = 50

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set hyperstore.threads.maxRepair=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

hyperstore.threads.maxWrite

[Former name in HyperStore 7.x and earlier: *hyperstore.maxthreads.write*]

On each node, the maximum number of threads to allow in the thread pool that the HyperStore Service uses to process writes to the HyperStore File System. When there is a new write processing task to be performed and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set hyperstore.threads.maxWrite=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use *hsctl config set* to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

hyperstore.threads.min

On each node, the minimum number of threads to keep in the thread pool that the HyperStore Service uses to process incoming HTTP(S) requests from clients.

The Admin Service uses a thread pool to process incoming HTTP(S) requests from clients. An initial pool of threads is created at server initialization time, and additional threads may be added as needed (within the limit set by "[hyperstore.threads.max](#)" (page 786)). Idle threads in the pool are terminated if not used within a thread timeout period — unless the number of threads in the pool is down to *hyperstore.threads.min*. If only this many threads are in the pool, then threads are kept open even if they've been idle for longer than the thread timeout.

Default = 100

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set hyperstore.threads.min=<integer>
# hsctl config apply hyperstore
# hsctl service restart hyperstore --nodes=ALL
```


13.2.7. IAM Service Settings

The table below lists configuration settings in the *iam* setting group, which configures the [IAM Service](#). When applicable the settings in the table are organized into sub-groups.

Setting Sub-Group	Setting	Default Value
iam.heap	iam.heap.maxSize (page 790)	512m
	iam.heap.minSize (page 791)	100m
iam.logging	iam.logging.level (page 791)	INFO
	iam.logging.rollover.* settings	see " Log Rolling Settings " (page 689) for default values
iam.ports	iam.ports.loadBalancer.http (page 793)	16080
	iam.ports.loadBalancer.https (page 793)	16443
	iam.ports.main.http (page 793)	16080
	iam.ports.main.https (page 794)	16443
iam.threads	iam.threads.idleTimeout (page 794)	60000
	iam.threads.max (page 795)	150
	iam.threads.min (page 795)	100
iam.tls	iam.tls.enabled / iam.tls.required (page 795)	true / false
n/a (miscellaneous individual settings)	iam.user.maxGroups (page 796)	10
	iam.autoInvalidateCache (page 789)	true
	iam.dnsName (page 790)	iam.mycloudianhyperstore.com
	iam.enabled (page 790)	true
	iam.lowRes.idleTimeout (page 792)	5000
	iam.maxGroups (page 792)	300
	iam.mfa.totp.issuer (page 792)	Cloudian
	iam.stackSize (page 794)	

13.2.7.1. iam.* Setting Descriptions

iam.autoInvalidateCache

[Former name in HyperStore 7.x and earlier: *cloudian_s3_autoinvalidateiamcache*]

Each service region of your HyperStore system maintains its own cache of IAM user information, which includes IAM group membership and IAM policy-based permissions. In each service region, the S3 Service when processing S3 requests that require IAM information will try first to obtain the information from the local service region's IAM cache. If the information is not in the cache, the S3 Service retrieves the needed information from the main IAM information store in the default service region, uses this information to complete the S3 request processing, and stores the information into the IAM cache.

When `iam.autoInvalidateCache` is set to `true`, each service region's cache of IAM user information is purged once every six hours.

When `iam.autoInvalidateCache` is set to `false`, each region's cache of IAM user information is not automatically cleared out every six hours, and instead the cache data is only updated whenever there is a change to IAM user information. This is the preferable setting if you have a multi-region HyperStore system with high latency between regions. The `false` setting reduces the need for the HyperStore S3 Service in non-default regions to make calls to the default region to retrieve IAM information when IAM users submit S3 requests to the non-default regions.

Default = `true`

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set iam.autoInvalidateCache=<boolean>
# hsctl config apply iam
# hsctl service restart s3 iam --nodes=ALL
```

iam.dnsName

[Former name in HyperStore 7.x and earlier: `iam_service_endpoint`]

Service endpoint (fully qualified domain name) for the IAM Service. This endpoint must be resolvable for IAM clients.

Default = Derived from operator input during installation.

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set iam.dnsName=<string>
# hsctl config apply iam cmc
# hsctl service restart iam cmc --nodes=ALL
```

Note You also have the option of configuring this setting by using the installer's "Advanced Configuration Options", as described in ["Changing S3, Admin, CMC, or IAM/STS Service Endpoints"](#) (page 859).

iam.enabled

[Former name in HyperStore 7.x and earlier: `iam_service_enabled`]

This setting determines whether the IAM Service is enabled, `true` or `false`. This also determines whether the IAM Service will start automatically when a host reboots; and whether IAM functionality will display in the CMC.

Default = `true`

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set iam.enabled=<boolean>
# hsctl config apply iam cmc
# hsctl service restart iam cmc --nodes=ALL
```

iam.heap.maxSize

On each node, the memory heap maximum size setting (Java `Xmx` setting) for the IAM Service application.

Default = 512m

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set iam.heap.maxSize=<string>
# hsctl config apply iam
# hsctl service restart iam --nodes=ALL
```

iam.heap.minSize

On each node, the memory heap minimum size setting (Java *Xms* setting) for the IAM Service application.

Default = 100m

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set iam.heap.minSize=<string>
# hsctl config apply iam
# hsctl service restart iam --nodes=ALL
```

iam.logging.level

This setting controls the level at which IAM Service application logging will occur. Only messages with a severity level that matches or exceeds your configured logging level will be logged.

Listed in order from the least inclusive logging to the most inclusive logging, the supported logging levels are:

- OFF -- The OFF "level" turns off logging.
- FATAL -- The FATAL level designates very severe error events that will likely lead the application to abort.
- ERROR -- The ERROR level designates error events that may still allow the application to continue running.
- WARN -- The WARN level designates potentially harmful situations.
- INFO -- The INFO level designates informational messages that indicate the progress of the application at coarse-grained level.
- DEBUG -- The DEBUG level designates fine-grained informational events that are most useful to debug an application.
- TRACE -- The TRACE level designates finer-grained informational events than the DEBUG level.

For example, if this setting is set to WARN then only messages with severity levels WARN, ERROR, or FATAL will be logged by the IAM Service.

Default = INFO

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set iam.logging.level=<string>
# hsctl config apply iam
# hsctl service restart iam --nodes=ALL
```

iam.lowRes.idleTimeout

Special, "low resource" maximum idle time to apply to IAM Service HTTP(S) connections when the number of simultaneous connections to an IAM Service node exceeds 1000. Configured in milliseconds. With this setting, you can have the IAM Service be less tolerant of connection idle time during times of high concurrent usage. (For a general description of idle timer behavior, see ["iam.threads.idleTimeout"](#) (page 794).)

Default = 5000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set iam.lowRes.idleTimeout=<integer>
# hsctl config apply iam
# hsctl service restart iam --nodes=ALL
```

iam.maxGroups

[Former name in HyperStore 7.x and earlier: iam_max_groups]

The maximum number of IAM groups allowed per Clodian account root.

Default = 300

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

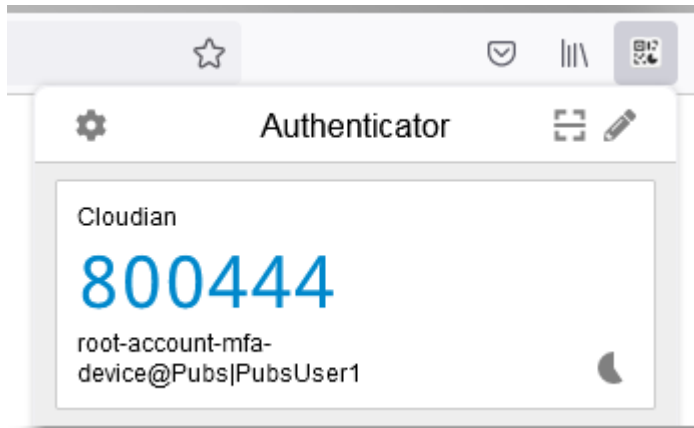
```
# hsctl config set iam.maxGroups=<integer>
# hsctl config apply iam
# hsctl service restart iam --nodes=ALL
```

Note The maximum number of **IAM users** per Clodian account root is 5000. This limit is not configurable.

iam.mfa.totp.issuer

[Former name in HyperStore 7.x and earlier: mfa_totp_issuer]

Users can enable multi-factor authentication (MFA) on their CMC login accounts (see ["Security Credentials"](#) (page 572)). The *iam.mfa.totp.issuer* setting specifies the MFA TOTP (time-based one-time password) Issuer name that will display in a user's virtual MFA device if the user enables MFA on their CMC account. The example below is from the virtual MFA device Google Authenticator, with the Issuer at its default value "Clodian".



Default = Cloudian

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set iam.mfa.totp.issuer=<string>
# hsctl config apply iam
# hsctl service restart iam s3 --nodes=ALL
```

iam.ports.loadBalancer.http

The IAM HTTP port on the load balancer.

Default = 16080

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set iam.ports.loadBalancer.http=<integer>
# hsctl config apply all
# hsctl service restart iam s3 cmc --nodes=ALL
```

iam.ports.loadBalancer.https

The IAM HTTPS port on the load balancer.

Default = 16443

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set iam.ports.loadBalancer.https=<integer>
# hsctl config apply all
# hsctl service restart iam cmc s3 --nodes=ALL
```

iam.ports.main.http

[Former name in HyperStore 7.x and earlier: iam_port]

On each node, the port on which the IAM Service listens for regular HTTP connections.

Default = 16080

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set iam.ports.main.http=<integer>
# hsctl config apply iam all
# hsctl service restart iam firewall cmc s3 --nodes=ALL
```

iam.ports.main.https

[Former name in HyperStore 7.x and earlier: *iam_secure_port*]

On each node, the port on which the IAM Service listens for HTTPS connections.

Default = 16443

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set iam.ports.main.https=<integer>
# hsctl config apply all
# hsctl service restart iam firewall cmc s3 --nodes=ALL
```

Note For information about enabling/disabling the local HTTPS listener for the IAM Service on each node, see "[iam.tls.enabled / iam.tls.required](#)" (page 795).

iam.stackSize

Memory stack size to use for the IAM Service.

If this setting is not explicitly set, the IAM Service uses the memory stack size specified by the "[java.stackSize](#)" (page 797) setting.

Default = not set (so uses *java.stackSize* value)

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set iam.stackSize=<string>
# hsctl config apply iam
# hsctl service restart iam --nodes=ALL
```

iam.threads.idleTimeout

When the IAM Service processes HTTP(S) requests from clients, the maximum allowed connection idle time in milliseconds. If this much idle time passes before a new request is received over an open connection with a client, or if this much idle time passes during the reading of headers and content for a request, or if this much idle time passes during the writing of headers and content of a response, the connection with the client is closed.

Default = 60000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set iam.threads.idleTimeout=<integer>
# hsctl config apply iam
# hsctl service restart iam --nodes=ALL
```

iam.threads.max

On each node, the maximum number of threads to allow in the thread pool that the IAM Service uses to process incoming HTTP(S) requests from clients. When there is a new HTTP(S) request processing task and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

Default = 150

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set iam.threads.max=<integer>
# hsctl config apply iam
# hsctl service restart iam --nodes=ALL
```

iam.threads.min

On each node, the minimum number of threads to keep in the thread pool that the IAM Service uses to process incoming HTTP(S) requests from clients.

The IAM Service uses a thread pool to process incoming HTTP(S) requests from clients. An initial pool of threads is created at server initialization time, and additional threads may be added as needed (within the limit set by "[iam.threads.max](#)" (page 795)). Idle threads in the pool are terminated if not used within a thread timeout period — unless the number of threads in the pool is down to *iam.threads.min*. If only this many threads are in the pool, then threads are kept open even if they've been idle for longer than the thread timeout.

Default = 100

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set iam.threads.min=<integer>
# hsctl config apply iam
# hsctl service restart iam --nodes=ALL
```

iam.tls.enabled / iam.tls.required

[Former name of *iam.tls.required* in HyperStore 7.x and earlier: *iam_secure*]

The ***iam.tls.enabled*** setting controls whether a local HTTPS listener is enabled for the IAM Service on each node. If enabled (*true*), the local HTTPS listener will be on the port number specified by the "[iam.ports.main.https](#)" (page 794) setting. If disabled (*false*), then on each node locally only the regular HTTP listener is opened for the IAM Service -- on the port specified by the "[iam.ports.main.http](#)" (page 793) setting.

Default = true

The ***iam.tls.required*** setting controls whether clients -- including other HyperStore services such as the CMC -- will be required to use HTTPS when connecting to the IAM Service. Other HyperStore services read this setting to determine whether they must use HTTPS to connect to the IAM Service. Since a load balancer is typically used in front of the IAM Service, these services if required to use HTTPS for IAM Service access will connect to the load balancer port specified by the "[iam.ports.loadBalancer.https](#)" (page 793) setting.

Default = false

Note that:

- With the default configuration -- *iam.tls.enabled* set to *true* and *iam.tls.required* set to *false* -- on each node the IAM Service's local HTTP listener and HTTPS listener will both be open, and other HyperStore services will access the IAM Service using HTTP not HTTPS.
- If you want to use TLS offloading for the IAM Service -- whereby IAM clients use TLS when connecting to a load balancer but the load balancer does not use TLS when connecting to the IAM Service on individual nodes -- set *iam.tls.enabled* to *false* (so that on each node the IAM Service locally opens only a regular HTTP listener) and *iam.tls.required* to *true* (so that other HyperStore services use HTTPS to access the IAM Service via the load balancer).
- For information about TLS certificate management, see ["Security and Privacy Features" \(page 177\)](#).

To change the *iam.tls.enabled* setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set iam.tls.enabled=<boolean>
# hsctl config apply iam
# hsctl service restart iam --nodes=ALL
```

To change the *iam.tls.required* setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set iam.tls.required=<boolean>
# hsctl config apply all
# hsctl service restart iam s3 cmc --nodes=ALL
```

To change the *iam.tls.enabled* setting and the *iam.tls.required* setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set iam.tls.enabled=<boolean>
# hsctl config set iam.tls.required=<boolean>
# hsctl config apply all
# hsctl service restart iam s3 cmc --nodes=ALL
```

iam.user.maxGroups

[Former name in HyperStore 7.x and earlier: *iam_max_groups_per_user*]

The maximum number of IAM groups that an IAM user can belong to simultaneously. If an IAM user belongs to this many groups, the IAM Service will not allow her to be joined to any more groups.

IMPORTANT ! The more IAM groups an IAM user belongs to, the more complex and time-consuming will be the operation to assess the various IAM policies that apply to the user when the user submits an S3 request (to determine whether the user has permission for that request). Therefore, exercise caution in raising the value of this setting.

Default = 10

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set iam.user.maxGroups=<integer>
```


13.2.8. Java Settings

The table below lists configuration settings in the *java* setting group, which configures the JVM for Java-based services.

Setting	Default Value
java.stackSize (page 797)	256k

13.2.8.1. java.* Setting Descriptions

java.stackSize

[Former name in HyperStore 7.x and earlier: *java_minimum_stack_size*]

Memory stack size to use for the HyperStore system's Java-based services (Cassandra, HyperStore Service, S3 Service, Admin Service, IAM Service, SQS Service).

Optionally this setting can be overridden on a per-service basis, by the *<service>.stackSize* settings (such as ["admin.stackSize" \(page 706\)](#)).

Default = 256k

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set java.stackSize=<string>
# hsctl config apply ALL
# hsctl service restart s3 hyperstore cassandra --nodes=ALL
```

13.2.9. Jobs Settings

The table below lists configuration settings in the *jobs* setting group, which configures system cron jobs.

Setting	Default Value
jobs.system.<jobName>.enabled (page 797)	specific to job
jobs.system.<jobName>.schedule (page 798)	specific to job

Note The settings *jobs.system.<jobName>.name* and *jobs.system.<jobName>.description* can be retrieved but are not editable.

13.2.9.1. jobs.* Setting Descriptions

jobs.system.<jobName>.enabled

This setting enables or disables the HyperStore system cron job identified by *<jobName>*:

- If set to *true* the cron job is enabled
- If set to *false* the cron job is disabled

For a list of configurable cron jobs and their job names see ["System cron Jobs" \(page 632\)](#).

Note All cron jobs are enabled by default with the exception of the *onDemandLogCollector* job and the *kmipRekey* job which are disabled by default.

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set jobs.system.<jobName>.enabled=<value>
# hsctl config apply jobs
```

No service restart is required.

jobs.system.<jobName>.schedule

This setting configures the scheduling of the HyperStore system cron job identified by *<jobName>*. For a list of configurable cron jobs and their job names see ["System cron Jobs" \(page 632\)](#).

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set jobs.system.<jobName>.schedule=<value>
# hsctl config apply jobs
```

The *<value>* is specified in standard crontab schedule formatting, enclosed in quotes.

For example:

```
# hsctl config set jobs.system.bucketLifecycle.schedule="0 4 * * *"
# hsctl config apply jobs
```

No service restart is required.

13.2.10. LB (hsDNS) Settings

The table below lists configuration settings in the *lb* setting group, which configures an *hsDNS* component that optionally can run on each HyperStore node to provide domain resolution and DNS-based load balancing for a HyperStore system.

Setting	Default Value
lb.hsDNS.enabled (page 799)	false
lb.hsDNS.healthcheck.probe.consult.addrs (page 799)	empty
lb.hsDNS.healthcheck.probe.consult.policy (page 799)	QUORUM
lb.hsDNS.healthcheck.probe.interval (page 800)	5
lb.hsDNS.healthcheck.probe.type (page 800)	HTTP
lb.hsDNS.locationAffinity (page 800)	true
lb.hsDNS.public (page 801)	false
lb.hsDNS.publishBackend (page 802)	false
lb.hsDNS.ttl (page 802)	5

13.2.10.1. lb.* Setting Descriptions

lb.hsDNS.enabled

hsDNS is a HyperStore component that can act as an authoritative nameserver residing on all HyperStore nodes and providing DNS based load balancing and service discovery for all HyperStore services. By default hsDNS is disabled (*lb.hsDNS.enabled* is set to *false*). To enable hsDNS set *lb.hsDNS.enabled* to *true*.

For more information see:

- ["hsDNS Feature Overview" \(page 139\)](#)
- ["Setting Up hsDNS" \(page 142\)](#)

Default = false

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set lb.hsDNS.enabled=<boolean>
# hscctl config apply lb
```

No service restart is required.

lb.hsDNS.healthcheck.probe.consult.addrs

You can use this setting if you want hsDNS to send an ICMP packet to one or more specified IP addresses and to require an ICMP reply in order to consider a service to be up and healthy. If you enable this type of health check, it will be **in addition** to the health check specified by ["lb.hsDNS.healthcheck.probe.type" \(page 800\)](#).

This additional type of health check can cover scenarios such as an interface, switch port, or gateway going down. A typical use would be to specify the gateway address over which client requests are received. If the node doesn't receive a reply from the gateway address, the node will be regarded as down and all services running on the node will be removed from hsDNS.

Note Do not specify the IP addresses of HyperStore nodes with this setting.

Default = empty

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set lb.hsDNS.healthcheck.probe.consult.addrs=<list>
# hscctl config apply lb
```

No service restart is required.

lb.hsDNS.healthcheck.probe.consult.policy

If you specify multiple addresses with the ["lb.hsDNS.healthcheck.probe.consult.addrs" \(page 799\)](#) setting, then the *lb.hsDNS.healthcheck.probe.consult.policy* settings determines how many of those addresses must return an ICMP reply before the actual health checks are evaluated. Options are ANY, QUORUM, ALL.

Default = QUORUM

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set lb.hsDNS.healthcheck.probe.consult.policy=<string>
# hscctl config apply lb
```

No service restart is required.

lb.hsDNS.healthcheck.probe.interval

The interval at which the hsDNS instance on each HyperStore node checks the health of the HyperStore service instances on that node, in seconds.

For more information about the health check see ["lb.hsDNS.healthcheck.probe.type" \(page 800\)](#).

Minimum value is 5, maximum value is 300.

Default = 5

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set lb.hsDNS.healthcheck.probe.interval=<integer>
# hscctl config apply lb
```

No service restart is required.

lb.hsDNS.healthcheck.probe.type

The type of health check that hsDNS will perform for each HyperStore service endpoint on each node. The hsDNS instance on each HyperStore node will perform this type of check for all the service instances on that node (S3 Service, Admin Service, IAM Service, and so on) and a service down event is shared with the other hsDNS instances. This configuration parameter must be set to one (and only one) of the following options:

- HTTP -- The health check is conducted as a lightweight HTTP request, using the node's default/public IP address. Only an HTTP status response of 200 will satisfy the check.
- process -- This approach checks the service process status as reported by *systemd*.

The 'HTTP' method provides a more complete check of the service instance availability.

Default = HTTP

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set lb.hsDNS.healthcheck.probe.type=<string>
# hscctl config apply lb
```

No service restart is required.

Note The frequency of the health check is configurable by the ["lb.hsDNS.healthcheck.probe.interval" \(page 800\)](#) setting. The default is 5 seconds.

lb.hsDNS.locationAffinity

The *lb.hsDNS.locationAffinity* setting controls whether hsDNS will use location affinity when publishing A records for HyperStore service endpoints.

If *lb.hsDNS.locationAffinity* is set to *true*, hsDNS instances within a given DC will only publish the IP addresses of service instances within the local DC. Exceptions are:

- Requests that must be resolved to a service region other than the local service region -- for example, Admin Service requests (which in most cases must be fulfilled by nodes in the default service region). In this case hsDNS will publish the IP addresses of nodes in the appropriate service region even if it is not the local region.
- When all instances of the service in the local DC are unavailable. In this case hsDNS will publish the IP addresses of healthy service instances in the remaining DCs. When one or more local service instances become available again, then those local instances' IP addresses will be published again rather than those in a different DC.

If *lb.hsDNS.locationAffinity* is set to *false*, hsDNS will not favor nodes within the local DC when publishing the node IP addresses corresponding to service endpoints. Instead, all healthy instances' IP addresses will be published regardless of location.

Default = true

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set lb.hsDNS.locationAffinity=<boolean>
# hscctl config apply lb
```

No service restart is required.

lb.hsDNS.public

The *lb.hsDNS.public* setting controls whether or not hsDNS will be reachable from outside of the internal (backend) network to which HyperStore is connected. Options are:

- *false* -- On each HyperStore node, hsDNS will bind only on *local/host* and on the backend network to which the HyperStore nodes are connected (listening on port 53, the standard DNS listening port). Also, the HyperStore Firewall will block external traffic to port 53. In this mode of operation, hsDNS is intended to provide DNS based service discovery, domain resolution, and load balancing only for service requests originating within the HyperStore cluster itself.
- *true* -- On each HyperStore node, hsDNS will listen (on port 53) on all interfaces including the default/public interface. Also, the HyperStore Firewall will allow all traffic to port 53. Setting *lb.hsDNS.public* to *true* facilitates DNS delegation, so that external requests (such as from S3 user applications) can be forwarded into hsDNS by your existing DNS system, and hsDNS can return the IP addresses of the closest healthy service instances, in round robin rotation -- sometimes called Global Server Load Balancing (GSLB). If you set *lb.hsDNS.public* to *true* be sure to also configure DNS delegation in your DNS system (see ["Setting Up DNS Delegation"](#) (page 802) below for general guidance).

Note If you allow external access to hsDNS through DNS delegation, hsDNS will not support DNS recursion for incoming external requests (for example it will not help to resolve an external request for *www.google.com*). By contrast, hsDNS will support DNS recursion for requests originating from the HyperStore nodes themselves (such as for accessing external NTP servers, or auto-tiering to an external S3 system, or sending Cloudbian Smart Support bundles).

Default = false

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hctl config set lb.hsDNS.public=<boolean>
# hctl config apply lb firewall
```

No service restart is required.

Setting Up DNS Delegation

If you set *lb.hsDNS.public* to *true* you should also configure DNS delegation in your external nameserver. With DNS delegation, a nameserver (in this case your nameserver that's outside your HyperStore cluster) is configured to delegate to a different nameserver (in this case hsDNS in your HyperStore cluster) responsibility for resolving a specified subdomain.

Suppose your HyperStore S3 Service has a service endpoint of *s3-region1.storage.enterprise.com*. In your external name server's configuration for the *enterprise.com* domain you would configure DNS delegation for the *storage* subdomain, by adding:

- A "glue" record (a type of A record that associates a nameserver with its IP address) for each hsDNS nameserver host that you want to include in the delegation; for example using three hsDNS nameserver hosts:

```
cloudian1.storage IN A 10.15.40.101
cloudian2.storage IN A 10.15.40.102
cloudian3.storage IN A 10.15.40.103
```

- NS records associating the delegated subdomain with each hsDNS nameserver to which the subdomain is delegated:

```
storage IN NS cloudian1.storage.enterprise.com
storage IN NS cloudian2.storage.enterprise.com
storage IN NS cloudian3.storage.enterprise.com
```

For more detail about DNS delegation, refer to any reputable online source about the subject.

Note For resiliency and load sharing you should specify multiple hsDNS nameservers in your DNS delegation configuration.

lb.hsDNS.publishBackend

The *lb.hsDNS.publishBackend* setting controls whether hsDNS will publish the back-end IP addresses of HyperStore nodes rather than the default/public IP addresses. Options are *true* (publish back-end IP addresses only) or *false* (publish default/public IP addresses only).

Default = false

To change this setting and apply your change, run the following *hctl* commands on the Config Controller node:

```
# hctl config set lb.hsDNS.publishBackend=<boolean>
# hctl config apply lb
```

No service restart is required.

lb.hsDNS.ttl

For A records (which map domains to IP addresses) published in hsDNS, the time to live (TTL) in number of seconds.

Minimum value is 1, maximum value is 3600.

Default = 5

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set lb.hsDNS.ttl=<integer>
# hscctl config apply lb
```

No service restart is required.

13.2.11. Logging Settings

The table below lists configuration settings in the *logging* setting group, which configures certain aspects of HyperStore logging.

Setting	Default Value
logging.syslog.remote.* (page 803)	see settings description for defaults

13.2.11.1. logging.* Setting Descriptions

logging.syslog.remote.*

logging.syslog.remote.* setting	Default Value
logging.syslog.remote.enabled	false
logging.syslog.remote.host	empty
logging.syslog.remote.port	514
logging.syslog.remote.protocol	UDP

If you wish you can have log messages from HyperStore's S3 Service, Admin Service, IAM Service, SQS Service, HyperStore Service, Firewall Service, and HyperStore Shell forwarded to a central logging server using *rsyslog*. This is not enabled by default, but you can enable it as described below. This procedure sets up your HyperStore logging so that log content is transmitted to a central logging server **in addition to** being written on each HyperStore node locally.

The procedure presumes that:

- You have a central logging server running *rsyslog*.
- Each of your HyperStore nodes has *rsyslog* installed. Note that *rsyslog* is included in HyperStore Appliances and also in most RockyLinux and RHEL distributions.

It is also assumed that you are managing and configuring the central *rsyslog* server in accord with your policies and preferences. Configuration of the central logging server is not covered here. The procedure below only describes how to have HyperStore forward its log messages to the central server.

IMPORTANT ! The central logging server **must not be one of your HyperStore nodes**.

To have HyperStore write application log and request log messages to a central logging server, on your HyperStore Config Controller node run the following *hscctl* commands:

```
# hscctl config set logging.syslog.remote.enabled=true
# hscctl config set logging.syslog.remote.host=<hostname or IP address of log server>
# hscctl config set logging.syslog.remote.protocol=<UDP or TCP>
```

```
# hscctl config set logging.syslog.remote.port=<log server listening port; 514 is typical>
# hscctl config apply ALL
# hscctl service restart s3 iam sqs hyperstore firewall --nodes=ALL
```

13.2.12. Monitoring and Notification Settings

The table below lists configuration settings in the *monitoring* setting group, which configures HyperStore system monitoring and notifications.

Setting	Default Value
monitoring.agent.logging.level (page 804)	WARN
monitoring.agent.logging.* rollover settings	see settings description for default values
monitoring.alerts.active (page 805)	see setting description for default value
monitoring.alerts.suppressHIQDuplicates (page 806)	false
monitoring.dataCollector.logging.level (page 807)	WARN
monitoring.event.acknowledgedTTI (page 807)	86400
monitoring.notifications.smtp.* (page 808)	see settings description for default values
monitoring.snmp.traps.receiver.* (page 809)	see settings description for default values

13.2.12.1. monitoring.* Setting Descriptions

monitoring.agent.logging.level

This setting controls the level at which the monitoring Agent's application logging will occur. Only messages with a severity level that matches or exceeds your configured logging level will be logged.

Listed in order from the least inclusive logging to the most inclusive logging, the supported logging levels are:

- OFF -- The OFF "level" turns off logging.
- FATAL -- The FATAL level designates very severe error events that will likely lead the application to abort.
- ERROR -- The ERROR level designates error events that may still allow the application to continue running.
- WARN -- The WARN level designates potentially harmful situations.
- INFO -- The INFO level designates informational messages that indicate the progress of the application at coarse-grained level.
- DEBUG -- The DEBUG level designates fine-grained informational events that are most useful to debug an application.
- TRACE -- The TRACE level designates finer-grained informational events than the DEBUG level.

For example, if this setting is set to WARN then only messages with severity levels WARN, ERROR, or FATAL will be logged by the monitoring Agent.

Default = WARN

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set monitoring.agent.logging.level=<string>
# hscctl config apply monitoring
```


No service restart is required.

monitoring.alerts.active

[Former name in HyperStore 7.x and earlier: *alert_filter.yaml* file]

For the Admin Service, HyperStore Service, Redis Monitor Service, S3 Service, and Phone Home Service, application log error messages and warning messages will trigger alerts only if **both** of the following are true:

- Alerts for that log message severity, for that service, are enabled in the CMC **Alert Rules** page. (By default, alerts are enabled for Error messages for each of these services, but not for Warning messages. You can modify this configuration in the **Alert Rules** page if you wish -- for example you could enable alerts for Warning messages from the S3 Service.)
- The message code for the particular log message matches against one of the message codes specified by the *monitoring.alerts.active* setting.

All warning and error messages for the Admin, HyperStore, Redis Monitor, S3, and Phone Home services are assigned alphanumeric message codes that help to identify the message (for more information on this feature see [Log Message Codes](#)). There are many hundreds of such messages and corresponding message codes. But only messages with a message code that is listed by the *monitoring.alerts.active* setting can trigger alerts. This setting is a filtering mechanism to reduce the number of log messages that can result in alerts.

By default the message codes listed by the *monitoring.alerts.active* setting are for Error level log messages for which an operator action is prescribed. If one of these log messages appears in a service application log, an alert is generated (presuming that alerts for Error level log messages are enabled in the CMC **Alert Rules** page, as they are by default). The prescribed operator action can then be seen in the CMC **Alerts** page (by holding your cursor over the message code in the alert) or in the CMC Help (by entering the message code in the Search box).

Note The *monitoring.alerts.active* setting does not apply the log messages generated by the Cassandra Service or the Redis Credentials DB or Redis QoS DB services, since these log messages do not have message codes.

Default =

DC040098,DC040103,HS060006,HS060011,HS060019,HS065001,HS065002,HS065003,HS065004,HS080-033,HS085002,
HS110008,HS110009,HS110012,HS110013,HS110016,HS112034,HS120115,HS120232,HS120233,HS160-002,HS160019,HS160023,
HS170006,HS170012,HS170015,HS172001,HS172002,HS172003,HS180110,HS180216,HS180218,HS180-255,HS180285,HS180323,
HS180331,HS181005,HS181006,HS182002,HS183004,HS183023,HS183045,HS188018,HS188019,HS188-032,HS200012,HS200043,
HS200061,HS200062,HS200063,HS200064,HS200065,HS200066,HS233014,HS233017,HS233024,HS233-059,HS233068,HS240018,
HS240051,HS240052,HS240058,HS240060,HS240067,HS240069,HS240076,HS240108,HS240113,HS250-001,HS250002,HS250003

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node.

```
# hstctl config set monitoring.alerts.active=<list>
# hstctl config apply monitoring
```

No service restart is required.

IMPORTANT ! Your value for this setting must be a quote-enclosed comma-separated list of **all** of the message codes for which you want to allow alerting. Since your specified value will **replace** the existing list, you cannot specify only codes that you want to add to the existing list -- you must specify the full list. You can do this by first running `hstctl config get monitoring.alerts.active` and copying the existing list, and then editing the list, and pasting in the edited list as the value when running the `config set` command.

monitoring.alerts.suppressHIQDuplicates

[Former name in HyperStore 7.x and earlier: `suppress_duplicate_hiq_alert`]

This setting is applicable only if you have [Cloudian HyperIQ](#).

The CMC and HyperIQ are both capable of generating alerts based on customizable alert rules (for more about this CMC capability see "[Alert Rules](#)" (page 560); for more about this HyperIQ capability see the [Cloudian HyperIQ User's Guide](#)). There is some overlap in the alerts that the CMC and HyperIQ cover. If you set `monitoring.alerts.suppressHIQDuplicates` to `true`, then the CMC will not notify you about alerts that HyperIQ notifies you about. With this setting enabled, alerts that are covered by HyperIQ will not display in the CMC's **Alerts** page and the CMC will not send emails or SNMP traps in regard to such alerts. This keeps you from getting duplicate notifications from the CMC and HyperIQ about the same system events or conditions.

This setting does not apply to alerts triggered by the appearance of warnings or errors in HyperStore application logs, since HyperIQ does not currently monitor for this type of alert. Only the CMC monitors and notifies you about this type of alert.

The specific types of alerts that will be suppressed in the CMC if you set `monitoring.alerts.suppressHIQDuplicates` to `true` are listed below:

- Number of GET transactions per second
- Number of PUT transactions per second
- Throughput for GET operations
- Throughput for PUT operations
- Network throughput (incoming)
- Network throughput (outgoing)
- Disk space available in each device
- Disk error
- CPU utilization
- Admin Service status
- Cassandra Service status
- HyperStore Service status
- Redis QoS Service status
- Redis Credentials Service status
- Redis Monitor Service status
- S3 Service status

Default = false (the duplicate suppression function is not turned on)

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set monitoring.alerts.suppressHIQDuplicates=<boolean>
# hscctl config apply monitoring
```

No service restart is required.

Note You also have the option of configuring this setting in the **Alert Settings** section of the CMC's **"Configuration Settings"** (page 498) page. In the CMC UI this setting is called "Suppress Alerts Duplicated by HyperIQ".

monitoring.dataCollector.logging.level

This setting controls the level at which the monitoring Data Collector's application logging will occur. Only messages with a severity level that matches or exceeds your configured logging level will be logged.

Listed in order from the least inclusive logging to the most inclusive logging, the supported logging levels are:

- OFF -- The OFF "level" turns off logging.
- FATAL -- The FATAL level designates very severe error events that will likely lead the application to abort.
- ERROR -- The ERROR level designates error events that may still allow the application to continue running.
- WARN -- The WARN level designates potentially harmful situations.
- INFO -- The INFO level designates informational messages that indicate the progress of the application at coarse-grained level.
- DEBUG -- The DEBUG level designates fine-grained informational events that are most useful to debug an application.
- TRACE -- The TRACE level designates finer-grained informational events than the DEBUG level.

For example, if this setting is set to WARN then only messages with severity levels WARN, ERROR, or FATAL will be logged by the monitoring Data Collector.

Default = WARN

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set monitoring.dataCollector.logging.level=<string>
# hscctl config apply monitoring
```

No service restart is required.

monitoring.event.acknowledgedTTI

[Former name in HyperStore 7.x and earlier: events.acknowledged.ttl]

Time-to-live for acknowledged system events in the Monitoring keyspace in Cassandra, in seconds. Acknowledged events will be automatically deleted this many seconds after an administrator acknowledges them.

Unacknowledged events are not subject to this timer. The time-to-live countdown on an event record does not begin until an administrator acknowledges the event through the CMC's [Alerts](#) page or [Node Status](#) page.

If you want alerts to be deleted immediately after they have been acknowledged, set this property to 1.

Maximum allowed value is 604800 (seven days).

Note As soon an alert occurs a record of it is written to the Smart Support log (which by default is uploaded to Cloudian Support once a day). So the Smart Support record of alerts persists even after the alerts have been deleted from your system. For more information on Smart Support see "[Smart Support and Diagnostics Feature Overview](#)" (page 117).

Default = 86400 (one day)

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set monitoring.event.acknowledgedTtl=<integer>
# hscctl config apply monitoring s3
# hscctl service restart s3 --nodes=ALL
```

No service restart is required.

monitoring.notifications.smtp.*

The *monitoring.notifications.smtp.** settings pertain to the SMTP server that you want the HyperStore system to use when it sends alert notification emails to system administrators. Providing the system with this information is essential for proper system monitoring and administration. To use this feature you will need information about your organization's mail server, and at least one system administrator email account must have already been set up with the mail server and be able to receive email.

monitoring.notifications.smtp.* setting	Default Value
monitoring.notifications.smtp.auth.password	empty
monitoring.notifications.smtp.auth.required	false
monitoring.notifications.smtp.auth.username	empty
monitoring.notifications.smtp.defaultToAddress	admin@smtp.notification.configure.me
monitoring.notifications.smtp.fromAddress	noreply@smtp.notification.configure.me
monitoring.notifications.smtp.host	smtp.notification.configure.me
monitoring.notifications.smtp.port	465
monitoring.notifications.smtp.protocol	smtps
monitoring.notifications.smtp.startTLS	false
monitoring.notifications.smtp.subjectHeader	Cloudian HyperStore Alert

The individual *monitoring.notifications.smtp.** settings are:

- *auth.password* -- If SMTP Authentication is required by the SMTP server, the password to submit with requests to the server. Comma (,) and double-quote (") are not supported in this setting's value.
- *auth.required* -- Whether the SMTP server that the HyperStore system will use requires clients to use SMTP Authentication. Options are *true* or *false*.

- *auth.username* -- If SMTP Authentication is required by the SMTP server, the username to submit with requests to the server. Comma (,) and double-quote (") are not supported in this setting's value.
- *defaultToAddress* -- Default system administrator email address(es) to which to send alert notification emails. If you want alert notification emails to go to multiple email addresses, enter the addresses as a comma-separated list. Note that this default recipient address can optionally be overridden for specific alerts when you configure alert rules in the CMC's **Alert Rules** page.
- *fromAddress* -- The "From" address to use when sending alert notification emails.
- *host* -- Fully qualified domain name (FQDN) of the SMTP server that the HyperStore system should utilize for sending alert notification emails to system administrators.
- *port* -- Listening port of the SMTP server that the HyperStore system should utilize for sending alert notification emails to system administrators.
- *protocol* -- Protocol to use when sending alert notification emails. Options are *smtp* or *smtps*.
- *startTLS* -- Whether to use STARTTLS when sending alert notification emails. Options are *true* or *false*. If you enable this option you should also set the SMTP Protocol setting to "smtp" (not "smtps") and set the SMTP Port setting to 587 (if your email server is using the typical STARTTLS port). Note that even if you enable STARTTLS, if the recipient email account is a Gmail account the account user will still need to set the 'Allow less secure apps' option on the Gmail dashboard.
- *subjectHeader* -- The "Subject" to use when sending alert notification emails.

To change any of the settings listed in the table above and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set <setting>=<value>
# hsctl config apply monitoring
```

No service restart is required.

For example:

```
# hsctl config set monitoring.notifications.smtp.subjectHeader="Object Storage System Alert"
# hsctl config apply monitoring
```

Note You also have the option of configuring these settings in the **SMTP/Email Settings for Alerts/Notifications** section of the CMC's **"Configuration Settings"** (page 498) page.

monitoring.snmp.traps.receiver.*

The *monitoring.snmp.traps.receiver.** settings configure an SNMP trap destination, in support of having HyperStore system alerts sent as SNMP traps.

Note For information about enabling SNMP trap sending for individual alert types, see **"Alert Rules"** (page 560). For traps to be sent, you must configure the SNMP trap destination settings and then also enable trap sending for individual alert types.

In the traps that HyperStore generates and sends to your specified destination, the OID is *enterprises.16458.4.1.1.1*. The trap payload also indicates the specific HyperStore host on which the trap-triggering event occurred. HyperStore uses SNMP version 2c. Trap are sent by the [HyperStore System Monitoring / Cron Job host](#).

monitoring.snmp.traps.receiver.* setting	Default Value
monitoring.snmp.traps.receiver.address	empty
monitoring.snmp.traps.receiver.port	162

The individual *monitoring.snmp.traps.receiver.** settings are:

- *address* -- IP address or FQDN of the SNMP manager to which the CMC will send system alerts (for alert types for which you've enabled SNMP trap sending in the CMC's **Alert Rules** page). This setting does not support specifying multiple SNMP managers. Specify just a single SNMP manager address.
- *port* -- Listening port used by the SNMP manager.

To **change either of the settings listed in the table above** and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set <setting>=<value>
# hsctl config apply monitoring
```

No service restart is required.

For example:

```
# hsctl config set monitoring.snmp.traps.receiver.address=123.123.123.123
# hsctl config apply monitoring
```

Note You also have the option of configuring these settings in the **SNMP Trap Destination Settings** section of the CMC's **"Configuration Settings"** (page 498) page.

13.2.13. Redis Monitor Settings

The table below lists configuration settings in the *redis.monitoring* setting group, which configures the [Redis Monitor](#) service.

Setting	Default Value
redis.monitoring.logging.level (page 810)	INFO
redis.monitoring.logging.rollover.* settings	see "Log Rolling Settings" (page 689) for default values

13.2.13.1. redis.monitoring.* Setting Descriptions

redis.monitoring.logging.level

This setting controls the level at which Redis Monitor application logging will occur. Only messages with a severity level that matches or exceeds your configured logging level will be logged.

Listed in order from the least inclusive logging to the most inclusive logging, the supported logging levels are:

- OFF -- The OFF "level" turns off logging.
- FATAL -- The FATAL level designates very severe error events that will likely lead the application to abort.
- ERROR -- The ERROR level designates error events that may still allow the application to continue running.
- WARN -- The WARN level designates potentially harmful situations.

- INFO -- The INFO level designates informational messages that indicate the progress of the application at coarse-grained level.
- DEBUG -- The DEBUG level designates fine-grained informational events that are most useful to debug an application.
- TRACE -- The TRACE level designates finer-grained informational events than the DEBUG level.

For example, if this setting is set to WARN then only messages with severity levels WARN, ERROR, or FATAL will be logged by the Redis Monitor.

Default = INFO

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set redis.monitoring.logging.level=<string>
# hscctl config apply redis
# hscctl service restart redismon --nodes=ALL
```

13.2.14. S3 Service Settings

The table below lists configuration settings in the `s3` setting group, which configures the [S3 Service](#). When applicable the settings in the table are organized into sub-groups.

Setting Sub-Group	Setting	Default Value
s3.batchOperations	s3.batchOperations.delete.delay (page 813)	30
	s3.batchOperations.delete.maxPerDC (page 813)	10
	s3.batchOperations.delete.queuePollInterval (page 814)	60
	s3.batchOperations.purge.threads.max (page 814)	6
s3.client	s3.client.connectionTimeout (page 815)	2000
	s3.client.maxRetries (page 816)	3
	s3.client.socketTimeout (page 816)	50000
	s3.client.upload.multipart.partSize (page 816)	5242880
	s3.client.upload.multipart.threshold (page 817)	16777216
	s3.client.upload.socketTimeout (page 817)	1800000
s3.heap	s3.heap.auto.initPercent (page 819)	25
	s3.heap.auto.limit (page 820)	30g
	s3.heap.auto.maxPercent (page 820)	15
	s3.heap.maxSize (page 820)	empty
	s3.heap.minSize (page 820)	empty
s3.hyperstore.connections	s3.hyperstore.connections.maxPerRoute (page 821)	specific to your system
	s3.hyperstore.connections.maxTotal (page 821)	specific to your system
	s3.hyperstore.connections.timeout (page 822)	10000
	s3.hyperstore.connections.sessionTimeout (page 821)	10000

Setting Sub-Group	Setting	Default Value
s3.logging	s3.logging.level (page 823)	INFO
	s3.logging.rollover.* settings	see "Log Rolling Settings" (page 689) for default values
s3.ports	s3.ports.loadBalancer.http (page 824)	80
	s3.ports.loadBalancer.https (page 825)	443
	s3.ports.main.http (page 825)	80
	s3.ports.main.https (page 825)	443
	s3.ports.proxy.http (page 826)	81
	s3.ports.proxy.https (page 826)	4431
s3.qos	s3.qos.bucketLevel (page 828)	false
	s3.qos.enabled (page 829)	false
	s3.qos.trackRate (page 829)	false
s3.sse	s3.sse.aes256 (page 830)	false
	s3.sse.kms.region (page 831)	us-east-1
	s3.sse.tls.required (page 831)	true
s3.threads	s3.threads.idleTimeout (page 832)	60000
	s3.threads.max (page 832)	specific to your system
	s3.threads.min (page 832)	100
s3.tiering	s3.tiering.endpoints.* (page 833)	see setting descriptions for defaults
	s3.tiering.forwardDeletes (page 836)	true
	s3.tiering.userAgent (page 836)	APN/1.0 Cloudian/1.0 Hyper-Store/
s3.tls	s3.tls.enabled / s3.tls.required (page 837)	true / false
	s3.tls.verifyCA (page 838)	false
s3.usage	s3.usage.bucketLevel (page 838)	false
	s3.usage.reports.dailyRollupTTL (page 839)	5616000 (65 days)
	s3.usage.reports.hourlyRollupTTL (page 839)	5616000 (65 days)
	s3.usage.reports.monthlyRollupTTL (page 839)	15552000 (180 days)
	s3.usage.trackRate (page 840)	false
s3.user	s3.user.maxBuckets (page 840)	100

Setting Sub-Group	Setting	Default Value
n/a (miscellaneous individual settings)	s3.user.maxCredentials (page 841)	5
	s3.bucketLogging.internalNetworks (page 815)	empty
	s3.crr.syncBucket (page 817)	false
	s3.cyberstorage.putObjectCopy.blockOverwriteEncryption (page 818)	true
	s3.directory.temp (page 818)	/tmp
	s3.enabled (page 819)	true
	s3.getBucketEncryption.allowPolicyResponse (page 819)	false
	s3.kmip.manageProfile.grantees (page 822)	SystemAdmin
	s3.lowRes.idleTimeout (page 824)	5000
	s3.multipart.maxParts (page 824)	10000
	s3.proxyProtocol.enabled (page 826)	false
	s3.putObject.maxSize (page 827)	5.36870912e+09
	s3.serverSignature (page 830)	CloudianS3
	s3.sharedBucket.enabled (page 830)	false
	s3.stackSize (page 831)	empty

13.2.14.1. s3.* Setting Descriptions

s3.batchOperations.delete.delay

[Former name in HyperStore 7.x and earlier: *cloudian.s3.batch.delete.delay*]

When executing the recurring batch processing operation to purge deleted objects from disk, the number of milliseconds to pause in between purges of individual objects.

If *s3.batchOperations.delete.delay* is set to 0, then when the batch processing job is running there is no delay between individual object purges.

The *s3.batchOperations.delete.delay* setting provides a way to "throttle" the execution of the deleted object batch processing operation (to slow it down so as to reduce its resource demands).

Default = 30

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.batchOperations.delete.delay=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.batchOperations.delete.maxPerDC

[Former name in HyperStore 7.x and earlier: *cloudian.dc.delete.instances*]

This setting specifies the maximum number of nodes that can concurrently run batch delete processing jobs within one data center. This provides a way of throttling how much batch delete processing work can happen concurrently in a data center.

For example, with `s3.batchOperations.delete.maxPerDC` set to 2, if batch delete processing jobs are running on two nodes in a data center and the time arrives for a batch delete processing job to kick off on another node in that same data center (based on "[s3.batchOperations.delete.queuePollInterval](#)" (page 814)), the kick-off of that job will wait until one of the currently running batch delete processing jobs completes -- so that no more than two jobs are running concurrently per data center.

If you set `s3.batchOperations.delete.maxPerDC` to 0, this means there is no limit -- so batch processing jobs can concurrently run on all nodes in a data center.

Default = 10

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set s3.batchOperations.delete.maxPerDC=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.batchOperations.delete.queuePollInterval

[Former name in HyperStore 7.x and earlier: `cloudian.delete.queue.poll.interval`]

When S3 requests for object deletion are received and successfully processed by HyperStore, the system immediately marks the objects as having been deleted, and stores this object deletion flag in a queue in the Metadata DB. But the actual purging of object data from disk does not occur until a batch processing job runs, to physically purge objects that have been marked as deleted.

The `s3.batchOperations.delete.queuePollInterval` property sets the interval at which to run the batch processing of queued object deletes, in number of minutes. With the default of 60, the object deletion batch processing job is run once per hour on each node. For performance reasons the kick-off times are staggered across the nodes -- so, all nodes run the job once per hour (by default configuration) but the jobs do not all kick off at the same time. However because a batch processing job takes some time to complete, there may be overlap such that batch processing jobs are running on multiple nodes concurrently.

Also for performance reasons, when a batch processing job is being run on a node, that node manages the batch but the work of implementing the purge actions contained in that batch is spread across the cluster.

Note This property can be set to "0" to disable batch processing of objects marked for deletion, but if you do this it should only be temporarily, and in consultation with Clouidian Support.

Default = 60

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set s3.batchOperations.delete.queuePollInterval=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.batchOperations.purge.threads.max

[Former name in HyperStore 7.x and earlier: `cloudian.delete.purge.bucket.threads`]

Maximum number of threads to devote to purging a bucket (while executing the purging associated with the Admin API call *POST /bucketops/purge*). For details on this API call see the "bucketops" section of the **Cloudian HyperStore Admin API Reference**.

Default = 6

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.batchOperations.purge.threads.max=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.bucketLogging.internalNetworks

[Former name in HyperStore 7.x and earlier: *s3_bucket_logging_internalnetworks*]

This setting applies to the S3 bucket logging feature, whereby a bucket owner can choose to have access logs for that bucket regularly saved into that bucket or a different bucket owned by that user. For more information on this feature see ["Enable/Disable Bucket Logging" \(page 334\)](#).

The *s3.bucketLogging.internalNetworks* setting gives you as the system administrator the ability to mask certain fields in bucket log entries for S3 requests that originate from within your internal network(s). This would include for example S3 requests that originate from the CMC, or from other hosts within your internal network (s). This prevents users of the bucket logging feature from seeing, in their bucket logs, information that you might consider sensitive. In particular the bucket log entry fields that can be masked are:

- Remote IP (request originating IP address)
- User-Agent

For each internal network (specified in CIDR format) that you list in the *s3.bucketLogging.internalNetworks* setting, when S3 requests originate from within those networks the Remote IP and User-Agent field values in bucket log entries will be replaced by a dash ("-").

Example of a correctly formatted setting value:

```
150.5.2.0/24, 150.5.3.0/24
```

Note To mask the Remote IP and User-Agent fields for S3 requests that originate from the **CMC**, include 127.0.0.0/8 in the list of internal networks (since the CMC usually uses *localhost* when submitting requests to the S3 Service).

Default = empty

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.bucketLogging.internalNetworks=<list>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.client.connectionTimeout

[Former name in HyperStore 7.x and earlier: *cloudian.s3.client.ConnectionTimeout*]

When HyperStore's native S3 client is interfacing with a remote S3 Service (such as during auto-tiering), the connection establishment timeout in milliseconds.

Default = 2000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.client.connectionTimeout=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.client.maxRetries

[Former name in HyperStore 7.x and earlier: cloudian.s3.client.MaxErrorRetry]

When HyperStore's native S3 client is interfacing with a remote S3 Service (such as during auto-tiering), the maximum number of times to retry sending a request that encounters a temporary error response from the remote service (such as a 503 error).

For auto-tiering, after this many retries the request will be put back in the auto-tiering queue and tried again at the next running of the auto-tiering cron job.

Default = 3

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.client.maxRetries=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.client.socketTimeout

[Former name in HyperStore 7.x and earlier: cloudian.s3.client.SocketTimeout]

When HyperStore's native S3 client is interfacing with a remote S3 Service (such as during auto-tiering), the request processing timeout in milliseconds. When a request is sent over an open connection, if a complete response is not received within this interval, the request times out and the connection is closed.

Default = 50000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.client.socketTimeout=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.client.upload.multipart.partSize

[Former name in HyperStore 7.x and earlier: s3.upload.part.minsize]

When HyperStore uses Multipart Upload to transmit a Node Diagnostics package to Clouidian Support (or a different S3 destination if you've configured the system to send these packages to a different destination), each of the parts will be this many bytes or larger — with the exception of the final part, which may be smaller. For example, if Multipart Upload is used for an 18MiB object, and the configured minimum part size is 5MiB, the object will be transmitted in four parts of size 5MiB, 5MiB, 5MiB, and 3MiB.

Default = 5242880 (5MiB)

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.client.upload.multipart.partSize=<integer>
# hsctl config apply s3 cmc
# hsctl service restart s3 cmc --nodes=ALL
```

s3.client.upload.multipart.threshold

[Former name in HyperStore 7.x and earlier: s3.upload.part.threshold]

When HyperStore transmits a Node Diagnostics package to an S3 destination such as Cloudian Support (or a different S3 destination if you've configured the system to send these packages to a different destination), it uses Multipart Upload if the package is larger than this many bytes.

Default = 16777216 (16MiB)

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.client.upload.multipart.threshold=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.client.upload.socketTimeout

[Former name in HyperStore 7.x and earlier: s3.client.timeout]

When uploading a Node Diagnostics package to Cloudian Support (or a different S3 destination if you've configured the system to send these packages to a different destination), the socket timeout in milliseconds.

Default = 1800000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.client.upload.socketTimeout=<integer>
# hsctl config apply s3 cmc
# hsctl service restart s3 cmc --nodes=ALL
```

s3.crr.syncBucket

[Former name in HyperStore 7.x and earlier: cloudian.s3.crr.syncbucket]

Once every four hours a HyperStore [cron job](#) processes any pending [cross-region replication](#) tasks. Typically with cross-region replication, objects are replicated to the destination bucket as soon as they are uploaded to the source bucket. Pending cross-region replication tasks result when an attempt to replicate an object from the source bucket to the destination bucket returns a temporary error such as a connection failure or an HTTP 5xx error. The cron job retries the replication of such objects.

- If *s3.crr.syncBucket* is set to *true*, then each time the cron job runs it also performs a complete sync-up of each bucket for which there are replication tasks in the retry queue. With this complete sync-up, any objects that are in these source buckets but not in their corresponding destination buckets will be replicated to the destination buckets. This includes objects that were in the source buckets before cross-region replication was enabled on the buckets, if any. (The exceptions are objects that were in the source buckets before versioning was enabled on the bucket, if any; and objects that are [encrypted by the SSE-C or AWS-KMS methods](#), if any. Such objects will not be replicated.).

- If `s3.crr.syncBucket` is set to `false`, the cron job will only process pending cross-region replication jobs (retries of object replication attempts that met with temporary errors). This makes the cron job less resource-intensive than it would be if `s3.crr.syncBucket` is set to `true`.

Default = false

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set s3.crr.syncBucket=<boolean>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

Note Setting `s3.crr.syncBucket` to `true` will not affect replication processing for source buckets for which there are not any pending replication tasks in the retry queue. It applies only to buckets for which there are pending replication tasks in the retry queue. If you want to trigger a complete replication sync-up for a particular source bucket you can do so as described in ["Triggering the Replication of a Bucket's Pre-Existing Objects"](#) (page 220).

s3.cyberstorage.putObjectCopy.blockOverwriteEncryption

This setting determines whether the HyperStore S3 Service will block S3 `PutObjectCopy` operations that are attempting to copy an object over itself, with the same content except encrypted by SSE-C encryption -- `true` or `false`. This is a potential attack vector for ransomware attacks, once an S3 credential has been stolen. The targeted objects would be encrypted in place, and with an SSE-C passphrase known only to the attackers.

If the S3 Service blocks this type of operation, a HTTP 403 response is returned to the client, and an alert is displayed in the CMC's **Alerts** interface. (Rather than one alert per object there will be one alert per node per SSE-C passphrase, per day.)

Your `s3.cyberstorage.putObjectCopy.blockOverwriteEncryption` configuration setting is a system-wide default setting. Bucket owners can override the system-wide setting for their own buckets if they wish, using the CMC's **Bucket Properties** interface. For example, if the system default is to block such requests, a user might temporarily allow such requests on their own bucket if they for some reason were actually making the requests themselves.

Default = true

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set s3.cyberstorage.putObjectCopy.blockOverwriteEncryption=<boolean>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.directory.tmp

[Former name in HyperStore 7.x and earlier: `cloudian.s3.tmpdir`]

The directory in which to temporarily store large files in order to reduce memory usage.

Default = `/tmp`

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set s3.directory.temp=<string>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.enabled

This setting determines whether the S3 Service is enabled, *true* or *false*. This also determines whether the S3 Service will start automatically when a host reboots.

Default = true

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.enabled=<boolean>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.getBucketEncryption.allowPolicyResponse

If this setting is *true*, then an S3 *GetBucketEncryption* operation for a bucket will indicate that server-side encryption is enabled for the bucket if either of these conditions is true:

- Encryption has been explicitly set for the bucket, by use of the S3 *PutBucketEncryption* operation.
- The bucket has inherited encryption from its storage policy. In the case where encryption has not been explicitly set for the bucket but the bucket has inherited encryption from its storage policy, in the *GetBucketEncryption* response body the *SSEAlgorithm* element will indicate *AES256* regardless of which encryption method the bucket's storage policy uses, regular SSE or SSE-KMIP.

If this setting is *false*, then an S3 *GetBucketEncryption* operation for a bucket will indicate that encryption is enabled for the bucket only if encryption has been explicitly set at the bucket level, by use of the S3 *PutBucketEncryption* operation. Encryption set at the storage policy level will not impact the *GetBucketEncryption* response.

Default = false

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.getBucketEncryption.allowPolicyResponse=<boolean>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.heap.auto.initPercent

[Former name in HyperStore 7.x and earlier: *cloudian_s3_init_heap_percent*]

Initial heap size (JAVA_OPTS -Xms value) for the S3 Service application, as a percentage of the -Xmx value.

Default = 25

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.heap.auto.initPercent=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.heap.auto.limit

[Former name in HyperStore 7.x and earlier: cloudian_s3_heap_limit]

Maximum heap size limit for the S3 Service application. The JAVA_OPTS -Xmx value passed to the JVM will be the **lower** of this value and the percent-of-system-RAM value set by *s3.heap.auto.maxPercent*.

Default = 30g

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.heap.auto.limit=<string>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.heap.auto.maxPercent

[Former name in HyperStore 7.x and earlier: cloudian_s3_max_heap_percent]

Maximum heap size for the S3 Service application, as a percentage of host system RAM. The JAVA_OPTS -Xmx value passed to the JVM will be the **lower** of this value and the limiting value set by *s3.heap.auto.limit*.

Default = 15

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.heap.auto.maxPercent=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.heap.maxSize

On each node, the memory heap maximum size setting (Java *Xmx* setting) for the S3 Service application.

If this is explicitly set, it overrides the upper limit derived from the *s3.heap.auto.** settings.

Default = not set

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.heap.maxSize=<string>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.heap.minSize

On each node, the memory heap minimum size setting (Java *Xms* setting) for the S3 Service application.

If this is explicitly set, it overrides the lower limit derived from the *s3.heap.auto.** settings.

Default = not set

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.heap.minSize=<string>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```


s3.hyperstore.connections.maxPerRoute

[Former name in HyperStore 7.x and earlier: *hyperstore_maxperrouteconnections*]

The maximum allowed number of concurrently open connections from each S3 Service instance to each HyperStore Service instance. This allows for limiting the traffic load between each S3 Service instance (as it processes incoming requests from S3 clients) and any single HyperStore Service instance. Note that each of your nodes has an S3 Service instance and a HyperStore Service instance.

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.hyperstore.connections.maxPerRoute=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use *hsctl config set* to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

s3.hyperstore.connections.maxTotal

[Former name in HyperStore 7.x and earlier: *hyperstore_maxtotalconnections*]

The maximum allowed number of concurrently open connections between each S3 Service instance and all HyperStore Service instances in aggregate. This allows for limiting the traffic load between each S3 Service instance (as it processes incoming requests from S3 clients) and the whole HyperStore Service storage layer. Note that each of your nodes has an S3 Service instance and a HyperStore Service instance.

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.hyperstore.connections.maxTotal=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use *hsctl config set* to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

s3.hyperstore.connections.sessionTimeout

[Former name in HyperStore 7.x and earlier: *hyperstore_timeout*]

For the S3 Service's connections to the HyperStore Service, the transaction completion timeout (session timeout) in milliseconds.

Default = 10000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.hyperstore.connections.sessionTimeout=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

For a diagram showing the place of this timeout within the S3 request processing flow, see the description of ["hyperstore.cassandra.thrift.timeout"](#) (page 764).

s3.hyperstore.connections.timeout

[Former name in HyperStore 7.x and earlier: *hyperstore_connection_timeout*]

For the S3 Service's connections to the HyperStore Service, the connection establishment timeout in milliseconds.

Default = 10000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.hyperstore.connections.timeout=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

For a diagram showing the place of this timeout within the S3 request processing flow, see the description of ["hyperstore.cassandra.thrift.timeout"](#) (page 764).

s3.kmip.manageProfile.grantees

This setting controls which CMC login roles will be allowed to create "KMIP profiles", using the [CMC's interface for creating these profiles](#). A KMIP profile identifies a specific Key Management System (KMS) that supports the Key Management Interoperability Protocol (KMIP) for purposes of server-side encryption of stored object data, and encapsulates the information needed for HyperStore to connect to that KMS.

With the *s3.kmip.manageProfile.grantees* setting you can specify one or more of these CMC roles:

- SystemAdmin
- GroupAdmin
- User

Use a comma-separated list if specifying more than one role (for example *"SystemAdmin,GroupAdmin,User"* to allow system administrators, group administrators, and regular users to create KMIP profiles).

The usage scope of a KMIP profile varies depending on which type of CMC login role created the profile:

- KMIP profiles created by a **system administrator** are available for the purpose of setting KMIP encryption as a storage policy attribute, and are also available to all users in all user groups for the purpose of setting KMIP encryption for buckets for which KMIP is not already configured as part of the buckets' storage policy.
- KMIP profiles created by a **group administrator** are available to all users in that group, for the purpose of setting KMIP encryption for buckets for which KMIP is not already configured as part of the buckets' storage policy.
- KMIP profiles created by a **regular user** are available only to that user, for the purpose of setting KMIP encryption for buckets for which KMIP is not already configured as part of the buckets' storage policy.

The *s3.kmip.manageProfile.grantees* setting works by having the CMC display or hide the KMIP profile creation interface, depending on the setting value:

- If and only if your *s3.kmip.manageProfile.grantees* setting value includes *SystemAdmin*, then when a system administrator is logged into the CMC they will see a KMIP profile creation interface at **Cluster -> KMIP Profiles**.

- If and only if your `s3.kmip.manageProfile.grantees` setting value includes *GroupAdmin* and/or *User*, then when a group administrator or regular user is logged into the CMC they will see a KMIP profile creation interface at **Server-Side Encryption -> KMIP Profiles**.

For general information about HyperStore's server-side encryption feature see ["Server-Side Encryption" \(page 186\)](#). For more information about setting up and using KMIP-based server-side encryption see ["Using SSE-KMIP" \(page 191\)](#).

Default = SystemAdmin

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set s3.kmip.manageProfile.grantees=<string>
# hsctl config apply s3 cmc
# hsctl service restart s3 cmc --nodes=ALL
```

s3.logging.level

This setting controls the level at which S3 Service application logging will occur. Only messages with a severity level that matches or exceeds your configured logging level will be logged.

Listed in order from the least inclusive logging to the most inclusive logging, the supported logging levels are:

- OFF -- The OFF "level" turns off logging.
- FATAL -- The FATAL level designates very severe error events that will likely lead the application to abort.
- ERROR -- The ERROR level designates error events that may still allow the application to continue running.
- WARN -- The WARN level designates potentially harmful situations.
- INFO -- The INFO level designates informational messages that indicate the progress of the application at coarse-grained level.
- DEBUG -- The DEBUG level designates fine-grained informational events that are most useful to debug an application.
- TRACE -- The TRACE level designates finer-grained informational events than the DEBUG level.

For example, if this setting is set to WARN then only messages with severity levels WARN, ERROR, or FATAL will be logged by the S3 Service.

Default = INFO

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set s3.logging.level=<string>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

Note This setting applies to the S3 Select Service as well as to the main S3 Service. (The S3 Service proxies S3 *SelectObjectContent* requests through to the S3 Select Service.)

s3.lowRes.idleTimeout

Special, "low resource" maximum idle time to apply to S3 Service HTTP(S) connections when the number of simultaneous connections to an S3 Service node exceeds 1000. Configured in milliseconds. With this setting, you can have the S3 Service be less tolerant of connection idle time during times of high concurrent usage. (For a general description of idle timer behavior, see [iam.threads.idleTimeout](#).)

Default = 5000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.lowRes.idleTimeout=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.multipart.maxParts

[Former name in HyperStore 7.x and earlier: *cloudian_s3_multipart_maxparts*]

Maximum number of parts to allow per S3 *CreateMultipartUpload* request.

Each individual part can be no larger than the limit configured by the "[s3.putObject.maxSize](#)" (page 827) setting.

Default = 10000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.multipart.maxParts=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

Note You also have the option of configuring this setting in the **S3 Request Restriction Settings** section of the CMC's "[Configuration Settings](#)" (page 498) page. In the CMC UI this setting is called "Multipart Upload Maximum Parts".

s3.ports.loadBalancer.http

[Former name in HyperStore 7.x and earlier: *ld_cloudian_s3_port*]

The S3 Service HTTP port on the load balancer.

Default = 80

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.ports.loadBalancer.http=<integer>
# hsctl config apply all
# hsctl service restart s3 iam cmc --nodes=ALL
```

Note You also have the option of configuring this setting by using the installer's "Advanced Configuration Options", as described in "[Changing S3, Admin, or CMC Listening Ports](#)" (page 858).

s3.ports.loadBalancer.https

[Former name in HyperStore 7.x and earlier: *ld_cloudian_s3_ssl_port*]

The S3 Service HTTPS port on the load balancer.

Default = 443

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.ports.loadBalancer.https=<integer>
# hsctl config apply all
# hsctl service restart s3 iam cmc --nodes=ALL
```

Note You also have the option of configuring this setting by using the installer's "Advanced Configuration Options", as described in ["Changing S3, Admin, or CMC Listening Ports"](#) (page 858).

s3.ports.main.http

[Former name in HyperStore 7.x and earlier: *cloudian_s3_port*]

On each node, the port on which the S3 Service listens for regular HTTP connections.

Default = 80

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.ports.main.http=<integer>
# hsctl config apply all
# hsctl service restart s3 firewall iam cmc --nodes=ALL
```

Note

- * You also have the option of configuring this setting by using the installer's "Advanced Configuration Options", as described in ["Changing S3, Admin, or CMC Listening Ports"](#) (page 858).
- * For information about enabling/disabling the local HTTP and HTTPS listeners for the S3 Service on each node, see ["s3.tls.enabled / s3.tls.required"](#) (page 837).

s3.ports.main.https

[Former name in HyperStore 7.x and earlier: *cloudian_s3_ssl_port*]

On each node, the port on which the S3 Service listens for HTTPS connections.

Default = 443

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.ports.main.https=<integer>
# hsctl config apply all
# hsctl service restart s3 firewall iam cmc --nodes=ALL
```

Note

- * You also have the option of configuring this setting by using the installer's "Advanced Configuration

Options", as described in ["Changing S3, Admin, or CMC Listening Ports"](#) (page 858).

* For information about enabling/disabling the local HTTP and HTTPS listeners for the S3 Service on each node, see ["s3.tls.enabled / s3.tls.required"](#) (page 837).

s3.ports.proxy.http

If you've configured the S3 Service to support the PROXY Protocol (you've set ["s3.proxyProtocol.enabled"](#) (page 826) to *true*), the *s3.ports.proxy.http* setting specifies the port on which the S3 Service will listen for incoming HTTP requests that use the PROXY Protocol.

Default = 81

To change this setting and apply your change, run the following *hstcl* commands on the Config Controller node:

```
# hstcl config set s3.ports.proxy.http=<integer>
# hstcl config apply s3 firewall
# hstcl service restart s3 firewall --nodes=ALL
```

s3.ports.proxy.https

If you've configured the S3 Service to support the PROXY Protocol (you've set ["s3.proxyProtocol.enabled"](#) (page 826) to *true*), the *s3.ports.proxy.https* setting specifies the port on which the S3 Service will listen for incoming HTTPS requests that use the PROXY Protocol.

Default = 4431

To change this setting and apply your change, run the following *hstcl* commands on the Config Controller node:

```
# hstcl config set s3.ports.proxy.https=<integer>
# hstcl config apply s3 firewall
# hstcl service restart s3 firewall --nodes=ALL
```

s3.proxyProtocol.enabled

[Former name in HyperStore 7.x and earlier: *s3_proxy_protocol_enabled*]

If you are using [HAProxy](#) as the load balancer in front of your S3 Service -- or a different load balancer that supports the [PROXY Protocol](#) -- you can set *s3.proxyProtocol.enabled* to *true* if you want the S3 Service to support the PROXY Protocol. If you do so, the following will happen:

- The S3 Service will create dedicated PROXY Protocol connectors listening on port 81 (for regular PROXY Protocol) and port 4431 (for PROXY Protocol with SSL/TLS). (These ports are configurable by the ["s3.ports.proxy.http"](#) (page 826) and ["s3.ports.proxy.https"](#) (page 826) settings; the defaults are 81 and 4431 respectively.)
- If you configure your load balancer to use the PROXY Protocol for communicating with the S3 Service, the load balancer when relaying each S3 request to the S3 Service will pass along the originating client's IP address.

Note For guidance on load balancer configuration consult with your Clodian Sales Engineering or Professional Services representative.

- In the [S3 request log](#), the S3 request entries will then show the true originating client IP address as the source address, rather than showing the loader balancer's IP address as the source. Also, the true originating client IP address for each request will be available to support using S3 bucket policies that filter based on source IP address; or billing rating plans that "allowlist" certain source IP addresses.

Setting `s3.proxyProtocol.enabled` to `true` is appropriate if you're using HAProxy for load balancing -- or a different load balancer that supports the PROXY Protocol -- and you want S3 Service request logging to show the true origin address associated with S3 requests; and/or you want to implement bucket policies or rating plans that are responsive to the origin address. If you're using a load balancer that supports PROXY Protocol, using this protocol is the preferred method for providing the originating client IP address to the S3 Service, rather than using the *X-Forwarded-for* header.

Note If you intend to use the PROXY Protocol with TLS/SSL (on S3 Service listening port 4431) you must set up TLS/SSL for the S3 Service, if you have not already done so. For instructions see ["Security and Privacy Features"](#) (page 177).

If `s3.proxyProtocol.enabled` is set to `true` then when you configure TLS/SSL for the S3 Service your configuration information will be applied to PROXY Protocol port 4431 as well as to the regular S3 HTTPS port 443.

If `s3.proxyProtocol.enabled` is set to `false` then the S3 Service will not support the PROXY Protocol and will not create PROXY Protocol listeners.

Default = false

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set s3.proxyProtocol.enabled=<boolean>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.putObject.maxSize

[Former name in HyperStore 7.x and earlier: `putobject_max_size`]

The maximum single-part object size that S3 clients will be allowed to upload to the system through an S3 *PutObject* request (or *POST Object* request), in number of bytes. If an S3 client submits a *PutObject* or *POST Object* request with an object size larger than this many bytes, the S3 Service rejects the request.

In the case of Multipart Upload operations, this maximum size restriction applies to each individual uploaded part. If a single part is larger than Put Object Maximum Size, the S3 Service rejects the S3 *UploadPart* request.

For best upload performance it's recommended for S3 clients to use the S3 *CreateMultipartUpload* operation for objects larger than 100MB (rather than *PutObject* or *POST Object*).

Default = 5368709120 (5 GiB)

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set s3.putObject.maxSize=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

Note You also have the option of configuring this setting in the **S3 Request Restriction Settings** section of the CMC's "**Configuration Settings**" (page 498) page. In the CMC UI this setting is called "Put Object Maximum Size (Bytes)".

Note In HyperStore version 7.1.x and older, the default was 5000000000 (5 GB rather than 5 GiB). If you upgraded from HyperStore 7.1.x to later versions, your system retains the default of 5000000000 unless you change the setting value.

s3.qos.bucketLevel

[Former name in HyperStore 7.x and earlier: `s3_perbucket_qos_enabled`]

This setting enables/disables the keeping of per-bucket stored-bytes counts (the number of net bytes -- excluding replication or erasure coding overhead -- stored in each bucket) and stored-objects counts (the number of objects in each bucket) in the system.

If you set `s3.qos.bucketLevel` to `true`, then:

- The system will keep track of stored-bytes and stored-objects counts for each bucket. These counts will be kept in the QoS DB.

Note In some atypical circumstances the keeping of these per-bucket stored-bytes and stored-object counts, which will be updated after every S3 transaction, may cause a minor-to-moderate decline in S3 write performance. Example circumstances would be if there are an exceptionally large number of buckets in your HyperStore system, or if you have a multi-data center HyperStore deployment with substantial latency between the data centers. If you want to enable per-bucket stored-bytes and stored-objects counters in your system but are concerned about potential performance impacts, consult with Cloudian Support.

- You will be able to query the current stored-bytes and stored-objects counts for individual buckets, by using the Admin API calls `GET /system/bytcount` and `GET /system/objectcount`. You will also be able to retrieve the byte and object counts for all buckets owned by all users in a specified group, by using the Admin API call `GET /system/bucketusage`. For more information on these Admin API calls see the "system" section of **Cloudian HyperStore Admin API Reference**.
- If you use Cloudian's monitoring and visualization product [HyperIQ](#), HyperIQ will be able to report stored-bytes and stored-objects counts for individual buckets.
- In the CMC, the [object listing page for a bucket](#) will display the bucket's current stored-bytes and stored-objects counts.

If `s3.qos.bucketLevel` is set to `false`, then per-bucket stored-bytes and stored-objects counts will not be maintained in the QoS DB; you will not be able to query per-bucket counts through the Admin API calls referenced above; HyperIQ will not be able to report on per-bucket stored-bytes and stored-objects counts; and the CMC's object listing page for a bucket will not show stored-bytes and stored-objects counts.

Default = false

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:


```
# hsctl config set s3.qos.bucketLevel=<boolean>
# hsctl config apply s3 cmc
# hsctl service restart s3 cmc --nodes=ALL
```

IMPORTANT ! After setting `s3.qos.bucketLevel` to `true` and applying your change (as described above), execute the Admin API call `POST /usage/repair?groupId=ALL` to bring the counters up to date for buckets that already have objects in them. For more information about this Admin API call see the "usage" section of the *Cloudian HyperStore Admin API Reference*.

Subsequently, the counters will automatically be updated for each bucket each time there is an S3 transaction that impacts the byte count or object count for the bucket.

s3.qos.enabled

[Former name in HyperStore 7.x and earlier: `cloudian_s3_qos_enabled`]

- If `s3.qos.enabled` is set to `true` the system will enforce Quality of Service (QoS) limits for number of stored bytes and number of stored objects, if you have [set such limits for users and groups](#). (If you want to **also** enforce QoS limits for data transfer rates, then also set the `"s3.qos.trackRate"` (page 829) setting to `true`.)
- If `s3.qos.enabled` is set to `false` the system will not enforce any QoS limits, even if you have set such limits for users and groups.

Default = false

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set s3.qos.enabled=<boolean>
# hsctl config apply s3 cmc
# hsctl service restart s3 cmc --nodes=ALL
```

Note You also have the option of configuring this setting in the **Quality of Service Settings** section of the CMC's ["Configuration Settings"](#) (page 498) page. In the CMC UI this setting is called "Enforce Configured QoS Limits for Storage Utilization".

s3.qos.trackRate

[Former name in HyperStore 7.x and earlier: `cloudian_s3_qos_rate_enabled`]

Applicable only if `"s3.qos.enabled"` (page 829) is set to `true` (by default it is set to `false`).

- If `s3.qos.trackRate` is set to `true` the system will enforce Quality of Service (QoS) limits for HTTP request rate, bytes-in rate, and bytes-out rate, if you have [set such limits for users and groups](#).
- If `s3.qos.trackRate` is set to `false` the system will not enforce QoS limits for HTTP request rate, bytes-in rate, and bytes-out rate, even if you have set such limits for users and groups.

Default = false

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hscctl config set s3.qos.trackRate=<boolean>
# hscctl config apply s3 cmc
# hscctl service restart s3 cmc --nodes=ALL
```

Note You also have the option of configuring this setting in the **Quality of Service Settings** section of the CMC's **"Configuration Settings"** (page 498) page. In the CMC UI this setting is called "Enforce Configured QoS Limits for Request Rates and Data Transfer Rates".

s3.serverSignature

[Former name in HyperStore 7.x and earlier: *cloudian.s3.server*]

HTTP Server header value for the S3 Service to return in responses to S3 requests. If you want no Server header value returned in responses to S3 requests, set this setting to to `[ ]` (brackets with no space between, this denotes an empty value).

Default = CloudianS3

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set s3.serverSignature=<string>
# hscctl config apply s3
# hscctl service restart s3 --nodes=ALL
```

s3.sharedBucket.enabled

[Former name in HyperStore 7.x and earlier: *cloudian.s3.enablesharedbucket*]

To enable an HyperStore extension to the S3 *ListBuckets* API call so that S3 users can list not only their own buckets but also any buckets that other users have shared with them, set *s3.sharedBucket.enabled* to *true*.

Default = false

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set s3.sharedBucket.enabled=<boolean>
# hscctl config apply s3
# hscctl service restart s3 --nodes=ALL
```

Note For information about the relevant S3 API call and how to use the extension, see *ListBuckets* in the S3 section of the **Cloudian HyperStore AWS APIs Support Reference**.

s3.sse.aes256

[Former name in HyperStore 7.x and earlier: *cloudian_s3_aes256encryption_enabled*]

Set this to *true* if you want to enable AES-256 for use with HyperStore server-side encryption. If this is set to *false*, AES-128 is used instead.

Default = false

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hsctl config set s3.sse.aes256=<boolean>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.sse.kms.region

[Former name in HyperStore 7.x and earlier: *util.awskmsutil.region*]

Amazon Web Services (AWS) service region to use if you are configuring your HyperStore system to support AWS-KMS as a method of server-side encryption. For complete instructions see ["Using AWS KMS" \(page 195\)](#).

Default = us-east-1

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.sse.kms.region=<string>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.sse.tls.required

[Former name in HyperStore 7.x and earlier: *cloudian.s3.ssec.usessl*]

This setting controls whether the S3 Service will require that incoming S3 requests use HTTPS (rather than regular HTTP) connections when the request is using Server Side Encryption with Customer-provided encryption keys (SSE-C). Leaving this setting at its default of *true* -- so that the S3 Service requires HTTPS connections for such requests -- is the recommended configuration. The only circumstance in which you might set this to *false* is if:

- You are using a load balancer in front of the S3 Service hosts -- and the load balancer, when receiving an incoming HTTPS request from clients, terminates the SSL and uses regular HTTP to connect to an S3 Service host over your internal network.
- AND, you trust your internal network to safely transport users' encryption keys from the load balancer to the S3 Service hosts over regular HTTP.

For background information about HyperStore support for server-side encryption (SSE and SSE-C), see ["Server-Side Encryption" \(page 186\)](#).

Default = true

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.sse.tls.required=<boolean>
# hsctl config apply all
# hsctl service restart s3 --nodes=ALL
```

Note If you disable TLS on the S3 Service's local listeners (by setting [s3.tls.enabled](#) to *false*) you must also set *s3.sse.tls.required* to *false*. Otherwise incoming SSE-C requests will fail, since *s3.sse.tls.required=true* sets a requirement for end-to-end TLS usage for SSE-C requests and this requirement cannot be met if TLS is disabled on the local S3 Service instances on your HyperStore nodes.

s3.stackSize

Memory stack size to use for the S3 Service.

If this setting is not explicitly set, the S3 Service uses the memory stack size specified by the **"java.stackSize"** (page 797) setting.

Default = not set (so uses *java.stackSize* value)

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.stackSize=<string>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.threads.idleTimeout

When the S3 Service processes HTTP(S) requests from clients, the maximum allowed connection idle time in milliseconds. If this much idle time passes before a new request is received over an open connection with a client, or if this much idle time passes during the reading of headers and content for a request, or if this much idle time passes during the writing of headers and content of a response, the connection with the client is closed.

Default = 60000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.threads.idleTimeout=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.threads.max

[Former name in HyperStore 7.x and earlier: *cloudian_s3_max_threads*]

On each node, the maximum number of threads to allow in the thread pool that the S3 Service uses to process incoming HTTP(S) requests from clients. When there is a new HTTP(S) request processing task and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.threads.max=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

Note A HyperStore performance optimization script automatically assigns this setting a value appropriate to your local hardware environment. If you use *hsctl config set* to change this setting, the value that you give to the setting will override the script-assigned value for the setting.

s3.threads.min

On each node, the minimum number of threads to keep in the thread pool that the S3 Service uses to process incoming HTTP(S) requests from clients.

The S3 Service uses a thread pool to process incoming HTTP(S) requests from clients. An initial pool of threads is created at server initialization time, and additional threads may be added as needed (within the limit set by **"s3.threads.max"** (page 832)). Idle threads in the pool are terminated if not used within a thread timeout period — unless the number of threads in the pool is down to *s3.threads.min*. If only this many threads are in the pool, then threads are kept open even if they've been idle for longer than the thread timeout.

Default = 100

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set s3.threads.min=<integer>
# hscctl config apply s3
# hscctl service restart s3 --nodes=ALL
```

s3.tiering.endpoints.*

[Former name in HyperStore 7.x and earlier: *cmc_bucket_tiering_default_destination_list*]

Two types of auto-tiering destinations need to be configured with *s3.tiering.endpoints.** settings:

- Tiering destinations that you want the CMC to display to end users as tiering destination options in the Bucket Lifecycle Policy interface.

Note The *s3.tiering.endpoints.** settings determine what displays for users in the CMC only if **"cmc.tiering.enabled"** (page 731) and **"cmc.tiering.allowPerBucketCredentials"** (page 729) are both set to *true*. If *cmc.tiering.allowPerBucketCredentials* is set to *false* then the CMC's Bucket Lifecycle Policy interface will display only the tiering destination that you specify with the **"cmc.tiering.defaultEndpointURL"** (page 730) setting.

- Custom S3 tiering destinations that you want your users to be able to tier to -- using either the CMC or a third party S3 application -- for which the destination endpoint URL does not comply with the standard S3 endpoint format of either *s3-<region>.<domain>* or *s3.<region>.<domain>*. This might include, for example, an external HyperStore system if the endpoint URL that administrators have assigned to that system does not comply with the standard S3 endpoint format.

By default the system has the following *s3.tiering.endpoints.** settings:

s3.tiering.endpoints.* setting	Default Value
s3.tiering.endpoints.AWS DEEP ARCHIVE.name	AWS DEEP ARCHIVE
s3.tiering.endpoints.AWS DEEP ARCHIVE.protocol	s3deeparchive
s3.tiering.endpoints.AWS DEEP ARCHIVE.showInCMC	true
s3.tiering.endpoints.AWS DEEP ARCHIVE.url	https://s3.amazonaws.com
s3.tiering.endpoints.AWS GLACIER.name	AWS GLACIER
s3.tiering.endpoints.AWS GLACIER.protocol	glacier
s3.tiering.endpoints.AWS GLACIER.showInCMC	true
s3.tiering.endpoints.AWS GLACIER.url	https://s3.amazonaws.com
s3.tiering.endpoints.AWS S3.name	AWS S3

s3.tiering.endpoints.* setting	Default Value
s3.tiering.endpoints.AWS S3.protocol	s3
s3.tiering.endpoints.AWS S3.showInCMC	true
s3.tiering.endpoints.AWS S3.url	https://s3.amazonaws.com
s3.tiering.endpoints.Azure.name	Azure
s3.tiering.endpoints.Azure.protocol	azure
s3.tiering.endpoints.Azure.showInCMC	true
s3.tiering.endpoints.Azure.url	https://blob.core.windows.net
s3.tiering.endpoints.Google.name	Google
s3.tiering.endpoints.Google.protocol	s3
s3.tiering.endpoints.Google.showInCMC	true
s3.tiering.endpoints.Google.url	https://storage.googleapis.com

The settings for each tiering endpoint are:

- *name* -- The name of the tiering destination. This name string is what will display to CMC users (if *showInCMC* is set to *true*) in a listing of tiering destination options in the Bucket Lifecycle Policy interface. Also, this name string determines the destination name segment in the setting names for this endpoint. For example, if you were to set *s3.tiering.endpoints.AWS GLACIER.name* to *Glacier*, then *Glacier* would replace *AWS GLACIER* in the names of each of the four settings for that tiering destination (for instance *s3.tiering.endpoints.AWS GLACIER.protocol* would become *s3.tiering.endpoints.Glacier.protocol*).
- *protocol* -- The protocol used by the tiering destination: must be one of *s3* or *glacier* or *s3deeparchive* or *azure* or *spectra*.
- *showinCMC* -- Whether to show this destination as a tiering destination option in the CMC's Bucket Lifecycle Policy interface, *true* or *false*.
- *url* -- The destination URL (including the *https://* segment, or less typically *http://*).

To **change any of the endpoint settings listed in the table above** and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set <setting>=<value>
# hscctl config apply s3 cmc
# hscctl service restart s3 cmc --nodes=ALL
```

Note that:

- If the setting name has a space in it you must enclose the setting name in quotes.
- If your desired setting value has a space (or a non-alphanumeric character) in it you must enclose the value in quotes.

For an example with a space-containing setting name:

```
# hscctl config set "s3.tiering.endpoints.AWS GLACIER.showInCMC"=false
# hscctl config apply s3 cmc
# hscctl service restart s3 cmc --nodes=ALL
```

To **add a new tiering destination**, run the following *hscctl* command on the Config Controller node:

```
# hsctl config create s3.tiering.endpoints <destinationName>
```

For example, to add a new tiering destination named *HyperStoreRemote*:

```
# hsctl config create s3.tiering.endpoints HyperStoreRemote
```

You can then retrieve the default *s3.tiering.endpoints.<destinationName>.** settings for that new destination. For example:

```
# hsctl config get s3.tiering.endpoints.HyperStoreRemote
HYPERSTORE OPTION                                VALUE
s3.tiering.endpoints.HyperStoreRemote.name        HyperStoreRemote
s3.tiering.endpoints.HyperStoreRemote.protocol     s3
s3.tiering.endpoints.HyperStoreRemote.region       s3
s3.tiering.endpoints.HyperStoreRemote.showInCMC    false
s3.tiering.endpoints.HyperStoreRemote.url          https://s3-region.example.domain
```

You then can modify the individual settings for that new destination as needed. At minimum you must set the *.url* value for that destination; and if the protocol is *s3* and the URL does not follow the standard *s3-<region>.<-domain>* or *s3.<region>.<domain>* format then you must also set the *.region* value (specifying the target service region at the destination system). Optionally you can change the *.protocol* and *.showInCMC* settings if you want a value different than the defaults.

```
# hsctl config set s3.tiering.endpoints.HyperStoreRemote.url="https://boston.company.com"
# hsctl config set s3.tiering.endpoints.HyperStoreRemote.region=boston
# hsctl config apply s3 cmc
# hsctl service restart s3 cmc --nodes=ALL
```

Here is a second example, creating a Spectra BlackPearl tiering destination and configuring that destination to display as one of the tiering destination options for end users in the CMC. First create the destination:

```
# hsctl config create s3.tiering.endpoints "Spectra BlackPearl"
```

Then retrieve the default settings for that new tiering destination:

```
# hsctl config get "s3.tiering.endpoints.Spectra BlackPearl"
HYPERSTORE OPTION                                VALUE
s3.tiering.endpoints.Spectra BlackPearl.name        Spectra BlackPearl
s3.tiering.endpoints.Spectra BlackPearl.protocol     s3
s3.tiering.endpoints.Spectra BlackPearl.region       s3
s3.tiering.endpoints.Spectra BlackPearl.showInCMC    false
s3.tiering.endpoints.Spectra BlackPearl.url          https://s3-region.example.domain
```

Then modify the *.protocol*, *.showInCMC*, and *.url* settings for that new tiering destination:

```
# hsctl config set "s3.tiering.endpoints.Spectra BlackPearl.protocol=spectra"
# hsctl config set "s3.tiering.endpoints.Spectra BlackPearl.showInCMC=true"
# hsctl config set "s3.tiering.endpoints.Spectra BlackPearl.url=https://bplab.spectrallogic.com"
# hsctl config apply s3 cmc
# hsctl service restart s3 cmc --nodes=ALL
```

If you wish you could also create and configure a second Spectra BlackPearl tiering destination (named for example "Spectra BlackPearl 2").

Note The default AWS S3 URL, AWS GLACIER URL, and AWS DEEP ARCHIVE URL configured with the *s3.tiering.endpoints* settings -- *https://s3.amazonaws.com* -- is for the default AWS service region, which is *us-east-1*. So, with those default S3 and Glacier and Glacier Deep Archive destinations displaying in the CMC interface, a user can tier to an existing bucket that they own in the AWS *us-east-1*

region, or they can have HyperStore create a tiering target bucket in that region. If you want users to be able to tier to a different AWS region you can add a tiering endpoint for that regional destination, with an AWS region-specific URL such as `https://s3.us-west-2.amazonaws.com`.

s3.tiering.forwardDeletes

[Former name in HyperStore 7.x and earlier: `cloudian.s3.deleteins3`]

This setting applies to HyperStore's [auto-tiering](#) behavior when the tiering destination has native S3 API support, such as AWS, Google Cloud Platform (GCP), or a remote HyperStore system.

If `s3.tiering.forwardDeletes` is set to `true`, then:

- For an object that has been transitioned to the tiering destination system, if a delete is performed on the object in the HyperStore source bucket then the object's metadata will be deleted from the local HyperStore system and **the tiered object itself will be deleted from the tiering destination bucket**.
- If a source bucket that has been configured for auto-tiering is deleted from the local HyperStore system, the **tiering destination bucket will be deleted from the tiering destination system**.

Note When HyperStore is deleting a tiered object or a tiering destination bucket from the tiering destination system, this operation must successfully complete before HyperStore deletes the corresponding local object metadata or local bucket.

If `s3.tiering.forwardDeletes` is set to `false`, then:

- For an object that has been transitioned to the tiering destination system, if a delete is performed on the object in the HyperStore source bucket then the object's metadata will be deleted from the local HyperStore system -- and so the object will no longer be accessible through the local HyperStore system -- but the tiered object itself will **not** be deleted from the tiering destination bucket.
- If a source bucket that has been configured for auto-tiering is deleted from the local HyperStore system, the tiering destination bucket will **not** be deleted from the tiering destination system.

Default = true

Note For tiering destinations that do not have native S3 API support -- such as Azure or Spectra Logic -- if a tiered object or a bucket configured for tiering are deleted from the local HyperStore system, the corresponding object or bucket are deleted from the tiering destination system. For these tiering destinations this behavior cannot be modified.

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set s3.tiering.forwardDeletes=<boolean>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.tiering.userAgent

[Former name in HyperStore 7.x and earlier: `cloudian_tiering_useragent`]

User agent string used by HyperStore when it acts as an S3 client for auto-tiering to an external S3 system.

Default = "APN/1.0 Clouddian/1.0 HyperStore/"

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.tiering.userAgent=<string>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.tls.enabled / s3.tls.required

[Former name of *s3.tls.required* in HyperStore 7.x and earlier: *cmc_storageuri_ssl_enabled*]

The **s3.tls.enabled** setting controls whether a local HTTPS listener is enabled for the S3 Service on each node. If enabled (*true*), the local HTTPS listener will be on the port number specified by the "**s3.ports.main.https**" (page 825) setting. If disabled (*false*), then on each node locally only the regular HTTP listener is opened for the S3 Service -- on the port specified by the "**s3.ports.main.http**" (page 825) setting.

Default = true

The **s3.tls.required** setting controls whether clients -- including other HyperStore services such as the CMC -- will be required to use HTTPS when connecting to the S3 Service. Other HyperStore services read this setting to determine whether they must use HTTPS to connect to the S3 Service. Since a load balancer is typically used in front of the S3 Service, these services if required to use HTTPS for S3 Service access will connect to the load balancer port specified by the "**s3.ports.loadBalancer.https**" (page 825) setting.

Default = false

Note that:

- With the default configuration -- *s3.tls.enabled* set to *true* and *s3.tls.required* set to *false* -- on each node the S3 Service's local HTTP listener and HTTPS listener will both be open, and other HyperStore services will access the S3 Service (via the load balancer) using HTTP not HTTPS.
- If you want to use TLS offloading for the S3 Service -- whereby S3 clients use TLS when connecting to the load balancer but the load balancer does not use TLS when connecting to the S3 Service on individual nodes -- set *s3.tls.enabled* to *false* (so that on each node the S3 Service locally opens only a regular HTTP listener) and *s3.tls.required* to *true* (so that other HyperStore services use HTTPS to access the S3 Service via the load balancer).
- If you choose to set *s3.tls.enabled* to *true* and *s3.tls.required* to *true*, then on each node the S3 Service's local HTTP listener will be automatically closed so that only the local HTTPS listener is open.
- These settings apply to the S3 Select Service as well as to the main S3 Service.
- For information about TLS certificate management, see "**Security and Privacy Features**" (page 177).

To change the *s3.tls.enabled* setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.tls.enabled=<boolean>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

To change the *s3.tls.required* setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hscctl config set s3.tls.required=<boolean>
# hscctl config apply all
# hscctl service restart s3 cmc iam --nodes=ALL
```

To change the *s3.tls.enabled* setting and the *s3.tls.required* setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set s3.tls.enabled=<boolean>
# hscctl config set s3.tls.required=<boolean>
# hscctl config apply all
# hscctl service restart s3 cmc iam --nodes=ALL
```

Note If you disable TLS on the S3 Service's local listeners by setting *s3.tls.enabled* to *false*, you must also set "**s3.sse.tls.required**" (page 831) to *false* (by default it is *true*). Otherwise incoming SSE-C requests will fail, since *s3.sse.tls.required=true* sets a requirement for end-to-end TLS usage for SSE-C requests and this requirement cannot be met if TLS is disabled on the local S3 Service instances on your HyperStore nodes.

s3.tls.verifyCA

Whether the S3 Service will require clients connecting over HTTPS to verify the CA that signed the S3 Service's TLS certificate -- *true* or *false*.

Default = *false*

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set s3.tls.verifyCA=<boolean>
# hscctl config apply s3 cmc iam
# hscctl service restart s3 cmc iam --nodes=ALL
```

s3.usage.bucketLevel

[Former name in HyperStore 7.x and earlier: *bucketstats_enabled*]

Whether to enable usage statistics reporting on a per-bucket basis, *true* or *false*. If you set this to *true*, you can subsequently retrieve usage data for a specified bucket or buckets by using the Admin API's *GET/usage* and *POST/usage/bucket* methods. Per-bucket usage statistics will be available only dating back to the point in time that you enabled this feature. Per-bucket usage statistics are not tracked by default and will not be available for time periods prior to when you set this parameter to *true*.

Note that:

- While the "**s3.qos.bucketLevel**" (page 828) setting is for enabling per-bucket stored-bytes and stored-object counters in the system -- so that you can at any time check the current counts for a given bucket - the *s3.usage.bucketLevel* setting is for enabling per-bucket usage tracking so that you can check to see the usage activity that occurred for a given bucket **during a specified period of time**.
- Setting *s3.usage.bucketLevel* to *true* will result in additional metadata being stored in the Metadata DB, and will create additional work for the cron jobs that roll up usage data into hourly, daily, and monthly aggregates.

Default = *false*

To change this setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hsctl config set s3.usage.bucketLevel=<boolean>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.usage.reports.dailyRollupTTL

[Former name in HyperStore 7.x and earlier: *reports.rollupday.ttl*]

Time-to-live for daily roll-up S3 usage data in the Reports keyspace in Cassandra, in seconds. Daily roll-up data will be automatically deleted this many seconds after its creation.

This applies to per-bucket usage data (if you have enabled per-bucket usage tracking) as well as to per-user and per-group usage data.

Default = 5616000 (65 days)

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.usage.reports.dailyRollupTTL=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.usage.reports.hourlyRollupTTL

[Former name in HyperStore 7.x and earlier: *reports.rolluphour.ttl*]

Time-to-live for hourly roll-up S3 usage data in the Reports keyspace in Cassandra, in seconds. Hourly roll-up data will be automatically deleted this many seconds after its creation.

This applies to per-bucket usage data (if you have enabled per-bucket usage tracking) as well as to per-user and per-group usage data.

Default = 5616000 (65 days)

Note The *billing* Admin API (and the CMC's "[Account Activity](#)" (page 427) page) calculates monthly bills for service users by aggregating **hourly** roll-up data. Once hourly data is deleted, you will not be able to use the Admin API (or the CMC) to generate bills for the service period covered by that data. So be sure to have *s3.usage.reports.hourlyRollupTTL* set to a value large enough to accommodate your billing routine.

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set s3.usage.reports.hourlyRollupTTL=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.usage.reports.monthlyRollupTTL

[Former name in HyperStore 7.x and earlier: *reports.rollupmonth.ttl*]

Time-to-live for monthly roll-up S3 usage data in the Reports keyspace in Cassandra, in seconds. Monthly roll-up data will be automatically deleted this many seconds after its creation.

This applies to per-bucket usage data (if you have enabled per-bucket usage tracking) as well as to per-user and per-group usage data.

Default = 15552000 (180 days)

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set s3.usage.reports.monthlyRollupTTL=<integer>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

s3.usage.trackRate

[Former name in HyperStore 7.x and earlier: `cloudian_s3_usagerates_enabled`]

This setting controls whether to have the system maintain counts for number of HTTP requests, number of bytes-in, and number of bytes-out for each user and for each user group. By default this functionality is disabled, and the system only tracks and reports on stored bytes and stored object counts per user and group.

You must set `s3.usage.trackRate` to `true` if you want the system to support usage reports that report on number of HTTP requests (GETs, PUTs, and DELETEs), bytes-in (upload) volume, and bytes-out (download) volume per user and group. This is important if you want to bill or charge-back users or groups based in part on request volume and/or data transfer volume.

Note that enabling this feature does not produce request counts and data transfer counts for user activity that occurred during the period when the feature was disabled. Rather, if you enable this feature then request tracking and data transfer tracking starts from that point forward.

Note This setting applies only to per-user and per-group usage statistic tracking. It does not apply to per-bucket usage tracking. If you have [enabled per-bucket usage tracking](#), then all usage types -- including request rates and data transfer rates -- are available on a per-bucket basis regardless of this setting.

Note If you enable this setting, then the retention period for raw usage data in the Metadata DB is automatically reduced from seven days to one day. This is due to the large amount of raw usage data that can accumulate quickly if request and data transfer usage tracking is enabled. Note that each hour, raw usage data gets automatically rolled into hourly roll-up usage data -- which has a default retention period of 65 days.

Default = false

To change this setting and apply your change, run the following `hsctl` commands on the Config Controller node:

```
# hsctl config set s3.usage.trackRate=<boolean>
# hsctl config apply s3
# hsctl service restart s3 --nodes=ALL
```

Note You also have the option of configuring this setting in the **Usage Tracking Settings** section of the CMC's ["Configuration Settings" \(page 498\)](#) page. In the CMC UI this setting is called "Track/Report Usage for Request Rates and Data Transfer Rates".

s3.user.maxBuckets

[Former name in HyperStore 7.x and earlier: `cloudian_s3_max_user_buckets`]

Maximum number of S3 storage buckets to allow per user. When a user has this many buckets and tries to create an additional bucket, the request is rejected with an error response.

Default = 100

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set s3.user.maxBuckets=<integer>
# hscctl config apply s3
# hscctl service restart s3 --nodes=ALL
```

Note You also have the option of configuring this setting in the **S3 Request Restriction Settings** section of the CMC's **"Configuration Settings"** (page 498) page. In the CMC UI this setting is called "Maximum Buckets Per User".

s3.user.maxCredentials

[Former name in HyperStore 7.x and earlier: `credentials.user.max`]

Maximum allowed number of S3 credentials per HyperStore user. Each credential is a key pair consisting of a public key (access key) and a private key (secret key). These credentials enable a HyperStore user to access the HyperStore S3 storage system through either the CMC or a third party S3 client.

Inactive credentials count toward this maximum as well as active credentials. Credentials can be created, made active or inactive, and deleted, through either the CMC or the Admin API.

Note If a HyperStore user creates IAM users under their HyperStore account and creates S3 credentials for those IAM users, the IAM users' credentials do not count toward the HyperStore user's maximum allowed number of S3 credentials. IAM user credentials are limited separately.

Default = 5

To change this setting and apply your change, run the following `hscctl` commands on the Config Controller node:

```
# hscctl config set s3.user.maxCredentials=<string>
# hscctl config apply s3
# hscctl service restart s3 --nodes=ALL
```

13.2.15. SQS Service Settings

The table below lists configuration settings in the `sqs` setting group, which configures the [SQS Service](#). When applicable the settings in the table are organized into sub-groups.

Setting Sub-Group	Setting	Default Value
sqs.heap	sqs.heap.maxSize (page 842)	1g
	sqs.heap.minSize (page 843)	100m
sqs.logging	sqs.logging.level (page 843)	INFO
	sqs.logging.* rollover settings	see settings description for default values
sqs.ports	sqs.ports.main.http (page 844)	18090

Setting Sub-Group	Setting	Default Value
	sqs.ports.main.https (page 844)	18443
sqs.threads	sqs.threads.idleTimeout (page 845)	60000
	sqs.threads.max (page 845)	150
	sqs.threads.min (page 845)	100
sqs.tls	sqs.tls.enabled / sqs.tls.required (page 846)	false / false
n/a (miscellaneous individual settings)	sqs.dnsName (page 842)	sqs.mycloudianhyperstore.com
	sqs.enabled (page 842)	false
	sqs.lowRes.idleTimeout (page 844)	5000
	sqs.stackSize (page 844)	

13.2.15.1. sqs.* Setting Descriptions

sqs.dnsName

[Former name in HyperStore 7.x and earlier: *sqs_endpoint*]

Service endpoint (fully qualified domain name) for the SQS Service. This endpoint must be resolvable for SQS clients.

Default = Derived from operator input during installation.

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set sqs.dnsName=<string>
# hsctl config apply sqs
# hsctl service restart sqs --nodes=ALL
```

sqs.enabled

[Former name in HyperStore 7.x and earlier: *sqs_enabled*]

This setting determines whether the SQS Service is enabled, *true* or *false*. This also determines whether the SQS Service will start automatically when a host reboots.

Default = false

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set sqs.enabled=<boolean>
# hsctl config apply sqs
# hsctl service restart sqs --nodes=ALL
```

sqs.heap.maxSize

On each node, the memory heap maximum size setting (Java *Xmx* setting) for the SQS Service application.

Default = 1g

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set sqs.heap.maxSize=<string>
# hsctl config apply sqs
# hsctl service restart sqs --nodes=ALL
```

sqs.heap.minSize

On each node, the memory heap minimum size setting (Java *Xms* setting) for the SQS Service application.

Default = 100m

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set sqs.heap.minSize=<string>
# hsctl config apply sqs
# hsctl service restart sqs --nodes=ALL
```

sqs.logging.level

This setting controls the level at which SQS Service application logging will occur. Only messages with a severity level that matches or exceeds your configured logging level will be logged.

Listed in order from the least inclusive logging to the most inclusive logging, the supported logging levels are:

- OFF -- The OFF "level" turns off logging.
- FATAL -- The FATAL level designates very severe error events that will likely lead the application to abort.
- ERROR -- The ERROR level designates error events that may still allow the application to continue running.
- WARN -- The WARN level designates potentially harmful situations.
- INFO -- The INFO level designates informational messages that indicate the progress of the application at coarse-grained level.
- DEBUG -- The DEBUG level designates fine-grained informational events that are most useful to debug an application.
- TRACE -- The TRACE level designates finer-grained informational events than the DEBUG level.

For example, if this setting is set to WARN then only messages with severity levels WARN, ERROR, or FATAL will be logged by the SQS Service.

Default = INFO

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set sqs.logging.level=<string>
# hsctl config apply sqs
# hsctl service restart sqs --nodes=ALL
```

sqs.lowRes.idleTimeout

Special, "low resource" maximum idle time to apply to SQS Service HTTP(S) connections when the number of simultaneous connections to an SQS Service node exceeds 1000. Configured in milliseconds. With this setting, you can have the SQS Service be less tolerant of connection idle time during times of high concurrent usage. (For a general description of idle timer behavior, see ["sqs.threads.idleTimeout"](#) (page 845).)

Default = 5000

To change this setting and apply your change, run the following *hctl* commands on the Config Controller node:

```
# hctl config set sqs.lowRes.idleTimeout=<integer>
# hctl config apply sqs
# hctl service restart sqs --nodes=ALL
```

sqs.ports.main.http

[Former name in HyperStore 7.x and earlier: sqs_port]

On each node, the port on which the SQS Service listens for regular HTTP connections.

Default = 18090

To change this setting and apply your change, run the following *hctl* commands on the Config Controller node:

```
# hctl config set sqs.ports.main.http=<integer>
# hctl config apply all
# hctl service restart sqs firewall s3 --nodes=ALL
```

sqs.ports.main.https

[Former name in HyperStore 7.x and earlier: sqs_secure_port]

On each node, the port on which the SQS Service listens for HTTPS connections.

Default = 18443

To change this setting and apply your change, run the following *hctl* commands on the Config Controller node:

```
# hctl config set sqs.ports.main.https=<integer>
# hctl config apply all
# hctl service restart sqs firewall s3 --nodes=ALL
```

sqs.stackSize

Memory stack size to use for the SQS Service.

If this setting is not explicitly set, the SQS Service uses the memory stack size specified by the ["java.stackSize"](#) (page 797) setting.

Default = not set (so uses *java.stackSize* value)

To change this setting and apply your change, run the following *hctl* commands on the Config Controller node:


```
# hsctl config set sqs.stackSize=<string>
# hsctl config apply sqs
# hsctl service restart sqs --nodes=ALL
```

sqs.threads.idleTimeout

When the SQS Service processes HTTP(S) requests from clients, the maximum allowed connection idle time in milliseconds. If this much idle time passes before a new request is received on an open connection with a client, or if this much idle time passes during the reading of headers and content for a request, or if this much idle time passes during the writing of headers and content of a response, the connection is closed.

Default = 60000

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set sqs.threads.idleTimeout=<integer>
# hsctl config apply sqs
# hsctl service restart sqs --nodes=ALL
```

sqs.threads.max

On each node, the maximum number of threads to allow in the thread pool that the SQS Service uses to process incoming HTTP(S) requests from clients. When there is a new HTTP(S) request processing task and there are no idle threads available in the thread pool:

- If the thread pool has fewer than its maximum allowed number of threads, the thread pool executor will create a new thread for the task to use.
- If the thread pool has its maximum allowed number of threads, the thread pool executor will queue the new task until a thread in the pool becomes available for the task to use.

Default = 150

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hsctl config set sqs.threads.max=<integer>
# hsctl config apply sqs
# hsctl service restart sqs --nodes=ALL
```

sqs.threads.min

On each node, the minimum number of threads to keep in the thread pool that the SQS Service uses to process incoming HTTP(S) requests from clients.

The SQS Service uses a thread pool to process incoming HTTP(S) requests from clients. An initial pool of threads is created at server initialization time, and additional threads may be added as needed (within the limit set by "[sqs.threads.max](#)" (page 845)). Idle threads in the pool are terminated if not used within a thread timeout period — unless the number of threads in the pool is down to *sqs.threads.min*. If only this many threads are in the pool, then threads are kept open even if they've been idle for longer than the thread timeout.

Default = 100

To change this setting and apply your change, run the following *hsctl* commands on the Config Controller node:

```
# hscctl config set sqs.threads.min=<integer>
# hscctl config apply sqs
# hscctl service restart sqs --nodes=ALL
```

sqs.tls.enabled / sqs.tls.required

The **sqs.tls.enabled** setting controls whether a local HTTPS listener is enabled for the SQS Service on each node. If enabled (*true*), the local HTTPS listener will be on the port number specified by the "**sqs.-ports.main.https**" (page 844) setting. If disabled (*false*), then on each node locally only the regular HTTP listener is opened for the SQS Service -- on the port specified by the "**sqs.ports.main.http**" (page 844) setting.

Default = false

The **sqs.tls.required** setting controls whether clients will be required to use HTTPS when connecting to the SQS Service.

Default = false

Note that:

- If you want to use TLS offloading for the SQS Service -- whereby SQS clients use TLS when connecting to a load balancer but the load balancer does not use TLS when connecting to the SQS Service on individual nodes -- set **sqs.tls.enabled** to *false* (so that on each node the SQS Service locally opens only a regular HTTP listener) and **sqs.tls.required** to *true* (so that other HyperStore services use HTTPS to access the SQS Service via the load balancer).
- For information about TLS certificate management, see "**Security and Privacy Features**" (page 177).

To change the **sqs.tls.enabled** setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set sqs.tls.enabled=<boolean>
# hscctl config apply sqs
# hscctl service restart sqs --nodes=ALL
```

To change the **sqs.tls.required** setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set sqs.tls.required=<boolean>
# hscctl config apply all
# hscctl service restart sqs s3 --nodes=ALL
```

To change the **sqs.tls.enabled** setting and the **sqs.tls.required** setting and apply your change, run the following *hscctl* commands on the Config Controller node:

```
# hscctl config set sqs.tls.enabled=<boolean>
# hscctl config set sqs.tls.required=<boolean>
# hscctl config apply all
# hscctl service restart sqs s3 --nodes=ALL
```

13.3. CMC's Configuration Settings Page

Through the CMC's **Configuration Settings** page you can dynamically change a variety of HyperStore system configuration settings. For detail see "**Configuration Settings**" (page 498).

13.4. Installer Advanced Configuration Options

The HyperStore installation tool supports several types of advanced system configurations which can be implemented at any time after initial installation of the system.

Note As a best practice, you should complete basic HyperStore installation first and confirm that the system is working properly (by running the installer's Validation Tests, under the "Cluster Management" menu) before you consider using the installer's advanced configuration options.

To access the advanced configuration options, on the Config Controller node change into your installation staging directory (`/opt/cloudian-staging/8.2`) and launch the installer.

```
# ./cloudianInstall.sh
```

*If you are using the **HyperStore Shell***

If you are using the HyperStore Shell (HSH) as a Trusted user, from any directory on the Config Controller node you can launch the installer with this command:

```
$ hspkg install
```

Once launched, the installer's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

At the installer main menu's Choice prompt enter **4** for Advanced Configuration Options.

```
Cloudian HyperStore(R) 8.0 Installation/Configuration
-----
0 ) Run Pre-Installation checks
1 ) Install Cloudian HyperStore
2 ) Cluster Management
3 ) Upgrade From a Previous Version
4 ) Advanced Configuration Options
5 ) Help
x ) Exit

Choice: █
```

This opens the "Advanced Configuration Options" sub-menu.

```
Advanced Configuration Options
-----

a ) Change server role assignments
b ) Change S3, Admin and CMC ports
c ) Change S3, Admin, CMC, or IAM/STS endpoints
d ) Configure diagnostic data collection options
e ) Configure SSL for Admin, CMC, S3, IAM/STS and SQS services
k ) Enable or disable hsDNS
l ) Configure Performance Parameters on Nodes
m ) Disable the root password
n ) Change CMC Application Name
r ) Exclude node(s) from configuration updates and service restarts
s ) Configure Firewall
t ) Configure 'force' behaviour
x ) Return to Main Menu

Choice: █
```

From this menu you can choose the type of configuration change that you want to make and then proceed through the interactive prompts to specify your desired settings.

a) Change server role assignments

This is for shifting role assignments — such as the QoS DB Master role or Credentials DB Master role — from one of the hosts in your cluster to another. You can also change the list of external NTP servers through this menu option. For more information see ["Change Node Role Assignments"](#) (page 616).

b) Change S3, Admin or CMC ports

This lets you interactively change listening ports for the S3, Admin, and CMC services. For more information see ["Changing S3, Admin, or CMC Listening Ports"](#) (page 858).

c) Change S3, Admin, CMC, or IAM/STS endpoints

This lets you interactively change the S3 service endpoint, S3 static website endpoint, Admin service endpoint, CMC endpoint, or IAM service endpoint. For more information see ["Changing S3, Admin, CMC, or IAM/STS Service Endpoints"](#) (page 859).

d) Configure diagnostic data collection options

This lets you interactively configure Phone Home (Smart Support) settings.

1. After selecting this option from the Advanced Configuration Options menu, follow the prompts to specify your desired settings. The prompts indicate your current settings. At each prompt press Enter to keep the current setting value, or type in a new value. When prompted, choose to apply your changes.
2. Use the `hsctl` command line tool to [restart the S3 Service](#).

e) Configure SSL for Admin, CMC, S3, IAM/STS, and SQS services

These options are for managing SSL certificates in support of using HTTPS for the Admin, CMC, S3, IAM/STS, and/or SQS services. For more information see ["Security and Privacy Features"](#) (page 177).

k) Enable or disable hsDNS

Selecting this option lets you enable or disable hsDNS. It also shows you the current status of hsDNS (enabled or disabled). For more information on hsDNS see:

- ["hsDNS Feature Overview" \(page 139\)](#)
- ["Setting Up hsDNS" \(page 142\)](#)

l) Configure Performance Parameters on Nodes

This lets you run a HyperStore performance configuration optimization script. The script runs automatically when you install your cluster and when you add nodes, so under normal circumstances you should not need to use this installer Advanced Configuration option. For more information see ["Tuning HyperStore Performance Parameters" \(page 860\)](#).

m) Disable the root password

For information on this option see ["Enabling the HSH and Managing HSH Users" \(page 106\)](#).

r) Exclude host(s) from configuration push and service restarts

Use this option if you want to temporarily exclude a particular node or nodes from configuration pushes and service restarts (that you execute via the install or via the *hscctl* tool). One scenario where it would be appropriate to do this is if you have a node that's down and can't be brought back up anytime soon, and in the interim you want to make a configuration change to your system.

Note This "exclude from configuration push and service restarts" status is different than "maintenance mode". The "excluded" status merely excludes a node from the list of nodes that the installer uses when it pushes out configuration changes to the cluster or restarts services across the cluster; it has no effect other than that. (For more information on "maintenance mode" see ["Start Maintenance Mode" \(page 489\)](#)).

If you put a node into "excluded" status, and you later exit the installer, then if you subsequently launch the installer again it will display a message indicating that there is a node in excluded status. In the sample below the node "cloudian-node6" is in this status.

```
# ./cloudianInstall.sh
The following nodes have been excluded for configuration updates and service
restarts: cloudian-node6
Press any key to continue ...
```

When the node is back up again and you are ready to have it again be eligible for configuration pushes and service restarts, return to the "Advanced Configuration Options" → "Exclude host(s) from configuration push and service restarts" function and enter "none" at the prompt.

If you made a system configuration change when a node was down and excluded, then after the node is back up and you've taken the node out of excluded status, do a configuration push and a service restart (whichever service restart is appropriate to the configuration change you made). This will bring the node's configuration up to date with the rest of the cluster.

Note There are a small number of circumstances where the system will **automatically** place a down node into the "excluded" status. One such circumstance is when the system is executing automatic fail-over of the system cronjob host role, in the event that the primary cronjob host goes down. The next time that you launch the installer it will display a message identifying the node that's in "excluded" status.

s) Configure Firewall

This lets you enable and configure the built-in HyperStore firewall, on all the HyperStore nodes in your system. For more information see ["HyperStore Firewall" \(page 850\)](#).

t) Configure 'force' behavior

If you specify the *force* option when running the installer on the command line, the *force* option will "stick" and will be used automatically for any subsequent times the installer is run to install additional nodes (such as when you do an "Add Node" operation via the Cloudian Management Console, which invokes the installer in the background). To turn the *force* option off so that it is no longer automatically used when the installer is run to add more nodes, launch the installer and go to the Advance Configuration Options. Then choose option **t** for **Configure 'force' behavior** and follow the prompts.

Note For information about the *force* option see ["cloudianInstall.sh" \(page 944\)](#).

13.5. Configuration Special Topics

13.5.1. HyperStore Firewall

Subjects covered in this section:

- *Introduction (immediately below)*
- ["Enabling or Disabling the HyperStore Firewall" \(page 851\)](#)
- ["Default Behavior of the HyperStore Firewall When Enabled" \(page 852\)](#)
- ["Customizing the HyperStore Firewall" \(page 853\)](#)
- ["HyperStore Firewall Logging" \(page 856\)](#)

Each HyperStore node includes a built-in HyperStore Firewall that is pre-configured with settings appropriate for a typical HyperStore deployment. In new installations of HyperStore 8.2, the HyperStore Firewall is disabled by default. If you have upgraded to HyperStore 8.2 from a previous HyperStore version, whether or not the HyperStore Firewall is enabled in your system depends on your past upgrade paths and any past actions you may have taken regarding the HyperStore Firewall. For instructions on checking whether the HyperStore Firewall is currently enabled in your system, see ["Enabling or Disabling the HyperStore Firewall" \(page 851\)](#).

Cloudian, Inc. strongly recommends using a firewall to protect sensitive internal services such as Cassandra, Redis, and so on, while allowing access to public services -- particularly in environments where no dedicated internal interface(s) have been specified during HyperStore installation; or the internal, back-end network is not a closed network only available between HyperStore nodes. The pre-configured HyperStore Firewall serves this purpose, if enabled.

Note The instructions in the rest of this section describe how to use the [hsctl](#) command line tool to work with the HyperStore Firewall. You can also perform some limited HyperStore Firewall configuration tasks by using the HyperStore installer's [Advanced Configuration](#) options. However, using the [hsctl](#) tool to configure the HyperStore Firewall is easier and offers a wider array of configuration options than does the installer.

13.5.1.1. Enabling or Disabling the HyperStore Firewall

To check whether the HyperStore Firewall is currently enabled in your system, on the Config Controller node run the following [hsctl](#) command:

```
# hsctl config get firewall.enabled
```

The response will be *true* if the HyperStore Firewall is enabled, or *false* if it's disabled.

Enabling the HyperStore Firewall

The HyperStore Firewall is implemented using the *firewalld* service (with the exception of HyperStore File Service nodes and Search Service nodes, which run on Kubernetes and the HyperStore Firewall is implemented with *Cilium*). If you have already been using a firewall service on your HyperStore object storage nodes, then do the following **before** enabling the HyperStore Firewall:

1. If you have created custom *firewalld* Zone and Service configuration files, make a backup copy of those files for your own retention needs. When you enable the HyperStore Firewall your existing Zone and Service configuration files will be deleted from the */etc/firewalld* directory.
2. Disable your existing firewall service on each object storage node. For example, to disable *firewalld* do the following on each object storage node:

```
# systemctl stop firewalld
# systemctl disable firewalld
```

To enable the HyperStore Firewall, on the Config Controller node run the following [hsctl](#) commands:

```
# hsctl config set firewall.enabled=true
# hsctl config apply firewall
```

No service restart is necessary.

This enables the HyperStore Firewall on all nodes in your HyperStore system, including HyperStore File Service nodes and Search Service nodes if you've deployed these auxiliary services in your system.

Disabling the HyperStore Firewall

Cloudian, Inc. strongly recommends using a firewall to protect sensitive internal services such as Cassandra, Redis, and so on, while allowing access to public services. If you are disabling the HyperStore Firewall, you should do so only temporarily, or if you intend to switch to using a different type of firewall.

To disable the HyperStore Firewall, on the Config Controller node run the following [hsctl](#) commands:

```
# hsctl config set firewall.enabled=false
# hsctl config apply firewall
```

No service restart is necessary.

This disables the HyperStore Firewall on all nodes in your HyperStore system, including HyperStore File Service nodes and Search Service nodes if you've deployed these auxiliary services in your system.

13.5.1.2. Default Behavior of the HyperStore Firewall When Enabled

When the HyperStore Firewall is enabled, the default behavior on each HyperStore node is as follows:

- On all IP interfaces, **all TCP ports will allow inbound traffic originating from other HyperStore nodes and from HyperIQ**. In a multi-DC or multi-region system, this includes inbound traffic originating from HyperStore nodes in other DCs or regions.
- On all IP interfaces, **only the following TCP ports will allow inbound traffic that originates from sources other than HyperStore nodes**:
 - On HyperStore object storage nodes:
 - Admin HTTP service port (18081 by default)
 - Admin HTTPS service port (19443 by default)
 - CMC HTTP service port (8888 by default)
 - CMC HTTPS service port (8443 by default)
 - IAM HTTP service port (16080 by default)
 - IAM HTTPS service port (16443 by default)
 - S3 HTTP service port (80 by default)
 - S3 HTTPS service port (443 by default)
 - S3 PROXY HTTP service port (81)
 - S3 PROXY HTTPS service port (4431)
 - SSH service port (22)
 - SQS HTTP service port (18090 by default)
 - SQS HTTPS service port (18443 by default)
 - On HyperStore File Service nodes:
 - File Service API HTTPS port (443)
 - NFS Service ports (111, 2049, 38465, 38466, 38467, 38468) (UDP and TCP for all these ports)
 - SMB Service ports (137 [UDP], 138 [UDP], 139 [TCP], 445 [TCP])
 - SSH service port (22)

Traffic originating from sources other than HyperStore nodes will be blocked (DROP'd) on all ports other than those listed above.

Note The HyperStore Firewall also allows incoming ICMP traffic originating from sources other than HyperStore nodes.

- On all IP interfaces, **outbound traffic is allowed on all ports**.

When the HyperStore Firewall is enabled, the HyperStore Firewall configuration will automatically adjust to system changes in the following ways:

- If you resize your cluster by adding or removing nodes, or by adding or removing a data center or service region, the HyperStore Firewall accommodates this change automatically. In the case of expanding your cluster, the HyperStore Firewall will be automatically enabled on new nodes, and the HyperStore Firewall on the existing nodes will allow inbound traffic from the new nodes, on any port. In

the case of removing nodes, the HyperStore Firewall on the existing nodes will be updated such that the removed nodes are no longer part of the cluster and can only access the cluster's public services.

Note If the HyperStore Firewall is disabled when you add new nodes to your cluster, the HyperStore Firewall will also be disabled on the new nodes.

- If you change the port number that a particular HyperStore public service uses, the HyperStore Firewall's configuration is automatically adjusted accordingly. After you complete a port change you only need to apply the updated HyperStore Firewall configuration to the cluster. For instructions see ["Changing S3, Admin, or CMC Listening Ports" \(page 858\)](#).

13.5.1.3. Customizing the HyperStore Firewall

As described in the sub-sections that follow, by using *hscctl config* settings you can customize the HyperStore Firewall's behavior in several ways. Note that none of your customizations will have any effect if the HyperStore Firewall is disabled. For your customizations to have effect, you can make the customizations while the HyperStore Firewall is enabled; or you can make customizations while the HyperStore Firewall is disabled, and then [enable it](#).

- ["Changing the Allow/Deny Settings for a Particular Service" \(page 853\)](#)
- ["Creating ACLs to Use in Allow/Deny Settings" \(page 855\)](#)
- ["Creating Custom Services For Firewall Configuration" \(page 855\)](#)
- ["HyperStore Firewall" \(page 850\)](#)

Changing the Allow/Deny Settings for a Particular Service

In the HyperStore Firewall configuration, each of HyperStore's public services has "allow" and "deny" settings. The setting naming format is as follows:

```
firewall.services.<service>.<protocol>.allow
firewall.services.<service>.<protocol>.deny
```

For some services the service name segment includes the suffix ".main" (this is a carry-over of the naming convention for service port configuration settings, where HyperStore for some services needs to be able to distinguish between a port number assignment at a load balancer and a port number assignment on hosts themselves, with the latter being designated as ".main" -- it's at this host level that the HyperStore Firewall operates). For example, for the S3 Service over HTTPS the HyperStore Firewall's "allow" and "deny" configuration settings are named:

```
firewall.services.s3.main.https.allow
firewall.services.s3.main.https.deny
```

The valid options for *.allow* and *.deny* settings are:

- ANY
- empty
- a named ACL (access control list name, referring to a list consisting of one or more IP addresses or subnets -- for instructions on creating ACLs for this purpose see ["Creating ACLs to Use in Allow/Deny Settings" \(page 855\)](#)).

By default, *.allow* settings are set to ALL and *.deny* settings are set to empty. This means that, at the level of the HyperStore Firewall running on each node, the service is available to any client host.

- To set a setting to ALL, it must be all caps *ALL* not *all* or *All*.
- To set a setting to empty, there are three ways that you can do it:
 - `<setting>=` (with nothing after the equal sign)
 - `<setting>=""` (two double-quote marks with no space between)
 - `<setting>=[]` (brackets with no space between; this is how `hsctl config` indicates empty when you get a setting's current value)
- In case of overlap between a *.deny* setting value and an *.allow* setting, the *.deny* setting takes precedence in regard to the overlap. For example you could have *.allow* set to ALL and set *.deny* to an ACL, and the service would be available to all clients except those within the specified ACL (and in fact this is the most typical use of a *.deny* setting).
- Although typically you would apply any HyperStore Firewall customizations that you make to your whole HyperStore system, if you wish you can limit the scope of a customization to a particular region or particular DC, by including `--region=<region>` or `--datacenter=<datacenter>` to the end of your `hsctl config set` command.
- After making any change to the HyperStore Firewall configuration, apply the change with `hsctl config apply firewall`.

For example, this would narrow the range of clients that the HyperStore Firewall allows to access S3 Service through HTTPS, from the default of ALL to an ACL named "goodguys":

```
# hsctl config set firewall.services.s3.main.https.allow=goodguys
# hsctl config apply firewall
```

No service restart is necessary.

The table below lists all supported *.allow* and *.deny* commands (except those that will be available if you configure any custom services as described in ["Creating Custom Services For Firewall Configuration" \(page 855\)](#)).

Setting	Default
firewall.services.admin.main.http.allow	ANY
firewall.services.admin.main.http.deny	[]
firewall.services.admin.main.https.allow	ANY
firewall.services.admin.main.https.deny	[]
firewall.services.cmc.main.http.allow	ANY
firewall.services.cmc.main.http.deny	[]
firewall.services.file.api.https.allow	ANY
firewall.services.file.api.https.deny	[]
firewall.services.file.share.nfs.allow	ANY
firewall.services.file.share.nfs.deny	[]
firewall.services.file.share.smb.allow	ANY
firewall.services.file.share.smb.deny	[]
firewall.services.cmc.main.https.allow	ANY
firewall.services.cmc.main.https.deny	[]
firewall.services.iam.main.http.allow	ANY

Setting	Default
firewall.services.iam.main.http.deny	☐
firewall.services.iam.main.https.allow	ANY
firewall.services.iam.main.https.deny	☐
firewall.services.s3.main.http.allow	ANY
firewall.services.s3.main.http.deny	☐
firewall.services.s3.main.https.allow	ANY
firewall.services.s3.main.https.deny	☐
firewall.services.s3.proxy.http.allow	ANY
firewall.services.s3.proxy.http.deny	☐
firewall.services.s3.proxy.https.allow	ANY
firewall.services.s3.proxy.https.deny	☐
firewall.services.sqs.main.http.allow	ANY
firewall.services.sqs.main.http.deny	☐
firewall.services.sqs.main.https.allow	ANY
firewall.services.sqs.main.https.deny	☐

Creating ACLs to Use in Allow/Deny Settings

If you wish you can create one or more Access Control Lists (ACLs), which you can then use for the *.allow* and *.deny* settings described in the preceding sub-section. To create a new ACL, use this *hsctl config* command:

```
# hsctl config create firewall.acl <acl-name>
```

The *<acl-name>* is a name of your choosing. For example:

```
# hsctl config create firewall.acl goodguys
```

Once you've created the named ACL, you can configure the membership of the ACL, in the form of one or more IP addresses or networks in CIDR notation. If you specify more than one value -- a list -- make it a double-quote enclosed, comma-separated list.

```
# hsctl config set firewall.acl.<acl-name>.addrs=<membership>
```

For example:

```
# hsctl config set firewall.acl.goodguys.addrs="10.1.10.0/24,100.10.10.0/24"
```

Once you've created an ACL and configured its membership, the ACL is ready to use for any of the *.allow* or *.deny* settings associated with particular services. For example:

```
# hsctl config set firewall.services.s3.main.https.allow=goodguys
# hsctl config apply firewall
```

Creating Custom Services For Firewall Configuration

In some environments you may have additional services that you wish to run on HyperStore nodes, such as an SNMP agent. To accommodate such services in the HyperStore Firewall, you can create a configurable "custom" service (for firewall purposes) with this command:

```
# hsctl config create firewall.services.custom <service-name>
```

For example:

```
# hstctl config create firewall.services.custom snmp
```

Once you've created the custom service, you can configure these settings for the service:

```
firewall.services.custom.<service-name>.allow
firewall.services.custom.<service-name>.deny
firewall.services.custom.<service-name>.ports
```

These settings will all default to empty. The *.allow* and *.deny* settings are configurable just like the *.allow* and *.deny* settings for any of the standard HyperStore services. For the *.ports* setting specify the value as *tcp/<port#>* or *udp/<port#>*, or a double-quote enclosed comma-separated list *"tcp/<port#>, udp/<port#>"*. For example:

```
# hstctl config set firewall.services.custom.snmp.allow=goodguys
# hstctl config set firewall.services.custom.snmp.deny=[]
# hstctl config set firewall.services.custom.snmp.ports="tcp/161,udp/161"
# hstctl config apply firewall
```

Note that if you want to apply an ACL for access to the SSH service, you can create a custom service and assign the service a *.ports* value of *tcp/22*:

```
# hstctl config create firewall.services.custom ssh
# hstctl config set firewall.services.custom.ssh.allow=goodguys
# hstctl config set firewall.services.custom.ssh.deny=[]
# hstctl config set firewall.services.custom.ssh.ports=tcp/22
# hstctl config apply firewall
```

Note If you're connected to the Config Controller node via SSH and if you try to assign the SSH service an *.allow* ACL that excludes the host from which you're connecting, the *hstctl config set* operation will fail. This is to prevent a lock-out of the host from which you're connecting.

13.5.1.4. HyperStore Firewall Logging

On each node, requests blocked by the HyperStore Firewall are logged to */var/log/cloudian/firewall.log*. For more information about this log including its rotation and retention behavior, see ["HyperStore Firewall Log" \(page 671\)](#).

13.5.2. Anti-Virus Software

If you are considering using anti-virus software on your HyperStore nodes, be aware that the HyperStore system is provisioned and uses server resources according to the server specifications and use case. Any use of third party software must ensure that it does not interfere with HyperStore's use of these resources (such as disk space, disk I/O, RAM, CPU, network, and ports). Unavailability or sharing of these resources can cause the HyperStore system to not function and/or have reduced performance.

If you want to use anti-virus software to monitor OS files **other than** HyperStore-related files, configure the anti-virus software to exclude these directories from monitoring:

- All HyperStore data directories (as specified by the configuration setting ["hyperstore.directory.data" \(page 766\)](#))
- */var/lib/{cassandra,cassandra_commit,redis}*
- */var/log/{cloudian,cassandra,redis}*
- */opt*

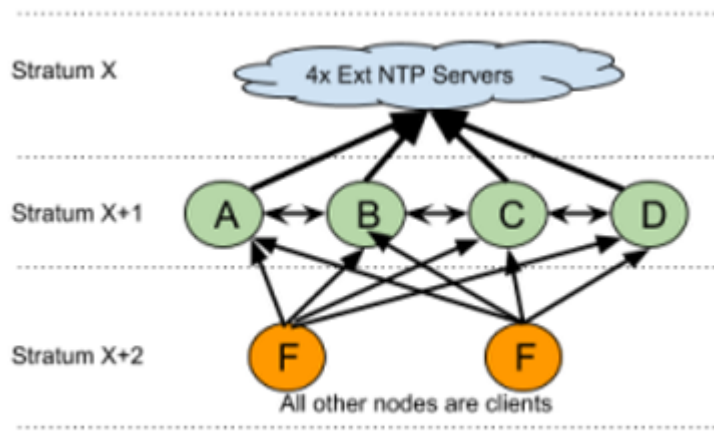
13.5.3. NTP Automatic Set-Up

Note This topic describes the NTP configuration that HyperStore automatically implements. If instead you are using your own custom NTP set-up, Cloudbian recommends using **at least 3 root clock sources**.

Accurate, synchronized time across the cluster is vital to HyperStore service. For example, object versioning relies on it, and so does S3 authorization. It's important to have a robust NTP set-up.

When you install your HyperStore cluster, the installation script **automatically** configures a robust NTP set-up using *chronyd*, as follows:

- In each of your HyperStore data centers, four HyperStore object storage nodes are automatically configured as internal NTP servers. These internal NTP servers will synchronize with external NTP servers -- from the *pool.ntp.org* project by default -- and are also configured as peers of each other.
 - If a data center has four or fewer HyperStore object storage nodes in total, then all the HyperStore object storage nodes in the data center are configured as internal NTP servers.
 - If a data center includes HyperStore File Service nodes or a HyperStore Search Service node, up to two of those nodes will also be automatically configured as internal NTP servers (in addition to the four object storage nodes configured as internal NTP servers).
- If there are more than four HyperStore object storage nodes in a data center, all other nodes are configured as clients of the four internal NTP servers.



- In the event that all four internal NTP servers in a DC are unable to reach any of the external NTP servers, the four internal NTP servers will use "orphan mode" -- which entails the nodes choosing one of themselves to be the "leader" to which the others will sync -- until such time as one or more of the external NTP servers are reachable again.

Each HyperStore data center is independently configured, using this same approach.

To see which of your HyperStore hosts are configured as internal NTP servers, go to the CMC's [Cluster Information](#) page.

On the CMC's **Cluster Information** page you can also view the list of external NTP servers to which the internal NTP servers will synchronize. By default the external NTP servers used are public servers from the *pool.ntp.org* project:

- 0.rocky.pool.ntp.org
- 1.rocky.pool.ntp.org
- 2.rocky.pool.ntp.org
- 3.rocky.pool.ntp.org

IMPORTANT ! In order to connect to the default external NTP servers the internal NTP servers must be allowed outbound internet access.

Note *chronyd* is configured to automatically start on host boot-up. After host boot-up you can confirm that it's running (and retrieve a list of connected time servers) by running *chronyc sources* on the host.

13.5.3.1. Changing the List of External NTP Servers or Internal NTP Servers

You can use the HyperStore installer's "Advanced Configuration Options" to change either the list of external NTP servers or the list of internal NTP servers. For more information see ["Change Internal NTP Servers or External NTP Servers"](#) (page 622).

13.5.4. Changing S3, Admin, or CMC Listening Ports

You can use the HyperStore installer's "Advanced Configuration Options" to change listening ports for the S3, Admin, and CMC services.

1. On the Config Controller node, change to the installation staging directory (*/opt/cloudian-staging/8.2*). Then launch the installer.

```
# ./cloudianInstall.sh
```

*If you are using the **HyperStore Shell***

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on Config Controller node you can launch the installer with this command:

```
$ hspkg install
```

Once launched, the installer's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

2. At the installer's main menu enter **4** for "Advanced Configuration Options". Then at the Advanced Configuration Options menu enter **b** for "Change S3, Admin or CMC ports".
3. Follow the prompts to specify your desired port numbers. The prompts indicate your current settings. At each prompt press Enter to keep the current setting value, or type in a new value. When prompted, choose to apply your changes.
4. Exit the installer. On the Config Controller node, [use the *hsctl* command line tool to restart the affected services](#):
 - For changed S3 or Admin ports, restart the S3 Service and the CMC. Note that the Admin service is restarted automatically when you restart the S3 Service.
 - For changed CMC port, restart the CMC

Note If you have the HyperStore firewall enabled (as described in ["HyperStore Firewall"](#) (page 850)), the firewall's configuration is automatically adjusted to accommodate the port number change that you made. But you must push the updated firewall configuration out to the cluster by running this `hscctl` command on the Config Controller node:

```
hscctl config apply firewall
```

13.5.5. Changing S3, Admin, CMC, or IAM/STS Service Endpoints

You can use the HyperStore installer's "Advanced Configuration Options" to change the S3 service endpoint, S3 static website endpoint, Admin service endpoint, CMC endpoint, or IAM endpoint. (For more information on these HyperStore service endpoints, their default values, and how the endpoints are used, see ["DNS Set-Up"](#) (page 53).)

Note In the current HyperStore release:

- * The CMC uses the IAM service endpoint to make STS calls as well as IAM calls. Therefore the installer's function for changing service endpoints will show one shared service endpoint -- the IAM endpoint -- for IAM and STS.
- * The installer does not support changing the SQS service endpoint. You can change that endpoint by editing the configuration setting `"sqs.dnsName"` (page 842).

1. On the Config Controller node, change to the installation staging directory (`/opt/cloudian-staging/8.2`). Then launch the installer.

```
# ./cloudianInstall.sh
```

*If you are using the **HyperStore Shell***

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Config Controller node you can launch the installer with this command:

```
$ hspkg install
```

Once launched, the installer's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

2. From the installer's menu select "Advanced Configuration Options" and then select "Change S3, Admin, CMC, or IAM/STS endpoints".
3. Follow the prompts to specify your desired endpoints. The prompts indicate your current settings. At each prompt press Enter to keep the current setting value, or type in a new value.

For S3 service endpoint, the typical configuration is one endpoint per service region but you also have the option of specifying multiple endpoints per region (if for example you want to have different S3 service endpoints for different data centers within the same region). To do so simply enter a comma-separated list of endpoints at the prompt for the region's S3 service domain URL. Do not enclose the comma-separated list in quotes. If you want to have different S3 endpoints for different data centers within the same service region, the recommended S3 endpoint syntax is `s3-<region-name>.<dcname>.<domain>`. For example if you have data centers named *chicago* and *cleveland* both within the *midwest* service region, and your domain is *enterprise.com*, the S3 endpoints would be `s3-`

midwest.chicago.enterprise.com and *s3-midwest.cleveland.enterprise.com*. (Make sure that your DNS set-up resolves the service endpoints in the way that you want -- for example, with one S3 service endpoint resolving to the virtual IP address of a load balancer in your Chicago data center and one S3 service endpoint resolving to the virtual IP address of a load balancer in your Cleveland data center).

If you have multiple S3 service endpoints for your system, the first S3 service endpoint in your comma-separated list will be the "default" S3 service endpoint (and so will be used in public URLs if CMC users generate public URLs for some of their objects).

Note

- * In an S3 service endpoint the only instance of the string "s3" should be the leading prefix. Do not also include "s3" in the `<regionname>` value, the `<dcname>` value, or the `<domain>` value . This is because having two instances of "s3" in the service endpoint may cause S3 service requests to fail for some S3 client applications. For example, do not have a service endpoint such as *s3-tokyo.s3.enterprise.com*.
- * HyperStore supports the *s3-<regionname>.<domain>* format and also the *s3.<regionname>.<domain>* format (with a dot after the "s3" rather than a dash).
- * Do not use an IP address as your S3 service endpoint.

For S3 static website endpoint you can only have one endpoint per service region. For the Admin service you can only have one endpoint for your whole HyperStore system. For the CMC service you can only have one endpoint for your whole HyperStore system. For the IAM service you can only have one endpoint for your whole HyperStore system.

When prompted, choose to apply your changes.

4. Exit the installer. On the Config Controller node, [use the *hsctl* command line tool to restart the S3 Service, the CMC, and the IAM Service](#).

Note If you are using your DNS environment for HyperStore service endpoint resolution, **update your DNS entries** to match your custom endpoints if you have not already done so. For guidance see ["DNS Set-Up" \(page 53\)](#) .

Note If you are using per-group filtering of S3 endpoint displays in the CMC and you change an S3 endpoint (using the procedure above), then you must go to the CMC's **Manage Groups** page and edit the group configuration for any groups that are using S3 endpoint display filtering. If you do not update the S3 endpoint display filtering for such groups, then neither the original S3 endpoint nor the replacement S3 endpoint will display for those groups. Note that per-group filtering of S3 endpoint displays is not the default behavior (by default all users can see all of your system's current S3 endpoints, listed in the CMC's **Security Credentials** page). If you did not explicitly configure any groups to use S3 endpoint display filtering, then after changing S3 endpoints in the system you do not need to take any action in regard to the CMC's display of S3 endpoints -- all CMC users will automatically be able to see the new S3 endpoints.

13.5.6. Tuning HyperStore Performance Parameters

The HyperStore system includes a performance configuration optimization script that is automatically run on each node when you install HyperStore; and that also automatically runs on any new nodes that you subsequently add to your cluster. The script adjusts OS configuration settings on each node, and certain

HyperStore system configuration settings, for optimal performance based on your particular environment (taking into account factors such as RAM and CPU specs).

Since the script runs automatically during installation and during cluster expansion, under normal circumstances you should not need to run the script yourself. However, you can run the performance configuration optimization script indirectly through the installer's Advanced Configuration Options menu, if for example you have made configuration changes on your own and your system is now under-performing as a result. Running the script in this way will return the configuration settings to the optimized values as determined by the script.

To run the script from the Advanced Configuration Options menu:

1. On the Config Controller node, change to the installation staging directory (`/opt/cloudian-staging/8.2`) and then launch the installer:

```
# ./cloudianInstall.sh
```

*If you are using the **HyperStore Shell***

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Config Controller node you can launch the installer with this command:

```
$ hspkg install
```

Once launched, the installer's menu options (such as referenced in the steps below) are the same regardless of whether it was launched from the HSH command line or the OS command line.

2. From the installer's menu select "Advanced Configuration Options" and then select "Configure Performance Parameters on Nodes".
3. At the prompt, specify a node for which to run the performance configuration optimization; or specify a comma-separated list of nodes; or leave the prompt blank and press enter if you want to run the optimization for all nodes in your system. When the script run is done the installer interface will prompt you to continue to the next steps. When prompted, choose to apply your changes.
4. Exit the installer. On the Config Controller node, [use the `hscctl` command line tool to restart the S3 Service, the HyperStore Service, and the Cassandra Service](#).

This page left intentionally blank

Chapter 14. Command Line Tools

14.1. `hsstool`

The HyperStore system includes its own cluster management utility called `hsstool`. This tool has functionality that in many respects parallels the Cassandra utility `nodetool`, with the important distinction that `hsstool` applies its operations to the [HyperStore File System \(HSFS\)](#) as well as to the Cassandra storage layer.

The `hsstool` utility is in the `/opt/cloudian/bin` directory of each HyperStore node. Because the HyperStore installation adds `/opt/cloudian/bin` to each host's `$PATH` environment variable, you can run the `hsstool` utility from any directory location on any HyperStore host.

The tool has the following basic syntax:

```
# hsstool -h <host> [-p <port>] <command> [<command-options>]
```

- The `<host>` is the hostname or IP address of the HyperStore node on which to perform the operation. Specify the actual hostname or IP address -- **do not use "localhost"**. For most commands that only retrieve information, the `-h <host>` argument is optional and (if not supplied) defaults to the hostname of the host on which you are executing `hsstool`. For commands that impact system data or processes -- such as a repair or cleanup operation -- the `-h <host>` argument is mandatory.
- The `<port>` is the HyperStore Service's JMX listening port. If you do not supply the port number when using `hsstool`, it defaults to 19082. There is no need to supply the port when using `hsstool` unless you've configured your system to use a non-default port for the HyperStore Service's JMX listener. The syntax summaries and examples in the documentation of individual commands omit the `-p <port>` option.
- The `hsstool` options `-h <host>` and (if you use it) `-p <port>` must precede the `<command>` and the `<command-options>` on the command line. For example, do not have `-h <host>` come after the `<command>`.
- For best results when running `hsstool` on the command line, run the commands as `root`. While some commands may work when run as a non-root user, others will return error responses.

If you are using the HyperStore Shell

If you are using the [HyperStore Shell \(HSH\)](#) you can run `hsstool` from the HSH command line, with the same capabilities and same syntax as if you were running it as `root`:

```
$ hsstool -h <host> [-p <port>] <command> [<command-options>]
```

All `hsstool` operations activity is logged in the [cloudian-hyperstore.log](#) file on the node to which you sent the `hsstool` command.

For usage information type `hsstool help` or `hsstool help <command>`. The usage information that this returns is not nearly as detailed as what's provided in this documentation, but it does provide basic information about syntax and command options.

The table below lists the commands that `hsstool` supports. For command options and usage information, click on a command name.

Note Most `hsstool` commands can be executed either on the command line or through the CMC's [Node Advanced](#) page. The documentation for these commands shows the CMC interface for each command as well as the command line syntax.

Command Type	Command	Purpose
Information Commands	info	View vNode and data load info for a physical node
	ring	View vNode info for whole cluster
	status	View summary status for whole cluster
	opstatus	View status of operations such as cleanup, repair, and rebalance
	proactiverepairq	View status of proactive repair queues
	repairqueue	View auto-repair schedule or enable/disable auto-repair
	ls	View vNode and data load info per mount point
	whereis	View storage location information for an S3 object
	metadata	View metadata for an S3 object
	trmap	View token range snapshot IDs or snapshot content
	madd	This is for internal use by the install script, or for use as directed by Cloudian Support.
Maintenance Commands	cleanup	Clean a node of replicated data that doesn't belong to it
	cleanupec	Clean a node of erasure coded data that doesn't belong to it
	cleanuptombstone	Manage Cassandra tombstone cleanup operations
	autorepair	In the CMC interface this invokes the repairqueue command
	repair	Repair the replicated data on a node
	repairec	Repair the erasure coded data in a data center
	repaircassandra	Repair only the Cassandra metadata on a node, not the object data
	rebalance	Shift data load from your existing nodes to a newly added node
	opctl	List or stop all repair and cleanup operations currently running in the region

14.1.1. hsstool cleanup

Subjects covered in this section:

- ["Command Syntax" \(page 864\)](#)
- ["Usage Notes" \(page 867\)](#)
- ["Checking Operation Status" \(page 869\)](#)

14.1.1.1. Command Syntax

Synopsis

```
# hsstool -h <host> cleanup [allkeyspaces|nokeyspaces] [-n] [-l <true|false>]
[-b] [-x] [-no] [-a] [-c <true|false>] [-d <mountpoint>] [-vnode <token>]
[-policy] [-stop]
```

Options

allkeyspaces | nokeyspaces *Set scope of metadata cleanup*

(Optional) You have three alternatives for choosing which Cassandra metadata keyspaces to clean up, while also cleaning replicated S3 object data in the HyperStore File System (HSFS):

- Use *allkeyspaces* to clean up replicated S3 objects in the HSFS and also clean up all the Cassandra keyspaces. Cassandra cleanup will be completed first, then HSFS replica cleanup. The Cassandra keyspaces that will be cleaned are: `UserData_<storage-policy-ID>` keyspaces; `AccountInfo`; `Reports`; `Monitoring`; and `ECKeyspace`. (For more information see the overview of [Cassandra keyspaces for HyperStore](#))
- Use *nokeyspaces* to clean up only replicated objects in the HSFS, and not any Cassandra keyspaces
- If you specify neither *allkeyspaces* nor *nokeyspaces* then the default behavior is to clean up replicated objects in the HSFS and also to clean the *Cassandra UserData_<storage-policy-ID>* keyspaces (which store object metadata). Cassandra cleanup will be completed first, then HSFS replica cleanup.

-n *Perform dry run without deleting data*

(Optional) Don't actually delete anything. Instead, just do a dry run that identifies the replica data that doesn't belong to the node. If you use the *-n* option you must also:

- Use the *nokeyspaces* option. The ability to do a dry run without actually deleting data is supported only for HSFS replica data. Therefore you must select the *nokeyspaces* option or else the cleanup run will actually clean Cassandra keyspace data.
- Have cleanup operation logging turned on (as it is by default). See the description of the *-l* option below.

Note If you use the *-n* option when you run the *cleanup* command, one of the response items will be "Number of files to be deleted count". The specific objects identified for deletion will be listed in the cleanup log file as in the *-l* option description below.

-l <true|false> *Log list of objects to be cleaned*

(Optional, defaults to true) Write to a log file a list of all the objects that were identified as not belonging to the node. Defaults to true, so you only need to specify the *-l* option if you do **not** want cleanup object logging (in which case you'd specify *-l false*).

If you use logging without using the *-n* option, then the list in the log file is a list of all objects that were deleted by the *cleanup* operation.

If you use logging in combination with the *-n* option, then the list in the log file is a list of HSFS replica objects that will be deleted if you run *cleanup* again without the *-n* option.

The log is named *clouidian-hyperstore-cleanup.log* and is written into the Clouidian HyperStore log directory of the target host. Activity associated with a particular instance of a *cleanup* command run is marked with a unique command number.

-b *Delete bucketless objects*

(Optional) Delete any objects that are not associated with a valid S3 bucket. If you use this option the cleanup operation will check each object to verify that it is part of a bucket, and delete any objects that are not part of a bucket. In typical cleanup scenarios it's not necessary to use this option.

-x *Clean only objects from unowned ranges*

(Optional) Remove only data that belongs to token ranges that the target node is not responsible for. This is the recommended option to use when you are cleaning the other nodes after adding a new node to a cluster. In this circumstance, using the `-x` option makes the cleanup more efficient and faster.

`-no` *Clean only objects that lack metadata*

(Optional) Ignore (do not clean) data that belongs to token ranges that the target node is not responsible for. This setting has the cleanup operation focus on deleting object data for which there is no corresponding object metadata. This is an efficient way to clean up garbage data after you have deleted a very large number of objects (in which case the standard hourly batch delete process may be taking too long to free up disk space). Especially in the case where you have recently added nodes and the cluster is still rebalancing, ignoring out-of-range data during cleanup enables the cleanup operation to more efficiently remove garbage blob data after you have deleted a large number of objects through the S3 interface or the Admin API (*POST bucketops/purge* operation).

Note that the `-no` option is the opposite of the `-x` option, and therefore you cannot use those two options in combination.

Note The CMC interface does not support the `-no` option. This option is supported only on the command line.

`-a` *Also clean erasure coded data*

(Optional) Clean up garbage replica data and then also clean up garbage erasure coded data. When you use the `-a` option, as soon as the *hsstool cleanup* operation completes on a node an *hsstool cleanuppec* operation is automatically run on the node as well. This is a convenient option if you are cleaning a node that is storing both replica data and erasure coded data.

Do **not** use this option if you are using either the `-d <mount-point>` option or the `-vnode <token>` option.

`-c <true|false>` *Confirm that other replicas exist before cleaning an object*

(Optional, defaults to true) If true, then before removing an object replica because it does not belong to the token ranges that the target node is responsible for, the system will first check to make sure that at least one replica of the object exists on the correct endpoint nodes within the cluster. If no other replicas exist, then the out-of-range replica will be left in place on the node that's being cleaned and an ERROR level message will be written to the HyperStore Service application log (which will also result in the triggering of an Alert in the CMC, if you are using the default alert rules).

This safety feature guards against the possibility of a cleanup operation deleting an incorrectly placed replica when no other replicas of the object exist in any of the correct locations within the cluster. The trade-off is that the cleanup operation will take longer if this approach is used.

This safety feature defaults to true, so there's no need to use the `-c` option unless you want to specify `-c false` in order to skip this safety check.

`-d <mount-point>` *Clean only this mountpoint*

(Optional) Clean only the specified HyperStore data mount point (for example */cloudian1*). This option may be useful if you want to delete garbage data from a particular disk, in an effort to free up space on the disk.

`-vnode <token>` *Clean only this token range*

(Optional) Clean only those objects mapped to the specified vNode (identified by its token such as 18315119863185730105557340630830311535). This option may be useful if during a full node cleanup (or a

disk-specific cleanup), the operation failed for a particular vNode. In that case you can then use the `-vnode <token>` option to retry cleaning just that one vNode.

`-policy` *Clean data from deleted storage policies*

(Optional) By default `cleanup` will evaluate and clean only data associated with storage policies that currently exist in your system. It will not evaluate or delete data associated with storage policies that you've deleted from your system. If you want `cleanup` to also delete from the target node **all data associated with storage policies that you've deleted from the system**, use the `-policy` option.

Note Before deleting a storage policy you are required to delete any buckets and objects that are stored under that policy. However, the way that object deletion works in HyperStore is that the system deletes the object metadata immediately but does not delete the actual object data until the next run of an automatic purging process that occurs hourly on each node. Meanwhile, for storage policy deletion, the full deletion of the metadata associated with the policy is implemented by a [daily cron job](#). Depending on when you delete buckets and objects associated with a storage policy and when you delete the policy itself, the timing may be such that the daily storage policy deletion cron job executes before some of the object data associated with the policy gets deleted by the hourly object data purging process. It's this residual "garbage data" that will be detected and removed if you use the `-policy` option when running `cleanup` on a node.

Note The CMC interface does not support the `-policy` option. This option is supported only on the command line.

`-stop` *Stop an in-progress cleanup*

(Optional) Use `hsstool -h <host> cleanup -stop` to stop an in-progress cleanup operation on the specified node.

You can subsequently use the ["hsstool opstatus"](#) (page 882) command to confirm that the cleanup has been stopped (status = TERMINATED) and to see how much cleanup progress had been made before the stop.

If you terminate an in-progress cleanup operation you will not subsequently be able to resume that operation from the point at which it stopped (the `hsstool cleanup` operation does not support a resume option). Instead, when you want to clean the node you can run a regular full cleanup operation.

14.1.1.2. Usage Notes

Use this [hsstool](#) command on a node when you want to identify and delete replica data that does not belong on the node. Broadly, `hsstool cleanup` removes two classes of "garbage" data from a target node:

- Data that belongs to a token range that the target node is no longer responsible for, as a result of a modified token range allocation within the cluster (as occurs when you add a new node).
- Data that should not be on the node even though the data falls within the token ranges that the node is responsible for. This can occur, for example, if data from objects that have been deleted through the S3 interface (or the Admin API's `POST bucketops/purge` operation) has not yet been removed from disk by the hourly batch delete job; or if an object delete request through the S3 interface succeeds for some but not all of the object's replicas.

By default `hsstool cleanup` performs both types of cleanups, but the command supports an `-x` option to perform only the first type and a `-no` option to perform only the second type.

Within the target node, the *hsstool cleanup* operation processes disks in order from the most full disks to the least full disks. By default 10 disks on the target node are processed concurrently (this is configurable by the setting "[hyperstore.threadPool.cleanup.jobs](#)" (page 780)), and so the behavior of cleaning the fullest disks first comes into play only if the target HyperStore host has more than 10 data disks.

By default you can only run *hsstool cleanup* on one node at a time per data center. This limit is configurable by the setting "[hyperstore.cleanup.operations.maxPerDC](#)" (page 766). If you want to raise this limit, consult with Cloudian Support first.

The system will not allow you to run *hsstool cleanup* on a node on which [hsstool repair](#) is currently running.

Note The *hsstool cleanup* operation will only clean objects for which the Last Modified timestamp is older than the interval configured by the setting "[hyperstore.cleanup.objectDelete.gracePeriod](#)" (page 765). By default this interval is one day. So by default no objects with Last Modified timestamps within the past 24 hours will be deleted by *hsstool cleanup*.

When to Use *hsstool cleanup*

The operational procedures during which you would use *hsstool cleanup* are:

- "[Restoring a Node That Has Been Offline](#)" (page 612)
- "[Delete a Storage Policy](#)" (page 544)

Please refer to those procedures for step-by-step instructions, including the proper use of *hsstool cleanup* within the context of the procedure.

You might also use *hsstool cleanup* at the end of the procedure for "[Adding Nodes](#)" (page 579). However, using *hsstool cleanup* at the end of the procedure for Adding Nodes is necessary **only if you do not use one of the options that integrates cleanup tasks into the *rebalance* operation** (the *rebalance -cleanupfile* option or the *rebalance -cleanup* option). For more information on these *rebalance* options, see "[hsstool rebalance](#)" (page 904).

If you do run *hsstool cleanup* at the end of the Adding Nodes procedure, use the *cleanup* options *allkeyspaces*, *-l*, *-x*, *-a*, and *-c*. Note that the *-a* option applies the cleanup operation to erasure coded data as well as replicated data. The system by default only allows you to run cleanup on one node at a time per data center. After cleanup completes on one node, initiate cleanup on a next node, and continue in this way until all of your previously nodes have been cleaned.

CMC Support For This Command

As an alternative to running *hsstool cleanup* on the command line, you can run command through the CMC UI:

The screenshot shows the Cloudian CMC UI. The top navigation bar includes 'Cluster' (highlighted with a red box and a red circle with '1'), 'Alerts', 'Admin', and 'Help'. The 'Nodes' tab is selected in the sub-navigation bar (highlighted with a red box and a red circle with '2'). The 'Advanced' tab is selected in the main content area (highlighted with a red box and a red circle with '3'). The 'Command Type' dropdown is set to 'Maintenance' (circled in red). The 'hsstool Command' dropdown is set to 'cleanup' (circled in red). The 'Target Node' dropdown is set to 'jarvis'. The 'Options' section shows 'default' selected for 'Options', 'Mount Point' is empty, and 'Virtual Node' is empty. The 'EXECUTE' button is visible at the bottom right.

Note If you launch the operation through the CMC UI, you can track the operation progress through the CMC's [Operation Status](#) page. This way of tracking operation progress is not supported if you launch the operation on the command line. However, regardless of how you launch the operation you can periodically check on its progress by using the [hsstool opstatus](#) command.

14.1.1.3. Checking Operation Status

The *cleanup* operation is a long-running operation. Its duration will depend on multiple factors including the amount of data in your system and the number of nodes in your system. For information about checking the status of an in-progress or recently completed *cleanup* operation see "[hsstool opstatus](#)" (page 882).

14.1.2. hsstool cleanupec

Subjects covered in this section:

- "[Command Syntax](#)" (page 869)
- "[Usage Notes](#)" (page 872)
- "[Checking Operation Status](#)" (page 873)

14.1.2.1. Command Syntax

Synopsis

```
# hsstool -h <host> cleanupec [-n] [-l <true|false>] [-b] [-x] [-no]
[-c <true|false>] [-d <mountpoint>] [-vnode <token>] [-policy] [-stop]
```

Options

-n Perform dry run without deleting data

(Optional) Don't actually delete anything. Instead, just do a dry run that identifies the erasure coded data that doesn't belong to the node. If you use the *-n* option you must also have cleanup operation logging turned on

(as it is by default). See the description of the `-l` option below.

Note If you use the `-n` option when you run the `cleanupec` command, one of the response items will be "Number of files to be deleted count". The specific objects identified for deletion will be listed in the cleanup log file as in the `-l` option description below.

`-l <true|false>` *Log list of objects to be cleaned*

(Optional, defaults to true) Write to a log file a list of all the objects that were identified as not belonging to the node. Defaults to true, so you only need to specify the `-l` option if you do **not** want cleanup object logging (in which case you'd specify `-l false`).

If you use logging without using the `-n` option, then the list in the log file is a list of all objects that were deleted by the `cleanupec` operation.

If you use logging in combination with the `-n` option, then the list in the log file is a list of objects that will be deleted if you run `cleanupec` again without the `-n` option.

The log is named `cloudian-hyperstore-cleanup.log` and is written into the Clodian HyperStore log directory of the target host. Activity associated with a particular instance of a `cleanupec` command run is marked with a unique command number.

`-b` *Delete bucketless objects*

(Optional) Delete any objects that are not associated with a valid S3 bucket. If you use this option the cleanup operation will check each object to verify that it is part of a bucket, and delete any objects that are not part of a bucket. In typical cleanup scenarios it's not necessary to use this option.

`-x` *Clean only objects from unowned ranges*

(Optional) Remove only data that belongs to token ranges that the target node is not responsible for. This is the recommended option to use when you are cleaning the other nodes after adding a new node to a cluster. In this circumstance, using the `-x` option makes the cleanup more efficient and faster.

`-no` *Clean only objects that lack metadata*

(Optional) Ignore (do not clean) data that belongs to token ranges that the target node is not responsible for. This setting has the cleanup operation focus on deleting object data for which there is no corresponding object metadata. This is an efficient way to clean up garbage data after you have deleted a very large number of objects (in which case the standard hourly batch delete process may be taking too long to free up disk space). Especially in the case where you have recently added nodes and the cluster is still rebalancing, ignoring out-of-range data during cleanup enables the cleanup operation to more efficiently remove garbage blob data after you have deleted a large number of objects through the S3 interface or the Admin API (`POST bucketops/purge` operation).

Note that the `-no` option is the opposite of the `-x` option, and therefore you cannot use those two options in combination.

Note The CMC interface does not support the `-no` option. This option is supported only on the command line.

`-c <true|false>` *Confirm that other fragments exist before cleaning an object*

(Optional, defaults to true) If true, then before removing an object fragment because it does not belong to the

token ranges that the target node is responsible for, the system will first check to make sure that all $k+m$ fragments of the object exist on the correct endpoint nodes within in the cluster. If fewer than $k+m$ fragments exist, then the out-of-range fragment will be left in place on the node that's being cleaned and an ERROR level message will be written to the HyperStore Service application log (which will also result in the triggering of an Alert in the CMC, if you are using the default alert rules).

This safety feature guards against the possibility of a cleanup operation deleting an incorrectly placed fragment when fewer than $k+m$ fragments of the object exist within the cluster. The trade-off is that the cleanup operation will take longer if this approach is used.

This safety feature defaults to true, so there's no need to use the `-c` option unless you want to specify `-c false` in order to skip this safety check.

`-d <mount-point>` *Clean only this mountpoint*

(Optional) Clean only the specified HyperStore data mount point (for example `/cloudian1`). This option may be useful if you want to delete garbage data from a particular disk, in an effort to free up space on the disk.

`-vnode <token>` *Clean only this token range*

(Optional) Clean only those objects mapped to the specified vNode (identified by its token such as 18315119863185730105557340630830311535). This option may be useful if during a full node cleanup (or a disk-specific cleanup), the operation failed for a particular vNode. In that case you can then use the `-vnode <token>` option to retry cleaning just that one vNode.

`-policy` *Clean data from deleted storage policies*

(Optional) By default *cleanupec* will evaluate and clean only data associated with storage policies that currently exist in your system. It will not evaluate or delete data associated with storage policies that you've deleted from your system. If you want *cleanupec* to also delete from the target node **all data associated with storage policies that you've deleted from the system**, use the `-policy` option.

Note Before deleting a storage policy you are required to delete any buckets and objects that are stored under that policy. However, the way that object deletion works in HyperStore is that the system deletes the object metadata immediately but does not delete the actual object data until the next hourly run of the automatic purging process that occurs hourly on each node. Meanwhile, for storage policy deletion, the full deletion of the metadata associated with the policy is implemented by a [daily cron job](#). Depending on when you delete buckets and objects associated with a storage policy and when you delete the policy itself, the timing may be such that the daily storage policy deletion cron job executes before some of the object data associated with the policy gets deleted by the hourly object data purging process. It's this residual "garbage data" that will be detected and removed if you use the `-policy` option when running *cleanup* on a node.

Note The CMC interface does not support the `-policy` option. This option is supported only on the command line.

`-stop` *Stop an in-progress cleanup*

(Optional) Use `hsstool -h <host> cleanupec -stop` to stop an in-progress *cleanupec* operation on the specified node.

You can subsequently use the **"[hsstool opstatus](#)"** (page 882) command to confirm that the *cleanupec* operation has been stopped (status = TERMINATED) and to see how much progress had been made before the stop.

If you terminate an in-progress *cleanupec* operation you will not subsequently be able to resume that operation from the point at which it stopped (the *hsstool cleanupec* operation does not support a resume option). Instead, when you want to clean the node you can run a regular full *cleanupec* operation.

14.1.2.2. Usage Notes

Note If you want to clean **replica data and also erasure coded data** on a node, use [hsstool cleanup](#) with the **-a** option. If you want to clean **only erasure coded data** on a node, use *hsstool cleanupec* as described below.

Use this [hsstool](#) command on a node when you want to identify and delete erasure coded data that does not belong on the node. Broadly, *hsstool cleanupec* removes two classes of "garbage" data from a target node:

- Data that belongs to a token range that the target node is no longer responsible for, as a result of a modified token range allocation within the cluster (as occurs when you add a new node).
- Data that should not be on the node even though the data falls within the token ranges that the node is responsible for. This can occur, for example, if data from objects that have been deleted through the S3 interface (or the Admin API's *POST bucketops/purge* operation) has not yet been removed from disk by the hourly batch delete job; or if an object delete request through the S3 interface succeeds for some but not all of the object's replicas.

By default *hsstool cleanupec* performs both types of cleanups, but the command supports an **-x** option to perform only the first type and a **-no** option to perform only the second type.

Within the target node, the *hsstool cleanupec* operation processes disks in order from the most full disks to the least full disks. By default 10 disks on the target node are processed concurrently (this is configurable by the setting **"[hyperstore.threadPool.cleanup.jobs](#)"** (page 780)), and so the behavior of cleaning the fullest disks first comes into play only if the target HyperStore host has more than 10 data disks.

By default you can only run *hsstool cleanupec* on one node at a time per data center. This limit is configurable by the setting **"[hyperstore.cleanup.operations.maxPerDC](#)"** (page 766). If you want to raise this limit, consult with Cloudbian Support first.

The system will not allow you to run *hsstool cleanupec* on a node on which [hsstool repairec](#) is currently running.

Note that:

- The *hsstool cleanupec* operation will only clean objects for which the Last Modified timestamp is older than the interval configured by the setting **"[hyperstore.cleanup.objectDelete.gracePeriod](#)"** (page 765). By default this interval is one day. So by default no objects with Last Modified timestamps within the past 24 hours will be deleted by *hsstool cleanupec*.
- HyperStore 7.5.1 and later supports Hybrid Storage Policies whereby objects larger than a **[configurable size threshold](#)** are erasure coded and objects at or smaller than that threshold are replicated. The *hsstool cleanupec* operation will evaluate and clean the erasure coded objects associated with such storage policies, but not the replicated objects. To evaluate and clean the replicated objects use **"[hsstool cleanup](#)"** (page 864).

- If you want to clean **replica data and erasure coded data** on a node with one command, use [hsstool cleanup](#) with the `-a` option.

When to Use *hsstool cleanup*

If you have erasure coded data in your HyperStore system, the operational procedure during which you would use *hsstool cleanup* are:

- ["Restoring a Node That Has Been Offline"](#) (page 612)
- ["Delete a Storage Policy"](#) (page 544)

Please refer to those procedures for step-by-step instructions, including the proper use of *hsstool cleanup* within the context of the procedure.

CMC Support For This Command

As an alternative to running *hsstool cleanup* on the command line, you can run command through the CMC UI:

The screenshot shows the Cloudian CMC interface. The top navigation bar includes 'Cluster' (highlighted with a red box and a red circle with '1'), 'Alerts', 'Admin', and 'Help'. Below this, the 'Nodes' tab is selected (highlighted with a red box and a red circle with '2'). The 'Advanced' sub-tab is also selected (highlighted with a red box and a red circle with '3'). In the 'Command Type' dropdown, 'Maintenance' is selected. In the 'hsstool Command' dropdown, 'cleanup' is selected. The 'Target Node' dropdown shows 'jarvis'. Under 'Options', the checkboxes for 'n', 'l', 'b', 'x', 'c', and 'stop' are visible, with 'l' and 'c' checked. There are also fields for 'Mount Point' and 'Virtual Node'. A description at the bottom reads: 'Cleanup erasure code HyperStore node data'. An 'EXECUTE' button is located at the bottom right.

Note If you launch the operation through the CMC UI, you can track the operation progress through the CMC's [Operation Status](#) page. This way of tracking operation progress is not supported if you launch the operation on the command line. However, regardless of how you launch the operation you can periodically check on its progress by using the [hsstool opstatus](#) command.

14.1.2.3. Checking Operation Status

The *cleanup* operation is a long-running operation. Its duration will depend on multiple factors including the amount of data in your system and the number of nodes in your system. For information about checking the status of an in-progress or recently completed *cleanup* operation see ["hsstool opstatus"](#) (page 882).

14.1.3. hsstool info

Subjects covered in this section:

- ["Command Syntax" \(page 874\)](#)
- ["Usage Notes" \(page 874\)](#)
- ["Command/Response Example" \(page 874\)](#)

14.1.3.1. Command Syntax

Synopsis

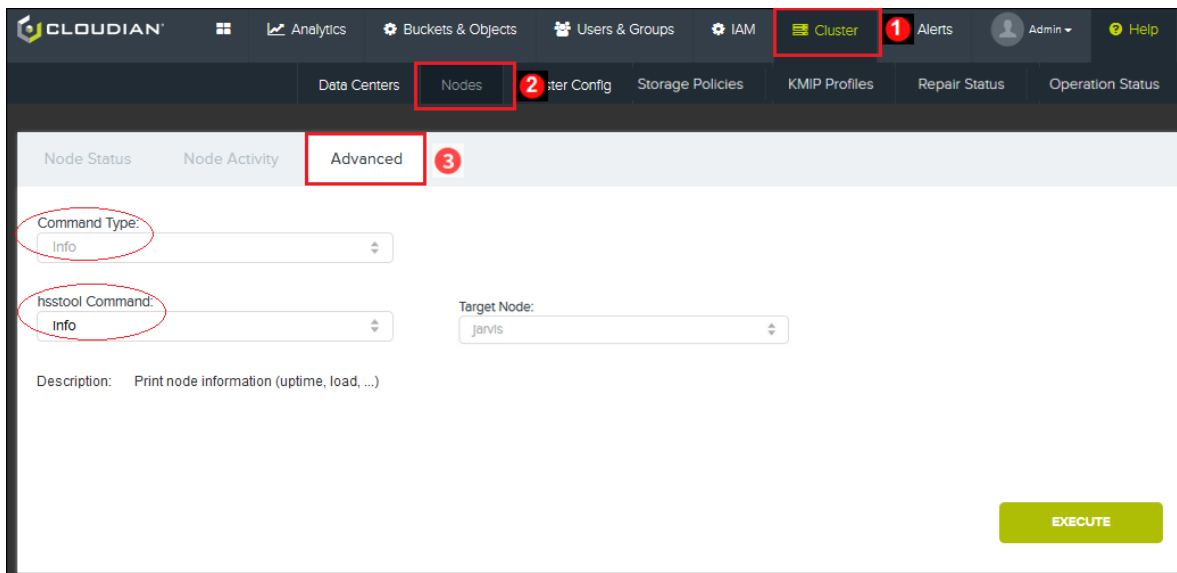
```
# hsstool [-h <host>] info
```

14.1.3.2. Usage Notes

This **hsstool** command returns **virtual node** (token range) information and data load information for a specified physical node within a storage cluster. The return includes a list of virtual nodes (tokens) assigned to the physical node.

CMC Support for This Command

As an alternative to running *hsstool info* on the command line, you can run it through the CMC UI:



14.1.3.3. Command/Response Example

The example below shows an excerpt from a response to the *info* command. The command returns information about a specific node, "cloudian-node1". The node's token (vNode) list is sorted in ascending order.

```
# hsstool -h cloudian-node1 info
Cloudian           : 7.5
Cloudian Load      : 1.12 TB / 42.3 TB (2.6%)
Uptime (seconds)   : 230681
Token              : 18315119863185730105557340630830311535
Token              : 21637484670727122681279388562251225465
Token              : 23572420191725176386844252744138398599
Token              : 25984083804572739863688602357781003456
Token              : 32049925251885239737462386844023262134
```

```

Token          : 34776961444994655981702644433932691872
Token          : 39011904510130716900391258282509705889
...
...
Token          : 141833557733030600282377220263378688424
Cassandra      : 2.0.11
Cassandra Load : 1.9 MB
Data Center    : DC1
Rack           : RAC1

```

Note To see which tokens are on which disks on a node, use [hsstool ls](#).

Response Items

Cloudian

Cloudian HyperStore software version installed on the node.

Cloudian Load

The total volume of S3 object data (replicas and/or erasure coded fragments) stored in the HyperStore File System on the node, across all HyperStore data disks combined.

This field also shows the total volume of disk space allocated for S3 object storage on the node (the total capacity of HyperStore data disks combined); and the percentage of used volume over total capacity.

Uptime

The number of seconds that the HyperStore Service has been running since its last start.

Token

A storage token assigned to the node. Tokens are randomly generated from an integer token space ranging 0 to $2^{127} - 1$, and distributed around the cluster. Each token is the top of a token range that constitutes a virtual node (vNode). Each vNode's token range spans from the next-lower token (exclusive) in the cluster up to its own token (inclusive). A physical node's set of tokens/vNodes determines which S3 object data will be stored on the physical node.

For more background information see ["How vNodes Work"](#) (page 83).

Cassandra

Cassandra software version installed on the node.

Cassandra Load

Cassandra storage load (quantity of data stored in Cassandra) on the node. There will be some Cassandra load even if all S3 object data is stored in the HyperStore File System. For example, Cassandra is used for storage of object metadata and service usage data, among other things.

Data Center

Data center in which the node resides.

Rack

Rack in which the node resides.

14.1.4. `hsstool ls`

Subjects covered in this section:

- ["Command Syntax" \(page 876\)](#)
- ["Usage Notes" \(page 876\)](#)
- ["Command/Response Example " \(page 876\)](#)

14.1.4.1. Command Syntax

Synopsis

```
# hsstool [-h <host>] ls
```

14.1.4.2. Usage Notes

This [hsstool](#) command returns a node's list of HyperStore data mount points, the list of storage tokens currently assigned to each mount point on the node, the current disk usage per mount point, and the recent errors per mount point if any.

CMC Support for This Command

As an alternative to running `hsstool ls` on the command line, you can run it through the CMC UI:

The screenshot shows the Cloudian CMC interface. The top navigation bar includes 'Cluster' (highlighted with a red box and a red circle with '1'), 'Alerts' (with a red circle with '1'), 'Admin', and 'Help'. The main navigation bar has 'Data Centers', 'Nodes' (highlighted with a red box and a red circle with '2'), 'Cluster Config', 'Storage Policies', 'KMIP Profiles', 'Repair Status', and 'Operation Status'. The 'Nodes' section has three tabs: 'Node Status', 'Node Activity', and 'Advanced' (highlighted with a red box and a red circle with '3'). The 'Advanced' tab contains a 'Command Type' dropdown set to 'Info' (circled in red), an 'hsstool Command' dropdown set to 'ls' (circled in red), and a 'Target Node' dropdown set to 'Jarvis'. Below these is a 'Description' field with the text 'List storage mount points and virtual nodes assigned to them'. An 'EXECUTE' button is located at the bottom right of the form.

14.1.4.3. Command/Response Example

The example below shows a response to an `hsstool ls` command. The command response snippet below is truncated; the actual response would list all the tokens assigned to each HyperStore data directory mount point on the node.

Filesystem	Size	Used	Avail	Use%	Mounted on	Status	Estimated Digest Keys	Disk Health	..
/dev/md126	3.5T	101G	3.2T	4%	/	OK	0	NORMAL	..

/dev/sda1	8.9T	9.9G	8.9T	1%	/cloudian1	OK	1877	NORMAL	..
/dev/sdab1	8.9T	9.9G	8.9T	1%	/cloudian3	OK	1879	NORMAL	..
...									
...									
/cloudian1/hsfs:									
	12449505273998224519582214417366908928					HfkGaHQIEduYOZDCRLzW4			
	37348191303441015132019860096080150528					r1D1xZzjR37DQC01SVBC4			
...									
...									

Response Items

Filesystem

Device name of the disk drive

Size

Total capacity of the disk

Used

Used capacity of the disk

All

Remaining available capacity of the disk

Use%

Percentage of disk capacity used

Mounted on

Mount point of the device

Status

Disk status: either OK or ERROR or DISABLED. For more information on the disk's status see the CMC's **Node Status** page, specifically the **Disk Detail Info** section.

Estimated Digest Keys

This is a rough estimate of the number of digest keys in the Digest DB for the disk. This corresponds to the number of HyperStore data files on the disk.

Disk Health

Disk health status

Disk errors

Count of recent disk errors, if any. These counts are categorized into filesystem errors, I/O errors, and file not found errors. The counts are reset whenever any of these events occur:

- A successful access operation on the disk (such as a read or write)
- A 30 minute interval passes without 100 total errors occurring for the disk. Specifically, the counters reset upon the next disk access operation after the interval passes. In broad terms, by default the coun-

ters reset approximately every 30 minutes if not enough disk errors occurred during that interval to trigger the system's configurable automated disk failure response.

- A restart of the node or a restart of the HyperStore Service on the node

Token list

For each HyperStore data mount point, the lower section of the `ls` command response lists all the storage tokens currently assigned to that mount point. Displayed alongside the decimal version of each storage token is the base62 encoding of the token. In the HyperStore File System, base62 encoded tokens will be part of the directory structure for stored S3 object data. For example, under directory `<mount-point>/hsfs/<base62-encoded-tokenX>/...` would be the S3 object replica data associated with the token range for which `tokenX` is the upper bound. For more information on S3 storage directory structure see ["HyperStore Service and the HSFS" \(page 66\)](#).

14.1.5. hsstool metadata

Subjects covered in this section:

- ["Command Syntax" \(page 878\)](#)
- ["Usage Notes" \(page 879\)](#)
- ["Command/Response Example" \(page 879\)](#)

14.1.5.1. Command Syntax

Synopsis

```
# hsstool [-h <host>] metadata <bucket>/<object> [-v <version>]
```

Options

`<bucket>/<object>` *Bucket and object name*

(Mandatory) Bucket name, followed by a forward slash, followed by the full object name (including "folder path", if any). For example, `mybucket/file1.txt` or `mybucket/Videos/Vacation/Italy_2024-06-27.mpg`.

If the object name has spaces in it, enclose the `bucket/object` name pair in quotes. For example, `"mybucket/big document.doc"`.

The `bucket/object` name is case-sensitive.

Note In the CMC UI implementation of this command, you enter the bucket name and the full object name (including folder path) in separate fields. For example, bucket name `mybucket` and full object name `Videos/Vacation/Italy_2024-06-27.mpg`.

`-v <version>` *Object version*

(Optional) Version ID of the object, if versioning has been used for the object. Versions are identified by `timeuuid` values in hexadecimal format (for example, `"fe1be647-5f3b-e87f-b433-180373cf31f5"`). If versioning has been used for the object but you do not specify a version number in this field, the operation returns metadata for the most recent version of the object.

14.1.5.2. Usage Notes

This [hsstool](#) command returns metadata for a specified S3 object, such as the object size and the date-time that the object was last accessed by an S3 client application.

CMC Support for This Command

As an alternative to running *hsstool metadata* on the command line, you can run it through the CMC UI:

The screenshot shows the Cloudian CMC interface. The top navigation bar includes 'Cluster' (highlighted with a red box and a red circle with '1'), 'Alerts', 'Admin', and 'Help'. The 'Nodes' tab is selected in the sub-navigation bar (highlighted with a red box and a red circle with '2'). The 'Advanced' tab is selected in the main content area (highlighted with a red box and a red circle with '3'). The 'Command Type' dropdown is set to 'Info' (circled in red). The 'hsstool Command' dropdown is set to 'metadata' (circled in red). The 'Target Node' dropdown is set to 'jarvis'. The 'Bucket' and 'Object' fields are empty. The 'Version(optional)' field is empty. A description at the bottom states: 'Display metadata information of a given object'. An 'EXECUTE' button is at the bottom right.

14.1.5.3. Command/Response Example

The *metadata* command example below returns the metadata for the specified object.

```
# hsstool -h cloudian-node1 metadata hsfbsn/so
Key: hsfbsn/so
Policy ID: 43c3e277945403c98e5b8f9441d85e16
Version: null
Compression: NONE
Create Time: 2024-07-02T06:16:54.681Z
Last Modified: 2024-07-02T06:16:54.681Z
Last Access Time: 2024-07-02T06:16:54.681Z
Digest: 7bcec86b667ff2be2982f637b67e4942
Size: 1048576
Region: region1

CLOUDIAN_METADATA metadata: {"Path": "hsfbsn/so", "Type": "FILE", "Etag":
"7bcec86b667ff2be2982f637b67e4942", "Locale": null, "GroupId": "CloudianTest1", "UserId":
"77600d5e7dfb7e0f46a198fbded86fe3", "CreatorId": "77600d5e7dfb7e0f46a198fbded86fe3",
"ttl": null, "CreateTime": "2024-07-02T06:16:54.681Z", "ModifyTime": null, "ContentType":
null, "HttpHeaders": null, "Size": 1048576, "UserMetadata": null, "Part": 0, "Acl": null,
"Version": null, "DeleteMarker": false, "Uri": "file://", "PartSize": 10485760, "UploadId":
null, "PartInfo": null, "Policy": null, "PublicUrl": null, "HyperStoreVersion": "4",
"BlobVersion": null, "TotalSize": "1048578", "websiteIndex": null, "websiteError": null,
"websiteRedirect": null, "encryptionKey": null, "encryptionInitVec": null, "Lifecycle": null,
```

```
"Compression": null, "LastAccessTime": null, "TransitionState": null, "Replication": null,
"ReplicationState": null, "Tagging": null, "WriteTime": "1593670614681614929-0A140151",
"DeleteMarkerExpired": null, "TransitionedVersion": null, "EventNotification": null,
"chunkLevelHashValues": null, "AccumulateSize": null, "LockInfo": null}
```

```
CLOUDIAN_OBJMETADATA metadata: {"Path": "hsfsbn/so", "Type": "FILE", "Etag":
"7bcec86b667ff2be2982f637b67e4942", "Locale": null, "GroupId": "CloudianTest1",
"UserId": "user1", "CreatorId": "77600d5e7dfb7e0f46a198fbded86fe3", "Ttl":
"CHUNK_SIZE=10485760", "CreateTime": "2024-07-02T06:16:54.681Z", "ModifyTime": null,
"ContentType": "application/octet-stream", "HttpHeaders": "", "Size": 1048576,
"UserMetadata": "", "Part": 0, "Acl": null, "Version": null, "DeleteMarker": null,
"Uri": "file://", "PartSize": 10485760, "UploadId": null, "PartInfo": null, "Policy":
null, "PublicUrl": null, "HyperStoreVersion": "4", "BlobVersion": null, "TotalSize":
"1048578", "websiteIndex": null, "websiteError": null, "websiteRedirect": null,
"encryptionKey": null, "encryptionInitVec": null, "Lifecycle": null, "Compression":
null, "LastAccessTime": null, "TransitionState": null, "Replication": null,
"ReplicationState": null, "Tagging": null, "WriteTime": "1593670614681614929-0A140151",
"DeleteMarkerExpired": null, "TransitionedVersion": null, "EventNotification": null,
"chunkLevelHashValues": [{"bytes": "{fËkf\u007Fô%)\u0082ô7~IB}"}], "AccumulateSize":
1048576, "LockInfo": null}
```

Response Items

Key

Key that uniquely identifies the S3 object, in format *<bucketname>/<objectname>*.

PolicyID

System-generated identifier of the storage policy that applies to the bucket in which this object is stored.

Version

Object version, if versioning has been used for the object. Versions are identified by *timeuuid* values in hexadecimal format. If versioning has not been used for the object, the Version field displays "Null".

Compression

Compression type applied to the object, if any.

Create Time

Timestamp for the original creation of the object. Format is ISO 8601 and the time is in Coordinated Universal Time (UTC).

Last Modified

Timestamp for last modification of the object. Format is ISO 8601 and time is in UTC.

Last Access Time

Timestamp for last access of the object. 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). Format is ISO 8601 and time is in UTC.

Digest

MD5 digest of the object. This will be a 32 digit hexadecimal number. This digest is used in a variety of

operations including data repair.

Size

The object's size in bytes.

Region

The HyperStore service region in which the object is stored.

CLOUDIAN_METADATA and CLOUDIAN_OBJMETADATA metadata

This is additional raw metadata for the object from the Cassandra CLOUDIAN_METADATA column family (in which object metadata is organized per bucket) and CLOUDIAN_OBJMETADATA column family (in which object metadata is organized per object). This raw object metadata may be useful if you are working Clodian Support to troubleshoot an issue in regard to the object.

Note There is overlap in the content of these two sets of raw object metadata.

14.1.6. hsstool opctl

Subjects covered in this section:

- ["Command Syntax" \(page 881\)](#)
- ["Usage Notes" \(page 881\)](#)
- ["Command/Response Examples" \(page 882\)](#)

14.1.6.1. Command Syntax

Synopsis

```
# hsstool -h <host> opctl [-l] [-stop]
```

Options

-l List all repair and cleanup operations

(Optional) List all in-progress *repair*, *repaipec*, *cleanup*, and *cleanupec* operations in the service region.

-stop Stop all repair and cleanup operations

(Optional) Terminate all in-progress *repair*, *repaipec*, *cleanup*, and *cleanupec* operations in the service region.

14.1.6.2. Usage Notes

This [hsstool](#) command returns a list of all [hsstool repair](#), [hsstool repaipec](#), [hsstool cleanup](#), and [hsstool cleanupec](#) operations **currently running** in a service region. You can also use the command to **stop** all those in-progress operations.

The target *<host>* can be any node in the service region. The command will apply to all nodes in the service region.

With this command you must use either the *-l* option or the *-stop* option ("*hsstool opctl*" by itself doesn't do anything).

Note The `hsstool opctl` command is not supported in the CMC UI.

14.1.6.3. Command/Response Examples

The example below shows the responses to `hsstool opctl` commands. The first command lists the in-progress repair and cleanup operations in the cluster (in this example only a replica repair operation is in progress). The second command stops the in-progress operation(s).

```
# hsstool -h cloudian-node1 opctl -l
10.10.0.184:
REPAIR: rebuild=false,keyspaces=nokeyspaces,max-modified-ts=1526942350092,primary-range=false,
min-modified-ts=0,logging=true,full-repair=true,check-metadata=true,cmdno=1,merkletrue=true,
computedigest=false

# hsstool -h cloudian-node1 opctl -stop
Aborted REPAIR on 10.10.0.184
```

14.1.7. hsstool opstatus

Subjects covered in this section:

- ["Command Syntax" \(page 882\)](#)
- ["Usage Notes" \(page 884\)](#)
- ["Status metrics for cleanup and cleanupec" \(page 885\)](#)
- ["Status metrics for repair and repairec" \(page 888\)](#)
- ["Status metrics for rebalance" \(page 895\)](#)

14.1.7.1. Command Syntax

Synopsis

```
# hsstool [-h <host>] opstatus [<operation>] [-a] [-q history] [-d <operation>]
```

Options

<operation> Operation type

(Optional) Type of operation for which to retrieve status. For example:

```
hsstool -h <host> opstatus repair
```

Valid operation types are listed below. If you do not specify a type, status is returned for all supported operation types.

- cleanup
- cleanupec
- cleanuptombstone

Note For the *cleanuptombstone* operation there is a different command syntax for checking current operation status. See ["Usage Notes" \(page 884\)](#) for more information.

- *decommissionreplicas*
- *decommissionec*
- *proactiverebalance*
- *proactiverepair*
- *proactiverepairec*
- *rebalance*
- *repair*
- *repaircassandra*
- *repairec*

Note that:

- The CMC does not support the "<operation>" option. In the CMC if you run *opstatus* it will return status for all operation types (except for *cleanuptombstone*).
- The *decommission* operation is automatically invoked by the CMC's "Uninstall" feature if you uninstall a node that's "live" in the Cassandra ring. Status reporting on a decommission operation is broken out to *decommissionreplicas* (for replicated data) and *decommissionec* (for erasure coded data).
- The *repaircassandra* operation is invoked not only when ["hsstool repaircassandra" \(page 915\)](#) is explicitly run but also when *hsstool repair* is run either with its default behavior (which includes a repair of user data keyspaces in Cassandra) or with the "allkeyspaces" option (which includes a repair of user data keyspaces and service metadata keyspaces in Cassandra).

-a *Verbose output*

(Optional) Verbose status output for a *repair* or *repairec* or *rebalance* operation. The additional status detail that this option provides can be helpful if you are working with Cludian Support to troubleshoot a repair or rebalance problem.

For example:

```
hsstool -h <host> opstatus repair -a
```

Note The CMC does not support the "-a" option.

-q history *Get operation history*

(Optional) Use *hsstool -h <host> opstatus -q history* to retrieve a history of rebalance, repair, and cleanup options performed on the target node. By default this history spans the past 90 days.

If you want just a history of a particular repair or cleanup operation type, use:

```
hsstool -h <host> opstatus -q history <operation>
```

where <operation> is *rebalance*, *repair*, *repairec*, *cleanup*, or *cleanupec*. For example:

```
hsstool -h <host> opstatus -q history repairec
```

Note The CMC does not support the "-q history" option.

`-d <operation>` Delete operation status

Only use this option if directed to do so by Clouidian Support. This may be needed in the unlikely event that an erroneous or obsolete operation status left over in the Metadata DB is preventing you from initiating a new operation.

14.1.7.2. Usage Notes

This [hsstool](#) command returns the status of the most recent runs of repair, cleanup, rebalance, or de-commission operations that have been performed on a specified node. For each operation type:

- If a run of the operation is **in progress** on the node, then that's the run for which status is returned.
- If the operation is not currently in progress on the node, then status is returned for the **most recent** run of that operation on the node, in the time since the last restart of the node.

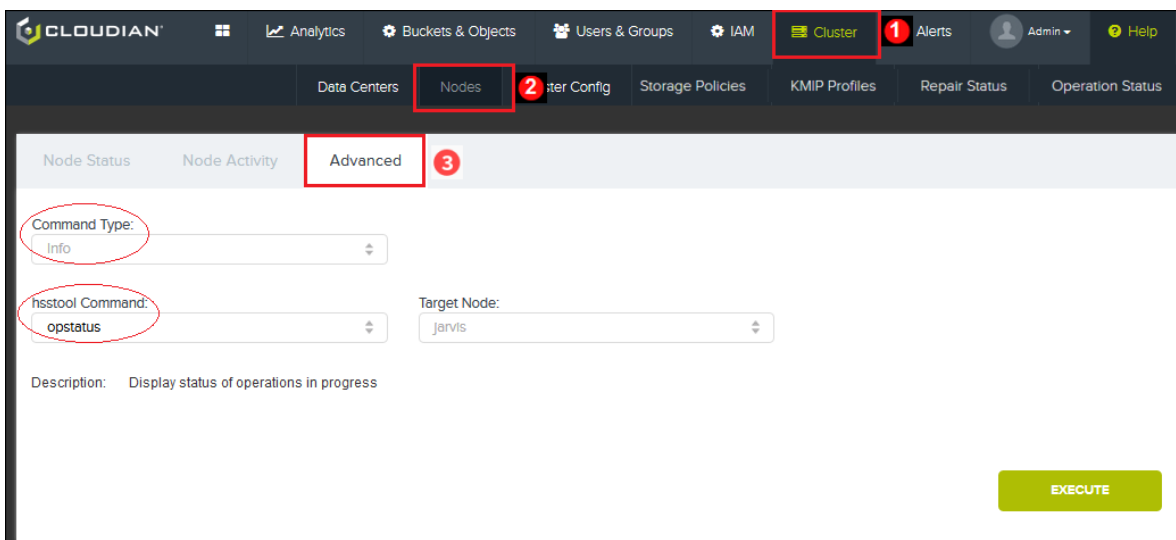
Note To get the current status for the *cleanuptombstone* operation, run *opstatus -q active cleanuptombstone*. For all other operation types simply running *opstatus <operation>* gets the current status.

For checking the status of repair runs other than the most recent run, *opstatus* also supports a command line option to return a **90-day history** of repairs performed on the target node.

Note For operations that you've launched through the CMC UI, a convenient way to check operation status is through the CMC's [Operation Status](#) page, which displays the same output as *opstatus*. This CMC page does not report on operations that you've launched on the command line -- for such operations, explicitly running *hsstool opstatus* is your only option for checking status.

CMC Support for This Command

As an alternative to running *hsstool opstatus* on the command line, you can run it through the CMC UI:



14.1.7.3. Status metrics for *cleanup* and *cleanupec*

The following shows sample output for *opstatus cleanup*. See further below for status metric descriptions. (For more information about the operation type that this is reporting on, see ["hsstool cleanup" \(page 864\)](#)).

```
# hsstool -h cloudian-nodel opstatus cleanup
optype: CLEANUP cmdno#: 1 status: COMPLETED
arguments: deleteobject-without-bucketinfo=false check-protection=true deletedata=true
deleteobject-without-policy=false keyspaces=UserData no-delete-out-of-range=false logging=true
relocate-objects=false cmdno=1 delete-only-outofrange-objects=false
operation ID: 765de07e-8f27-162d-a913-000c2938a438
start: Mon Oct 21 10:11:20 PST 2024
end: Mon Oct 21 10:32:58 PST 2024
duration: 1297.241 sec
progress percentage: 100%
time remaining: 0 ms
cassandra cleanup time: 0.029 sec
task count: 1313587
completed count: 1313587
Number of files deleted count: 10
failed count: 0
skipped count: 1313577
rcvd connection count: 0
open connection count: 0
job threadpool size: 10
message threadpool active: 0
total size of cleanup objects: 100.00 MB
timer: batch.copies.find.timer: count=442 mean=43036.53462127773; batch.fragments.find.timer:
count=0 mean=NaN; batch.cleanup.file.timer: count=442 mean=4.490631563368095E9;
total amount of object data deleted: 102.50 MB
```

The following shows same output for *opstatus cleanupec*. See further below for status metric descriptions. (For more information about the operation type that this is reporting on, see ["hsstool cleanupec" \(page 869\)](#)).

```
# hsstool -h cloudian-nodel opstatus cleanupec
optype: CLEANUPEC cmdno#: 1 status: COMPLETED
arguments: deleteobject-without-bucketinfo=false check-protection=true deletedata=true
deleteobject-without-policy=false no-delete-out-of-range=false logging=true
relocate-objects=false cmdno=1 delete-only-outofrange-objects=false
operation ID: 407d4d16-d08b-1a22-88ed-0a1769c69819
start: Tue Feb 13 03:19:08 UTC 2024
end: Tue Feb 13 03:25:01 UTC 2024
duration: 5 minutes 53 seconds
progress percentage: 100%
time remaining: 0 ms
task count: 200000
completed count: 200000
Number of files deleted count: 10000
failed count: 0
skipped count: 190000
rcvd connection count: 0
open connection count: 0
job threadpool size: 10
message threadpool active: 0
```

```
total size of cleanup objects: 2.44 GB
timer:  batch.copies.find.timer: count=0 mean=NaN; batch.fragments.find.timer: count=68
mean=85069.69420885573; batch.cleanup.file.timer: count=68 mean=5.230084050540501E9;
total amount of object data deleted: 221.00 MB
```

Metric Descriptions for *cleanup* and *cleanupec*

Unless otherwise noted these metrics apply to both *cleanup* and *cleanupec*.

optype

The type of *hsstool* operation.

cmdno#

Command number of the run. Each run of a command is assigned a number.

status

Status of the command run: INPROGRESS, COMPLETED, FAILED, or TERMINATED

A COMPLETED status means only that the operation did not error out and prematurely end. It does not mean that the operation succeeded in respect to every object checked by the operation. For high-level information about object cleanup successes and failures (if any), see the other fields in the *cleanup* response.

A FAILED status means that the operation ended prematurely due to errors. For additional status detail see the other fields in the *cleanup* response. For details on any FAILED operation you can also scan *cloudian-hyperstore.log* for error messages from the period during which the operation was running.

A TERMINATED status means that the cleanup run was terminated by an operator, using *cleanup -stop*, or by some other interruption such as the HyperStore Service doing down on the target node or the target node being rebooted.

arguments

Value of the command arguments used for the run, if any. The status results use internal system names for the arguments which may not exactly match the command-line arguments that are defined in a command's syntax, but the relationships should be clear.

operation ID

Globally unique identifier of the *cleanup* run. This may be useful if Cloudian Support is helping you troubleshoot a repair failure.

Note The "cmd#" (described further above) cannot serve as a globally unique identifier because that counter resets to zero -- and subsequently starts to increment again -- when the HyperStore Service is restarted.

start

Start time of the operation.

end, duration

For a completed operation, the end time and duration of the operation.

progress percentage

Of the total work that the operation has identified as needing to be done, the percentage of work that has been

completed so far.

time remaining

For an in-progress operation, the estimated time remaining.

estimated completion time

For an in-progress operation, the estimated completion time.

cassandra cleanup time (cleanup only)

The time spent cleaning metadata in Cassandra (the Metadata DB). This is applicable only to a *cleanup* operation, not a *cleanupec* operation.

task count

The number of object replicas (in the case of *cleanup*) or erasure code fragments (in the case of *cleanupec*) on the node, which the operation must evaluate to determine whether they correctly belong on the node.

completed count

The number of object replicas (in the case of *cleanup*) or erasure code fragments (in the case of *cleanupec*) that the operation evaluated to determine whether they correctly belong on the node.

Number of files deleted count

The number of object replicas (in the case of *cleanup*) or erasure code fragments (in the case of *cleanupec*) that the operation successfully deleted from the node because they don't belong on the node.

Note If you use the *-n* option when you run the *cleanup* command, this response item will be "Number of files to be deleted count" rather than "Number of files deleted count".

failed count

The number of object replicas (in the case of *cleanup*) or erasure code fragments (in the case of *cleanupec*) that the operation tried to delete (because they don't belong on the node), but failed.

skipped count

The number of object replicas (in the case of *cleanup*) or erasure code fragments (in the case of *cleanupec*) that the operation left on the node because they belong on the node.

rcvd connection count

In support of implementing the operation, the current number of open TCP connections incoming to the target node (the node on which you ran the *cleanup* or *cleanupec* command) from the other nodes in the cluster.

open connection count

In support of implementing the operation, the current number of open TCP connections outgoing from the target node (the node on which you ran the *cleanup* or *cleanupec* command) to the other nodes in the cluster.

job threadpool size

The size of the job threadpool for the operation. This determines how many cleanup 'jobs' can run concurrently on a node. For more information about how *cleanup* and *cleanupec* work is divided into high-level 'jobs' see the description of the configuration setting "[hyperstore.threadPool.cleanup.jobs](#)" (page 780) .

message threadpool active

In support of implementing the operation, the current number of active threads managing communications between the target node (the node on which you ran the *cleanup* or *cleanupeccommand*) and the other nodes in the cluster.

timer

This shows detailed timing metrics for various parts of the operation. These timing metrics may be useful if Cloudbian Support is working with you to troubleshoot *cleanup* or *cleanupeccommand* performance issues in your environment.

total amount of object data deleted

The total amount of object data deleted from the target node (the node on which you ran the *cleanup* or *cleanupeccommand*) as a result of the operation.

In the case of a *cleanup* operation (cleaning replica data from the target node), for a hypothetical object that is 10MB in size, if one of that object's replicas is cleaned from the target node, that's 10MB removed from the node (since each replica of a 10MB object is 10MB).

In the case of a *cleanupeccommand* operation (cleaning erasure coded data from the target node), for a hypothetical object that is 10MB in size -- and supposing a 4+2 erasure coding storage policy is being used -- if one of that object's erasure coded fragments is cleaned from the target node, that's 2.5MB removed from the node (since each erasure coded fragment of a 10MB object is 2.5MB if 4+2 erasure coding is being used).

14.1.7.4. Status metrics for *repair* and *repairec*

The following shows sample output for *opstatus repair*. See further below for status metric descriptions. (For more information about the operation type that this is reporting on, see ["hsstool repair"](#) (page 907)).

```
# hsstool -h cloudbian-node1 opstatus repair
optype: REPAIR cmdno#: 2 status: COMPLETED
operation ID: 765d84e0-8f27-162d-a913-000c2938a438
start: Thu Feb 23 05:58:09 PST 2023
end: Thu Feb 23 05:58:11 PST 2023
duration: 1 second
progress percentage: 100%
time remaining: 0 ms
arguments: rebuild=false keyspaces=UserData max-modified-ts=1677160689935 primary-range=false
min-modified-ts=0 logging=true full-repair=true check-metadata=true cmdno=2 merkle=true
computedigest=false
total range count: 6
executed range count: 6
failed range count: 0
keyspace count: 1
repair file count: 2
failed count: 0
skipped count: 0
repaired count: 2
pr queued count: 0
completed count: 23
total bytes streamed: 305.93 KiB
scan time: 0 ms
```

```
metadata time: 0 ms
stream time: 0 ms
```

The following shows sample output for *opstatus repairec*. See further below for status metric descriptions. (For more information about the operation type that this is reporting on, see ["hsstool repairec" \(page 918\)](#)).

```
# hsstool -h cloudian-node1 opstatus repairec
optype: REPAIREC cmdno#: 1 status: COMPLETED
arguments: mountPoint=null logging=true range=null cmdno=1 computedigest=false oldRepairec=false
operation ID: f06d79e2-af53-1ef7-858a-0a749a422815
start: Thu Jul 11 09:04:26 UTC 2024
end: Thu Jul 11 10:36:50 UTC 2024
duration: 1 hour 32 minutes 23 seconds
progress percentage: 100%
time remaining: 0 ms
total ranges: 0
object count: 1000018
task count: 1000000
completed count: 1000000
repaired count: 667531
failed count: 0
skipped count: 332469
rcvd connection count: 0
open connection count: 0
message threadpool active: 0
timer: cassandra.iterating.timer: count=1000021 mean=1263.4357607049558; metrics.batch.executing:
count=500 mean=1.4656684591620892E11; metrics.batch.idle: count=500 mean=1.7746020372798523E11;
count.estimator.timer: count=1 mean=5.005086883E9; metrics.batch.waiting: count=500
mean=539844.2383711435;
```

Metric Descriptions for *repair*

optype

The type of *hsstool* operation.

cmdno#

Command number of the run. Each run of a command is assigned a number.

status

Status of the command run: INPROGRESS, COMPLETED, FAILED, or TERMINATED.

A COMPLETED status means only that the operation did not error out and prematurely end. It does not mean that the operation succeeded in respect to every object checked by the operation. For example in the case of repair, a COMPLETED status means that all objects in the scope of the operation were checked to see if they needed repair. It does not mean that all objects determined to need repair were successfully repaired. For high-level information about object repair successes and failures (if any), see the other fields in the *repair* response.

A FAILED status means that the operation ended prematurely due to errors. For additional status detail see the other fields in the *repair* response. For details on any FAILED operation you can also scan *cloudian-hyperstore.log* for error messages from the period during which the operation was running. More details can also be had by running *hsstool -h <host> opstatus repair -a* on the command line.

A `TERMINATED` status means that the repair run was terminated by an operator, using `repair -stop`, or by some other interruption such as the HyperStore Service doing down on the target node or the target node being rebooted.

operation ID

Globally unique identifier of the `repair` run. This may be useful if Cloudbian Support is helping you troubleshoot a repair failure.

Note The "cmd#" (described further above) cannot serve as a globally unique identifier because that counter resets to zero -- and subsequently starts to increment again -- when the HyperStore Service is restarted.

start

Start time of the operation.

end, duration

End time and duration of a completed operation.

progress percentage

Of the total work that the operation has identified as needing to be done, the percentage of work that has been completed so far.

time remaining

Approximate time remaining for the operation. This is a rough estimate.

arguments

Value of the command arguments used for the run, if any. The status results use internal system names for the arguments which may not exactly match the command-line arguments that are defined in a command's syntax, but the relationships should be clear. For example, `hsstool repair` command-line syntax supports a `-pr` option, and within "arguments" response item the use or non-use of this option is indicated as `"primary-range=true"` or `"primary-range=false"`.

total range count

The total number of token ranges that will be repaired during this repair operation. The repair operation will encompass not only the token ranges assigned to the target repair node but also certain token ranges on other nodes in the cluster. These are token ranges in which are stored replicas of the same objects that are on the target repair node. As a simplified example, if objects residing in token range "c to d" on the target repair node are also replicated in ranges "d to e" and "e to f" on other nodes in the cluster, then ranges "d to e" and "e to f" will be among the ranges repaired by the repair operation (as well as "c to d"). Consequently the total number of ranges to be repaired will be larger than the number of tokens assigned to the node that is the target of the repair.

The exception is if the `-pr` option was used when the `hsstool repair` operation was executed, in which case the repair operation addresses only the target node's "primary ranges". In this case the "total range count" value will equal the number of tokens assigned to the node.

Note For repair status detail for each token range, run `hsstool -h <host> opstatus repair -a`. This detailed, per-range status information can be helpful if you are working with Cloudbian Support to troubleshoot repair problems.

executed range count

For each of the token ranges in the "total range count" metric, the repair thread pool schedules a sub-job. The "executed range count" metric indicates the number of sub-jobs scheduled. This does not necessarily mean that all these ranges were successfully repaired -- only that the per-range sub-jobs were scheduled by the thread pool.

failed range count

The number of token ranges for which the range repair sub-job did not run successfully. For example if range repair sub-jobs are scheduled by the thread pool but then you subsequently terminate the *repair* operation (using the *-stop* option) or reboot the node, this will result in some failed ranges. Problems with the endpoint nodes or the disks impacted by repair of particular token ranges are other factors that may result in failed ranges.

For information about such failures, on the target node for the repair you can scan `/var/log/cloudian/cloudian-hyperstore-repair.log` for the time period during which the repair operation was running. Running `hsstool -h <host> opstatus repair -a` on the command line will also provide useful details about repair failures.

keyspace count

Number of Cassandra keyspaces repaired. With the default repair behavior this will equal the number of storage policies that are in your HyperStore system. There is one Cassandra `UserData_<policyid>` keyspace for each storage policy. This is where object metadata is stored.

repair file count

Of all the replica files evaluated by the repair operation, this is the number of files that were determined to be in need of repair. This figure may include files on other nodes as well as files on the target repair node. For example, if an object is correct on the target node but one of the object's replicas on a different node is missing and needs repair, then that counts as one toward the repair file count. For a second example, if two of an object's three replicas are found to be out-dated, that counts as two toward the "repair file count".

failed count

Of the files that were found to be in need of repair, the number of files for which the attempted repair failed. For information about such failures, on the target node for the repair you can scan `/var/log/cloudian/cloudian-hyperstore-repair.log` for the time period during which the repair operation was running. Running `hsstool -h <host> opstatus repair -a` on the command line will also provide useful details about repair failures.

If possible, files for which the repair attempt fails are added to the proactive repair queue (see "pr queued count" below).

The "repaired count" plus the "failed count" plus the "pr queued count" should equal the "repair file count".

Note One thing that can increment the "failed count" is if the operation entails writing data to a disk that is in a [stop-write](#) condition (which by default occurs when a disk is 90% full). Such write attempts will fail.

skipped count

The number of replica files that were evaluated and determined not to need repair.

repaired count

Of the files that were found to be in need of repair, the number of files for which the repair succeeded. The "repaired count" plus the "failed count" plus the "pr queued count" should equal the "repair file count".

pr queued count

The number of files that the *hsstool repair* operation adds to the proactive repair queue, to be fixed by the next proactive repair run. If the *hsstool repair* operation fails to repair a file that requires repair, it adds the file to the proactive repair queue. Proactive repair is a different type of repair operation and may succeed in cases where regular *hsstool repair* failed. By default proactive repair runs every 60 minutes.

The "repaired count" plus the "failed count" plus the "pr queued count" should equal the "repair file count".

completed count

The total number of files that were assessed to see if they were in need of repair. This number reflects replication across the cluster — for example, if an object is supposed to be replicated three times (with one replica on the target repair node and two replicas on other nodes), then repair assessment of that object counts as three files toward the "completed count". Note that "completed" here does not necessarily mean that all object repair attempts succeeded. For more information on success or failure of object repair attempts, see the other status metrics.

The "completed count" should equal the "repair file count" plus the "skipped count".

total bytes streamed

Total number of bytes streamed to the target repair node or other nodes in order to implement repairs. For example, if a 50000 byte object on the target repair node is found to be out-dated, and an up-to-date replica of the object is streamed in from a different node, that counts as 50000 toward "total bytes streamed". As a second example, if a 60000 byte object exists on the target node, and replicas of that object are supposed to exist on two other nodes but are found to be missing, and the repair streams the good object replica from the target repair node to the two other nodes — that counts as 120000 toward "total bytes streamed".

scan time

Total time that it took to scan the file systems and build the Merkle Tree that is used to detect discrepancies in object replicas across nodes.

metadata time

Total time that it took to retrieve from the Metadata DB the metadata needed to perform the operation.

stream time

Total time that it took to stream replicas across nodes, in order to implement needed repairs.

Metric Descriptions for *repairrec*

optype

The type of *hsstool* operation.

cmdno#

Command number of the run. Each run of a command is assigned a number.

status

Status of the command run: INPROGRESS, COMPLETED, FAILED, or TERMINATED.

A COMPLETED status means only that the operation did not error out and prematurely end. It does not mean that the operation succeeded in respect to every object checked by the operation. For example in the case of repair, a COMPLETED status means that all objects in the scope of the operation were checked to see if they needed repair. It does not mean that all objects determined to need repair were successfully repaired. For high-level information about object repair successes and failures (if any), see the other fields in the *repair* response.

A FAILED status means that the operation ended prematurely due to errors. For additional status detail see the other fields in the *repair* response. For details on any FAILED operation you can also scan *cloudian-hyper-store-repair-failure.log* for error messages from the period during which the *repair* operation was running.

A TERMINATED status means that the repair run was terminated by an operator, using *repair -stop*, or by some other interruption such as the HyperStore Service doing down on the target node or the target node being rebooted.

arguments

Value of the command arguments used for the run, if any. The status results use internal system names for the arguments which may not exactly match the command-line arguments that are defined in a command's syntax, but the relationships should be clear. For example, *hsstool repair* command-line syntax supports a "-pr" option, and within "arguments" response item the use or non-use of this option is indicated as "primary-range=true" or "primary-range=false".

operation ID

Globally unique identifier of the *repair* run. This may be useful if Clouidian Support is helping you troubleshoot a repair failure.

Note The "cmd#" (described further above) cannot serve as a globally unique identifier because that counter resets to zero -- and subsequently starts to increment again -- when the HyperStore Service is restarted.

start

Start time of the operation.

end, duration

End time and duration of a completed operation.

progress percentage

Of the total work that the operation has identified as needing to be done, the approximate percentage of work that has been completed so far.

time remaining

Estimated time remaining to complete the operation.

total ranges

The total number of token ranges for which data is being evaluated to determine if it needs repair.

task count

In the HyperStore File System, objects are stored as "chunks". Objects smaller than or equal to the chunk size threshold (10MB) are stored as a single chunk. Objects larger than the chunk size threshold are broken into and stored as multiple chunks, with no chunk exceeding the threshold in size. In the case of large objects that S3 client applications upload to HyperStore by the Multipart Upload method, HyperStore breaks the individual parts into chunks if the parts exceed the chunk size threshold. In the context of erasure coding storage policies, after objects (or object parts) are broken into chunks, each object chunk is erasure coded.

In the *repair* operation, the evaluation of a single chunk -- to determine whether all of its erasure coded fragments are present on the nodes on which they should be stored -- constitutes a single "task". For example, the evaluation of a 100MB object that has been broken into 10 chunks -- each of which has been erasure coded using a 4+2 erasure coding scheme -- would count as 10 "tasks", with one task per chunk.

The "task count" metric, then, is the total number of chunks that are being evaluated to determine whether any of them are in need of repair.

completed count, repaired count, failed count, skipped count

The "completed count" is the number of tasks completed so far, from among the tasks in the "task count" (for the definition of a "task" see the description of "task count" above). A "completed" task means that an erasure coded object chunk was evaluated and, if it needs repair, an attempt was made to repair it.

A completed task has one of three possible results: a successful repair, a failed repair attempt, or the determination that the chunk does not need repair (i.e., all of the chunk's fragments are in the proper locations within the cluster). These results are tallied by other *repair* response metrics:

- "repaired count" -- The number of erasure coded object chunks for which a repair was found to be necessary and was successfully executed.
- "failed count" -- The number of erasure coded object chunks for which a repair was found to be necessary, but the repair attempt failed.
- "skipped count" -- The number of erasure coded object chunks that were evaluated and determined not to need repair.

The "repaired count", "failed count", and "skipped count" should add up to equal the "completed count".

Note Notice from the descriptions above that a failed repair attempt counts as a "completed" task. In other words, "completed" in this context does not necessarily mean success. It means only that the *repair* operation has finished its processing of that chunk, resulting in one of the three outcomes described above.

Note If the "completed count" is less than the "task count" this means that the repair was interrupted in such a way that some erasure coded object chunks were identified by a scan of object metadata in Cassandra (and thus counted toward the "task count") but were not yet evaluated or repaired.

To get more detailed metrics about **repair failures** -- in the event that the "failed count" in the *opstatus repair* response is non-zero -- you can run *hsstool -h <host> opstatus repair -a* (the *-a* is the verbose output flag). This will return the same status metrics that are returned by *opstatus repair*, followed by a categorization of repair failures (if any) into various failure types. The response excerpt below is for an operation in which no failures occurred.

```
...
Reason: CONNECTION_ERROR Count: 0
Reason: UNKNOWN Count: 0
Reason: CREATE_TASK_FAILED Count: 0
Reason: EC_DECODE_FAILED Count: 0
Reason: EC_ENCODE_FAILED Count: 0
Reason: REPAIR_TASK_EXPIRED Count: 0
Reason: REPAIR_CYCLE_EXCEEDED Count: 0
Reason: REPAIR_MESSAGE_REQUEST_FAILED Count: 0
Reason: CASSANDRA_CHECK_FAILED Count: 0
Reason: NODE_DOWN Count: 0
```

These same metrics for failure types are logged in *cloudian-hyperstore.log*, as part of the standard logging of *repairec* operations.

Also, the log file *cloudian-hyperstore-repair.log* records an entry for each object that the *repairec* operation was able to repair; and *cloudian-hyperstore-repair-failure.log* records an entry for each object that the *repairec* operation was unable to repair.

For more information about these logs see ["HyperStore Service Logs"](#) (page 671).

rcvd connection count

In support of implementing the repair operation, the current number of open TCP connections incoming to the target node (the node on which you ran the *repairec* command) from the other nodes in the cluster.

open connection count

In support of implementing the repair operation, the current number of open TCP connections outgoing from the target node (the node on which you ran the *repairec* command) to the other nodes in the cluster.

message threadpool active

In support of implementing the repair operation, the current number of active threads managing communications between the target node (the node on which you ran the *repairec* command) and the other nodes in the cluster.

timer

This shows detailed timing metrics for various parts of the *repairec* operation. These metrics may be useful if Cloudian Support is working with you to troubleshoot *repairec* performance issues in your environment.

Note that even more detailed timing information for a completed *repairec* operation is available in the [HyperStore Service application log](#). The timing metrics lines in the log are preceded by a line that says "Performance meters".

14.1.7.5. Status metrics for *rebalance*

The following shows sample output for *opstatus rebalance*. See further below for status metric descriptions. (For more information about the operation type that this is reporting on, see ["hsstool rebalance"](#) (page 904)).

```
# hsstool -h cloudian-node1 opstatus rebalance
optype: REBALANCE cmdno#: 1 status: COMPLETED
arguments: cleanup=false logging=false cmdno=1 cleanupFile=true
operation ID: 763d04a2-f8bc-1901-a07c-5254000f88dc
start: Thu Oct 15 09:54:33 CST 2024
end: Thu Oct 15 10:02:43 CST 2024
```

```
duration: 489.969 sec
progress percentage: 100%
time remaining: 0 ms
task count: 2033
completed count: 2033
streamed count: 1283
failed count: 0
skipped count: 0
rcvd connection count: 0
open connection count: 0
prqueued count: 750
job threadpool size: 4
message threadpool active: 0
stream jobs total: 8
stream jobs completed: 8
stream jobs failed: 0
streamed bytes: 1491288352
delete count: /10.20.2.135=186 /10.20.2.172=342 /10.20.2.173=328
/10.20.2.174=319 /10.20.2.136=0 /10.20.2.171=108
```

Metric Descriptions for *rebalance*

optype

The type of *hsstool* operation.

cmdno#

Command number of the run. Each run of a command is assigned a number.

status

Status of the command run: INPROGRESS, COMPLETED, FAILED, or TERMINATED.

A COMPLETED status means only that the operation did not error out and prematurely end. It does not mean that the operation succeeded in respect to every object checked by the operation. For additional status detail see the other fields in the operation status response.

A FAILED status means that the operation ended prematurely due to errors. For additional status detail see the other fields in the operation status response. For details on any FAILED operation you can also scan *cloudian-hyperstore.log* for error messages from the period during which the operation was running.

A TERMINATED status means that the rebalance run was terminated by an operator using *rebalance -stop*, or by some other interruption such as the HyperStore Service doing down on the target node or the target node being rebooted.

arguments

Value of the command arguments used for the run, if any. The status results use internal system names for the arguments which may not exactly match the command-line arguments that are defined in a command's syntax, but the relationships should be clear.

operation ID

Globally unique identifier of the *rebalance* run. This may be useful if Cloudian Support is helping you troubleshoot a rebalance failure. Note that when *rebalance* is run, the REBALANCE part of the response (for replica data) and REBALANCEEC part of the response (for erasure coded data) will both have the same

operation ID.

Note The "cmd#" (described further above) cannot serve as a globally unique identifier because that counter resets to zero -- and subsequently starts to increment again -- when the HyperStore Service is restarted.

start

Start time of the operation.

end, duration

End time and duration of a completed operation.

progress percentage

Of the total work that the operation has identified as needing to be done, the percentage of work that has been completed so far.

time remaining

Estimated time remaining to complete the operation.

task count

From the existing cluster, the total number of files that are evaluated for possible streaming (copying) to the newly added node, based on the newly added node's assigned tokens. Each such file constitutes a "task".

completed count

From the total task count, the number of tasks that have been completed so far -- that is, the number of files that the system has evaluated and if appropriate has attempted to stream to the new node. Each "completed" task results in the incrementing of either the "streamed count", the "failed count", the "skipped count", or the "prqueued count".

At the end of the operation the "completed count" should equal the "task count".

streamed count

The number of files successfully streamed (copied) from the existing nodes to the new node.

failed count

The number of files for which the attempt to stream the file to the new node fails, and the resulting attempt to insert the file stream job into the proactive repairqueue fails also.

Note If the stream attempt for a file fails, but the stream job for that file is successfully added to the proactive repair queue, that file is counted toward the "prqueued count" -- not the "failed count".

For detail about rebalance streaming failures, on the target node for the rebalance operation (the new node) you can scan `/var/log/cloudian/cloudian-hyperstore.log` for error messages from the time period during which the rebalance operation was running.

skipped count

For rebalancing of replicated object data: Replicas of each object to be streamed (copied) to a newly added

node will typically exist on multiple existing nodes. For example, in a 3X replication environment, for a given object that should be streamed to the new node (based on the new node's token ranges), typically a replica file will reside on three of the existing nodes. The evaluation and processing of each such replica file counts towards the "task count" (so, 3 toward the task count in our example). But once a replica is streamed from one existing node to the new node, it doesn't need to be streamed from the other two existing nodes to the new node. On those other two existing nodes the replica file is "skipped".

rcvd connection count

In support of implementing the rebalance operation, the current number of open TCP connections incoming to the target node (the new node on which you ran the *rebalance* command) from the other nodes in the cluster.

open connection count

In support of implementing the rebalance operation, the current number of open TCP connections outgoing from the target node (the new node on which you ran the *rebalance* command) to the other nodes in the cluster.

prqueued count

The number of files for which the attempt to stream the file to the new node fails (even after automatic retries), but the stream job for that file is successfully added to the proactive repair queue. These stream jobs will then be automatically executed by the next run of the hourly proactive repair process.

job threadpool size

The size of the threadpool dedicated to processing "jobs" on the target node (the new node on which you ran the *rebalance* command). The threadpool size determines how many jobs can run concurrently on the new node. For more information about rebalance jobs see the description of "stream jobs total" below.

message threadpool active

In support of implementing the rebalance operation, the current number of active threads managing communications between the target node (the new node on which you ran the *rebalance* command) and the other nodes in the cluster.

stream jobs total

The system breaks the overall rebalance streaming workload down into multiple "jobs", with each job entailing streaming (copying) a subset of the data from an existing node to the new node. This metric shows the total number of jobs into which the rebalance streaming workload has been divided. The total depends on factors such as the number of storage policies in the system, the number of existing nodes in the cluster to which the new node is being added, and the number of token ranges involved in the rebalancing.

Note For rebalance status detail for each individual job, run *hsstool -h <host> opstatus rebalance -a*. This detailed information can be helpful if you are working with Cloudbian Support to troubleshoot rebalance problems.

stream jobs completed

The number of stream jobs completed so far. Note that "completed" means that work on the job has ended and does not necessarily mean success -- if any of the stream jobs ended in failure, they will count toward the "stream jobs failed" metric.

stream jobs failed

The number of stream jobs that failed.

Note If the rebalance operation as a whole has finished with status COMPLETED, but the 'stream jobs failed' metric indicates that some stream jobs failed, you can run `hsstool -h <host> rebalance -retry` to have the system try again to execute those stream jobs. For more information see the description of the [-retry](#) option.

streamed bytes

The number of bytes of object data streamed to the new node.

streamed bytes

The number of bytes of object data streamed to the new node.

delete count

This metric is included in the opstatus response only if you included the "[-cleanupfiles](#)" option when you ran `hsstool rebalance`. It indicates how many files were deleted from each of your existing nodes, after those files were successfully copied to the new node.

14.1.8. hsstool proactiverepairq

Subjects covered in this section:

- "[Command Syntax](#)" (page 899)
- "[Usage Notes](#)" (page 901)
- "[Command/Response Examples](#)" (page 902)

14.1.8.1. Command Syntax

Synopsis

```
# hsstool -h <host> proactiverepairq [-a] [-delete <host>]
[-start [-type replicas|ec]] [-stop] [-enable true|false]
```

Options

-a Show verbose output

(Optional) If you use `hsstool -h <host> proactiverepairq` **without the -a flag** or other options -- the command will return the number of nodes that are in need of proactive repair, the IP addresses of those nodes, and an **estimate** of the number of objects in need of proactive repair on those nodes.

If you use `hsstool -h <host> proactiverepairq -a`, the command will return the number of nodes that are in need of proactive repair and the IP addresses of those nodes, and precise information about the count and total size of objects that are in need of proactive repair on each node. When you use the *-a* option, a scan of Cassandra metadata is done. This is a much more resource intensive operation than if you omit the *-a* option.

Note To see how much proactive repair work **has already been completed** on a given node, use the "[hsstool opstatus](#)" (page 882) command on that node.

-delete <host> Delete a specified node's proactive repair queue

(Optional) Use this option to completely clear out a specified node's current proactive repair queue. If you do

so, the objects that had been in the proactive repair queue will not be fixed by proactive repair, and instead you will need to fix them by running **"[hsstool repair](#)"** (page 907) and **"[hsstool repairec](#)"** (page 918) on the node.

Under normal circumstances you should not need to use the `-delete <host>` option. You might however use this option if you are working with Clouidian Support to troubleshoot problems on a node.

Note The CMC interface does not support the `-delete <host>` option. This option is supported only on the command line.

`-start [-type replicas|ec|rebalance]` *Start proactive repair now*

(Optional) Use `hsstool -h <host> proactiverepairq -start` to immediately initiate proactive repair on the specified host. This applies **only to the specified host**.

Optionally you can restrict the immediate proactive repair to a particular category of proactive repair: proactive repair of replica data for an existing node; proactive repair of erasure coded data for an existing node; or proactive repair of object data streaming failures from a recently completed rebalance operation for a newly added node. For example, use `hsstool -h <host> proactiverepairq -start -type rebalance` to immediately initiate proactive repair on a new host after a rebalance operation that reported failures for some objects.

If you do not include the `-type` option, then using `-start` will immediately initiate proactive repair for all types of proactive repairs that are currently needed the target node.

Note Proactive repair is triggered automatically every hour (by default configuration; see **"[hyperstore.repair.proactive.interval](#)"** (page 773)), on all nodes that are in need of proactive repair, for all proactive repair types. No operator action is required. So there is no need to use the `-start` option unless for some reason you want proactive repair to begin immediately on a particular node rather than waiting for the next automatic hourly run of proactive repair.

`-stop` *Stop proactive repair*

(Optional) Use `hsstool -h <host> proactiverepairq -stop` to immediately stop any in-progress proactive repair on the specified host. This applies **only to the specified host**. Remaining repair tasks that are still in the proactive repair queue for that host will be processed the next time that proactive repair is run.

If proactive repair is in progress on multiple hosts and you want to stop all the proactive repairs, you must submit a `hsstool -h <host> proactiverepairq -stop` command separately for each of those hosts.

Note Unlike the `-start` option, the `-stop` option does not support specification of a proactive repair type. Instead the `-stop` option stops all types of in-progress proactive repair on the target host.

`-enable true|false` *Enable or disable proactive repair throughout the service region*

(Optional) Enable or disable the proactive repair feature. By default the feature is enabled.

If you use `hsstool -h <host> proactiverepairq -enable false` to disable the proactive repair feature, this applies to **all nodes in the service region of the specified host**. So, it doesn't matter which host you specify in the command as long as it's in the right service region. Likewise `hsstool -h <host> proactiverepairq -enable true` re-enables the proactive repair feature for all nodes in the service region.

Disabling the proactive repair feature **does not abort in-progress proactive repairs**. Rather, it prevents any additional proactive repairs from launching. To stop in-progress proactive repairs on a particular node use the `-stop` option.

IMPORTANT ! The proactive repair feature is important for maintaining data integrity in your system. Do not leave it disabled permanently.

Note The proactive repair feature can also be disabled and re-enabled by using `hsstool repairqueue -h <host> -enable true|false`. The difference is that the "[hsstool repairqueue](#)" (page 925) approach disables and re-enables scheduled auto-repairs and also proactive repairs, whereas the `hsstool proactiverepairq` approach disables and re-enables only proactive repair.

If you use both types of commands, then whichever command you used most recently will be operative in regard to the proactive repair feature. For example if you run `hsstool repairqueue -h <host> -enable false` and then subsequently you run `hsstool -h <host> proactiverepairq -enable true`, then proactive repair will be enabled in the cluster (while the scheduled auto-repair feature will remain disabled).

14.1.8.2. Usage Notes

This [hsstool](#) command returns information about nodes that are in need of automated proactive repair. This includes nodes for which automated proactive repair is in progress as well as nodes for which automated proactive repair will begin shortly. You can also use the command to immediately start proactive repair (rather than waiting for the automatic hourly run); or to stop in-progress proactive repairs; or to temporarily disable the proactive repair feature (and to re-enable it after having disabled it).

For retrieving proactive repair queue information (`hsstool -h <host> proactiverepairq`) or for disabling or re-enabling the proactive repair feature (`hsstool -h <host> proactiverepairq -enable true|false`), the `<host>` can be any host in your cluster. The command applies to all nodes in the service region of the host that you specify.

For starting or stopping a proactive repair (`hsstool -h <host> proactiverepairq -start` or `hsstool -h <host> proactiverepairq -stop`), the `<host>` is the host on which you want to start or stop a proactive repair.

For more information about the HyperStore proactive repair feature see "[Automated Data Repair Feature Overview](#)" (page 130).

CMC Support for This Command

As an alternative to using the command line, in the CMC UI you can run the `hsstool proactiverepairq` command's proactive repair queue status reporting function through this interface:

The screenshot shows the Cloudian CMC interface. The top navigation bar includes 'Cluster' (highlighted with a red box and a red circle with '1'), 'Alerts', 'Admin', and 'Help'. The main navigation bar includes 'Data Centers', 'Nodes' (highlighted with a red box and a red circle with '2'), 'Cluster Config', 'Storage Policies', 'KMIP Profiles', 'Repair Status', and 'Operation Status'. The 'Advanced' tab is selected (highlighted with a red box and a red circle with '3'). The 'Command Type' dropdown is set to 'Info'. The 'hsstool Command' dropdown is set to 'proactiverepairqueue'. The 'Target Node' dropdown is set to 'Jarvis'. The 'Options' section shows a checkbox for 'a'. The description reads: 'Display proactive repair queues'. An 'EXECUTE' button is at the bottom right.

The functions for disabling or re-enabling proactive repair, immediately starting a proactive repair, or stopping an in-progress proactive repair have their own separate CMC interface and the command is there renamed as "proactiverepair" (although *hsstool proactiverepairq* is being invoked behind the scenes):

The screenshot shows the Cloudian CMC interface. The top navigation bar includes 'Cluster' (highlighted with a red box and a red circle with '1'), 'Alerts', 'Admin', and 'Help'. The main navigation bar includes 'Data Centers', 'Nodes' (highlighted with a red box and a red circle with '2'), 'Cluster Config', 'Storage Policies', 'KMIP Profiles', 'Repair Status', and 'Operation Status'. The 'Advanced' tab is selected (highlighted with a red box and a red circle with '3'). The 'Command Type' dropdown is set to 'Maintenance'. The 'hsstool Command' dropdown is set to 'proactiverepair'. The 'Target Node' dropdown is set to 'Jarvis'. The 'Options' section shows radio buttons for 'Enable', 'Disable', 'stop', and 'start', with 'Enable' selected. There is also a dropdown for 'All'. The description reads: 'Enable/Disable proactive repair of storage data (cluster-wide setting). Start/Stop proactive repair of storage data on target node. For more information about this setting, please see the online Help.' An 'EXECUTE' button is at the bottom right.

14.1.8.3. Command/Response Examples

The *proactiverepairq* command example below shows that one node in the cluster is in need of proactive repair, and that an estimated 100 objects are queued for proactive repair on that node. This proactive repair occurs automatically; no operator action is required. The repair is either already underway or will be triggered at the next interval (default is hourly).

```
# hsstool -h cloudian1 proactiverepairq
Proactive repair: true
Number of nodes to repair: 1
Estimated number of PR events per node:
cloudian7(10.20.2.57) 100
```

The *proactiverepairq -a* command example below shows that one node in the cluster is in need of proactive repair, and that the needed repair involves eight object replicas totaling about 1.6 MBs. This proactive repair occurs automatically; no operator action is required. The repair is either already underway or will be triggered at the next interval (default is hourly).

```
# hsstool -h cld04 proactiverepairq -a
Proactive repair: true
Number of nodes to repair: 1
cld01(10.20.2.123): REPLICAS[count = 8 bytes = 1602141] EC[count = 0 bytes = 0] REBALANCE[count = 0 bytes = 0]
```

Note The *proactiverepairq -a* option provides more precisely accurate information about the number of queued objects than does the *proactiverepairq* command without the *-a* option -- but using the *-a* option is resource intensive.

Response Items

Proactive repair

This indicates whether the proactive repair feature is enabled, *true* or *false*. By default proactive repair is enabled.

Number of nodes to repair: <#>

Number of nodes that are in need of proactive repair.

Estimated number of PR events per node

This is an **estimate** of the number of objects queued for proactive repair, on each node that currently has objects in its proactive repair queue. This gives you a general idea of the number of queued objects but it is not an exact count.

An exact count is available if you use the *proactiverepairq -a* option, but using that option is resource intensive.

<hostname>(<IPAddress>)

Hostname and IP address of a node that is in need of proactive repair. The results will show one line for each node that needs proactive repair, with each such line starting with the node's IP address.

REPLICAS[count = <#> bytes = <#>]

For the node identified by <IP Address>, the number of object replicas that need to be written to the node by proactive repair, and the total aggregate size of those replicas in bytes.

If the proactive repair is in progress, these numbers indicate how much is left to do.

EC[count = <#> bytes = <#>]

For the node identified by <IP Address>, the number of erasure coded object fragments that need to be written to the node by proactive repair, and the total aggregate size of those replicas in bytes.

If the proactive repair is in progress, this number indicates how much is left to do.

REBALANCE[count = <#> bytes = <#>]

Starting in HyperStore version 7.2, rebalance retries are no longer included automatically within proactive repair runs, so if this appears in the response the count should be 0.

14.1.9. hsstool rebalance

Subjects covered in this section:

- ["Command Syntax" \(page 904\)](#)
- ["Usage Notes" \(page 906\)](#)
- ["Checking Operation Status" \(page 907\)](#)

14.1.9.1. Command Syntax

Synopsis

```
# hsstool -h <host> rebalance [-cleanupfile] [-list] [-l <true|false>] [-stop] [-resume] [-newrun]
```

Options

-cleanupfile Remove from old nodes the data successfully copied to new node

(Optional) When you add a new node to your cluster, the new node takes over some portions of the token space from the existing nodes in the cluster. Based on the new token space allocation, the *rebalance* operation copies certain object replicas and/or erasure coded fragments to the new node, from the existing nodes. Then, having been copied to the new node, replicas and/or fragments can be deleted from existing nodes on which they no longer belong (as a result of some portions of the token space having been taken over by the new node). This "cleanup" action frees up storage space on those existing nodes.

To have the system delete each replica or fragment from the appropriate existing node as soon as it is successfully copied to the new node, use the *-cleanupfile* option when you run *rebalance*. This option also cleans up the corresponding metadata in Cassandra.

Note If you do not use the *-cleanupfile* option when you run *rebalance*, then after the *rebalance* operation completes for the new node -- or after *rebalance* operations complete for all new nodes if you are adding multiple new nodes -- you will need to run [hsstool cleanup](#) on each of the older nodes, one node at a time, in order to remove data that no longer belongs on those nodes and free up storage space on those nodes.

-list List new nodes and their rebalance status

(Optional) If you use this option the command returns a list of newly added nodes in the service region and their status in regard to the rebalance operation.

- A node status of "REQUIRED" means that the node has been added to the cluster (through the CMC's **Add Node** operation) but that you have not yet run *hsstool rebalance* on the new node.
- A node status of "DONE" means that rebalance has been successfully completed for the node.
- A node status of "FAILED" means that the rebalance operation failed for one or more token ranges.

Note This option is supported only on the command line -- not in the CMC UI.

Sample command and response, where one new node has been added to the cluster and rebalance has been successfully completed on the node:

```
[root@vm151 bin]# ./hsstool -h vm118 rebalance -list
vm124 (10.50.41.49):DC1:ca-sm3    DONE
```

The response format is *host(IPAddress):datacenter:regionstatus*

-l <true|false> Log a list of rebalanced objects

(Optional) If you use the *-l true* option, entries for each object rebalanced by this operation will be written to the *cloudian-hyperstore-rebalance.log* file on the newly added node. Object replicas will be logged with the string "REBR" and object erasure coded fragments will be logged with the string "REBEC".

Defaults to false. If you omit the *-l true* option, this verbose logging will not occur.

Note This option is supported only on the command line -- not in the CMC UI.

-stop Stop an in-progress rebalance operation

(Optional) Use *hsstool -h <host> rebalance -stop* to stop an in-progress rebalance on the specified node. The system will keep track of which token ranges have been processed so far, and then when you resume the rebalance operation it will continue with the ranges that have not yet been processed.

After stopping a rebalance, if you wish you can use the **"[hsstool opstatus](#)" (page 882)** command to confirm that the rebalance has been stopped (status = TERMINATED) and to see how much rebalance progress had been made before the stop. **It may take a while for the operation to become TERMINATED**, since the operation will first complete the processing of whichever token range it was working on when you executed the *-stop* option.

When you are ready to resume a rebalance that you stopped, use the *rebalance -resume* option.

The system allows stopping multiple rebalance operations concurrently, on different nodes (by running *hsstool -h <host> rebalance -stop* separately for each node for which you want to stop a rebalance operations). So if you have added multiple nodes to your cluster and are running rebalance operations on the new nodes concurrently, if you wish you can concurrently stop the rebalance operations on each node -- so that no rebalance operations are running in your cluster --and then later, concurrently resume the rebalance operations on each node. (For limitations on rebalance concurrency see **"[Usage Notes](#)" (page 906)**.)

Note During a rebalance operation the system automatically disables the [auto-repair and proactive repair](#) features, and the system will not automatically re-enable these features until all rebalance operations on all new nodes in the service region have been fully completed. So if you stop a rebalance operation, auto-repair and proactive repair will remain disabled during the time that the rebalance operation is stopped. Auto-repair and proactive repair will remain disabled until the rebalance operation has been resumed and completed (and rebalance operations on any other new nodes in the service region have been completed).

-resume Resume a stopped rebalance operation (or retry failed stream jobs)

(Optional) Use *hsstool -h <host> rebalance -resume* to resume a rebalance operation that you previously stopped. The rebalance operation will continue with the token ranges that have not yet been processed.

Another use for this option is if an initial run of *hsstool rebalance* finishes with status COMPLETED but indicates that some "stream jobs" failed. You can run *hsstool -h <host> rebalance -resume* to have the system try again to execute those failed stream jobs. (In earlier releases of HyperStore this function was filled by a *-retry* option, but in HyperStore 7.5.1 and later this function is covered by the *-resume* option.)

When running `hsstool -h <host> rebalance -resume`, you do not need to include any other options in the command (such as the `-cleanupfile` option). Instead, the resume operation will automatically implement the same option(s) that you used in the original rebalance run, if any.

IMPORTANT ! If an `hsstool rebalance` operation finishes with status **FAILED** do not run `hsstool -h <host> rebalance -resume`. A FAILED operation status indicates that the operation errored out and exited prematurely. In this scenario, run `hsstool -h <host> rebalance -newrun`. Do not use the `-resume` option.

`-newrun` *Force a new rebalance operation rather than resuming a prior one*

(Optional) Use `hsstool -h <host> rebalance -newrun` if a prior rebalance run on the same host was terminated short of successful completion -- by use of the `rebalance -stop` option or by some other interruption such as the target node being rebooted -- and for some reason you need to run a fresh new rebalance operation on the node rather than resuming the prior rebalance operation. This might happen if for example the prior rebalance operation is 'stuck' in some way and not able to be resumed. Another scenario where the `-newrun` option can be used is if the original rebalance run on the target node ended with status FAILED.

If you run a fresh rebalance operation on the node in this way, it will not re-stream to the new node any data that had already been streamed there by the prior rebalance operation (there will not be a duplication of data on the new node). Instead the new rebalance run will detect that this data is already in place on the new node and will not re-stream it.

Since you are starting a fresh rebalance operation with this option, you can specify the `-cleanupfile` option also (if you want data removed from the existing nodes after it's successfully copied to the new node): `hsstool -h <host> rebalance -newrun -cleanupfile`.

14.1.9.2. Usage Notes

This [hsstool](#) command copies S3 object data from your existing nodes to a specified new node that you have added to your HyperStore cluster. The `rebalance` operation populates the new node with its share of S3 object replica data and erasure coded data, based on the token ranges that the system automatically assigned the new node when you added it to the cluster.

The target `<host>` must be a newly added node, unless the `-list` option is used in which case the target `<host>` can be any node.

All other nodes in the cluster must be up and running when you run `rebalance` on a new node.

When to Use `hsstool rebalance`

The only time to use this command is when you have added a new node or nodes to an existing data center. You will then run `rebalance` on the new node(s).

For complete instructions on adding new nodes including the proper use of the `rebalance` command within the context of the procedure, see:

- ["Adding Nodes" \(page 579\)](#)

The rebalance is a background operation that may take many hours or days, depending on your HyperStore cluster size and stored data volume.

You can run `rebalance` on multiple new nodes concurrently. However, **do not run `rebalance` concurrently on a number of new nodes that exceeds one-third the number of your already existing nodes** in that same

data center. For example, if you have 12 existing nodes in a DC and you are adding 6 new nodes to that DC, do not run *rebalance* on more than 4 of the new nodes concurrently (one-third of 12). In this scenario you would run *rebalance* on 4 of the new nodes concurrently, and then when the *rebalance* operation completes on at least 2 of those nodes, you can start *rebalance* on the other 2 new nodes. In this way *rebalance* is never running on more than 4 nodes at a time (never running on more than one-third the number of the pre-existing nodes in the DC).

In the event that the rebalance operation fails for some objects that are supposed to be copied to the new node, these failures will subsequently be corrected automatically by the "[Proactive Repair](#)" (page 131) feature.

CMC Support for This Command

As an alternative to running *hsstool rebalance* on the command line, you can run it through the CMC UI:

The screenshot shows the Cloudian CMC interface. The top navigation bar includes 'Cluster' (highlighted with a red box and a red circle with '1'), 'Alerts', 'Admin', and 'Help'. Below this, the 'Nodes' tab is selected (highlighted with a red box and a red circle with '2'). The 'Advanced' sub-tab is also selected (highlighted with a red box and a red circle with '3'). In the 'Advanced' section, the 'Command Type' dropdown is set to 'Maintenance' (circled in red), and the 'hsstool Command' dropdown is set to 'rebalance' (circled in red). The 'Target Node' dropdown is set to 'cloudian-node5'. Under 'Options', there are checkboxes for 'cleanupfile', 'stop', 'resume', and 'newrun'. The 'Description' field contains the text 'Rebalance data after adding new nodes to a data center'. An 'EXECUTE' button is located at the bottom right of the form.

Note If you launch the operation through the CMC UI, you can track the operation progress through the CMC's [Operation Status](#) page. This way of tracking operation progress is not supported if you launch the operation on the command line. However, regardless of how you launch the operation you can periodically check on its progress by using the [hsstool opstatus](#) command.

14.1.9.3. Checking Operation Status

The *rebalance* operation is a long-running operation. Its duration will depend on multiple factors including the amount of data in your system and the number of nodes in your system. For information about checking the status of an in-progress or recently completed *rebalance* operation see "[hsstool opstatus](#)" (page 882).

14.1.10. hsstool repair

Subjects covered in this section:

- "[Command Syntax](#)" (page 908)
- "[Usage Notes](#)" (page 911)

- ["Checking Operation Status" \(page 915\)](#)

14.1.10.1. Command Syntax

Synopsis

```
# hsstool -h <host> repair [allkeyspaces|nokeyspaces] [-computedigest] [-pr]
[-m <true|false>] [-d <mount-point> -rebuild] [-range <start-token,end-token>]
[-f <input-file> [-opid <operationId>]] [-t <min-timestamp,max-timestamp>]
[-l <true|false>] [-stop] [-resume]
```

Options

allkeyspaces | nokeyspaces *Set scope of metadata repair*

(Optional) You have three alternatives for choosing which Cassandra metadata keyspaces to repair, while also cleaning replicated S3 object data in the HyperStore File System (HSFS):

- Use *allkeyspaces* to repair replicated objects in the HSFS and also repair all the Cassandra keyspaces. Cassandra repair will be completed first, then HSFS replica repair. The Cassandra keyspaces that will be repaired are: *UserData_<storage-policy-ID>* keyspaces; *AccountInfo*; *Reports*; *Monitoring*; and *ECKeyspace*. (For more information see the overview of [Cassandra keyspaces for HyperStore](#)).
- Use *nokeyspaces* to repair only replicated objects in the HyperStore File System, and not any Cassandra keyspaces.
- If you specify neither *allkeyspaces* nor *nokeyspaces* then the default behavior is to repair replicated objects in the HSFS and also to repair the Cassandra *UserData_<storage-policy-ID>* keyspaces (which store object metadata). Cassandra repair will be completed first, then HSFS replica repair.

-computedigest *Repair corrupted replicas as well as missing replicas*

(Optional) Use this option if you want to check for and repair not only missing replicas but also any replicas that are present but corrupted. When doing the repair with the *-computedigest* option, the system will compute a fresh digest for each replica rather than using cached digests. If the re-computed digest of a given replica does not match the original digest (stored in a file alongside the object data) or does not match the re-computed digests of the other replicas of that same object (on other nodes), the replica is considered to be corrupted and is replaced by a copy of a correct replica streamed in from a different node.

This way of running *repair* is resource-intensive.

Note If you wish, you can have some or all of the [scheduled auto-repairs](#) of replica data use the *"-computedigest"* option to combat bit rot. This aspect of auto-repair is controlled by the configuration setting *"hyperstore.repair.scheduled.computeDigest.frequency"* (page 776). By default the *"-computedigest"* option is not used in auto-repair runs.

-pr *Repair only the node's primary ranges*

(Optional) Only repair objects that fall within the target node's primary ranges (objects for which the hash of the object name falls into one of the token ranges assigned to the node). Do not repair objects that fall into other nodes' primary ranges and that are replicated on to the target node. For background information on replication in HyperStore, see ["How vNodes Work" \(page 83\)](#).

If each node in the cluster is being repaired in succession, using this option makes the successive repair operations less redundant and more efficient.

Note The HyperStore auto-repair feature -- which automatically runs node repairs on a schedule -- uses the `-pr` option when it initiates the `hsstool repair` operation. For more on this feature see ["Automated Data Repair Feature Overview"](#) (page 130).

`-m <true|false>` *Repair only objects for which metadata exists*

(Optional, defaults to true) If this option is true, then before repairing a given object the repair process will verify that metadata for the object exists in Cassandra. This prevents attempts to repair objects that had been intentionally deleted by users, as indicated by the absence of corresponding object metadata in Cassandra. Defaults to true, so you only need to specify an `-m` option if you do **not** want the object metadata check to be performed (in which case you'd specify `-m false`).

`-d <mountpoint> -rebuild` *Repair only a specified mountpoint*

(Optional) If you use this option the repair is performed only for objects mapped to the vNodes that are assigned to the specified HyperStore data mount point (for example `hsstool -h <host> repair -d /cloudian1 -rebuild`), either as primary replicas or as secondary or tertiary replicas. This option may be useful in circumstances where data is known or suspected to be missing or incorrect on a particular disk.

You can repair disks on multiple nodes concurrently, by running the `hsstool -h <host> repair -d <mountpoint> -rebuild` command separately for each target host and mountpoint. However you cannot repair multiple disks on the same node concurrently. You can only repair multiple disks concurrently if the disks are on different nodes, so that on any individual node only one targeted disk (mountpoint) is being repaired at a time.

Note If you use the `-d <mount-point> -rebuild` option do not use the `-pr` option or the `-range <start-token,end-token>` option. The system does not support using these options in combination.

`-range <start-token,end-token>` *Repair only a specified token range*

(Optional) If you use this option the repair is performed only for objects mapped to your specified token range. You specify the range by indicating the start token and end token (an example of a token is 18315119863185730105557340630830311535). The repair operation will repair all objects that fall within the token range bounded by the start and end tokens.

The **end-token must be a token that is assigned to the target host**. To see what tokens are assigned to each node you can use the ["hsstool ring"](#) (page 930) command. The **start-token must be the next-lower token in the ring** from the end-token.

This option may be useful if a previous full node repair failed for particular ranges. You can obtain information about range repair failures (including the start and end tokens that bound any failed ranges) by running `hsstool -h <host> opstatus repair -a` on the command line.

Note If you use the `-range <start-token,end-token>` option do not use the `-pr` option or the `-d <mount-point> -rebuild` option. The system does not support using these options in combination.

`-f <input-file> [-opid <operationId>]` *Repair only the objects listed in an input file*

(Optional) See ["Repairing Objects Specified in a File"](#) (page 913).

Note The CMC does not support the `-f <input-file>` option. This option is only supported if you run `hsstool repair` on the command line.

`-t <min-timestamp,max-timestamp>` *Repair only objects last modified within a specified time period*

(Optional) If you use this option the repair is performed only for objects that have a last-modified timestamp equal to or greater than *min-timestamp* and less than *max-timestamp*. When using this option, use Unix milliseconds as the timestamp format.

Note This option is not supported in the CMC interface -- only on the command line.

`-l <true|false>` *Log a list of repaired objects*

(Optional, defaults to true) If this option is true, write to a log file a list of all the objects that were repaired. Defaults to true, so you only need to specify an `-l` option if you do **not** want repair object logging (in which case you'd specify `-l false`).

The log is named *clouddian-hyperstore-repair.log* and is written into the Clouddian HyperStore log directory of the target node. Activity associated with a particular instance of a command run is marked with a unique command number.

`-stop` *Stop an in-progress repair*

(Optional) Use `hsstool -h <host> repair -stop` to stop an in-progress repair on the specified node.

If **Cassandra repair is in-progress** — that is, if *repair* was launched with the default behavior or the *allkeyspaces* option, and the Cassandra part of the repair is still in-progress — the Cassandra repair is terminated. The HSFS replica repair — which would normally launch after the Cassandra repair — is canceled.

If **HSFS replica repair is in-progress** — that is, if *repair* was launched with the *nokeyspaces* option, or if it was launched with the default behavior or the *allkeyspaces* option and the Cassandra part of the repair has already completed and HSFS replica repair is underway — the HSFS replica repair is terminated.

You can subsequently use the "[hsstool opstatus](#)" (page 882) command to confirm that the repair has been stopped (status = TERMINATED) and to see how much repair progress had been made before the stop.

If you subsequently want to resume a repair that you stopped, you can use the *repair -resume* option (but see the first bullet point below for exceptions).

Note that:

- If you want to stop a repair of a particular mountpoint, or a repair of a specified token range, or a repair of objects listed in a file, you can stop the repair operation with simply `hsstool -h <host> repair -stop`. However, be aware that resuming a mountpoint repair (by running the *repair -resume* option) is supported, but resuming a token range repair or a repair of objects listed in a file is not supported.
- The `-stop` option stops a single in-progress repair. It does not disable the HyperStore [scheduled auto-repair](#) feature.

`-resume` *Resume a stopped or interrupted repair*

(Optional) Use `hsstool -h <host> repair -resume` to resume the most recent *repair* run, if you had stopped that run (with the `-stop` command option) or if the run was interrupted (by the target host going down during the repair, for instance). The incomplete repair run will resume from the point of progress at which it was when the repair run was stopped or interrupted.

Note that:

- If during the incomplete repair run you used the `-pr` option or the `-d <mount-point>` option, you do not need to re-specify the option when running `repair -resume`. The system will automatically detect that one of those options was used in the previous repair and will use the same option when resuming the repair.
- If during the incomplete repair run you used the `-range` option, then resuming the repair is not supported. Repair resumption works by starting from whichever token ranges were not repaired by the previous repair. Since a repair that uses the `-range` option targets just one token range, resuming such a repair would not be any different than running that same repair over again. If you want to run the repair over again, do `repair -range <start-token,end-token>` again, not `repair -resume`.
- Resuming a repair of the objects listed in an input file is not supported.
- If you run `repair -resume`, then in the command results the Arguments section will include "resuming-cmd=<n>", where <n> is the number of times that `repair -resume` has been run on the node since the last restart of the HyperStore Service. This Argument string -- which serves to distinguish `repair -resume` result output from regular `repair` result output -- also appears in "[hsstool opstatus](#)" (page 882) results for `repair -resume` operations.

14.1.10.2. Usage Notes

Use this [hsstool](#) command to check whether a physical node has all of the replicated data that it is supposed to have (based on the node's assigned tokens and on replication settings for the system); and to replace or update any data that is missing or out-of-date. Replacement or update of data is implemented by retrieving correct and current replica data from other nodes in the system.

The system will not allow you to run `hsstool repair`:

- On a node on which [hsstool cleanup](#) is currently running.
- On any node if there is a [disabled disk](#) on any node in the same service region.
- On any node if `hsstool repair` is already running on a different node in the same service region. The exception to this rule is if you use the `-pr` option with each repair run (you can run `hsstool repair -pr` on multiple nodes concurrently).

When to Use `hsstool repair`

The HyperStore system automatically uses a combination of [read repair, proactive repair, and scheduled auto-repair](#) to keep the replica data on each node complete and current. Consequently, you should rarely need to manually initiate a full `repair` operation.

However, there are these uncommon circumstances when you should manually initiate `repair` on a specific node:

- If you are removing a "dead" node from your cluster. In this circumstance, after removing the dead node you will run repair on each of the remaining nodes, one node at a time. See "[Removing a Node](#)" (page 603) for details.
- If a node is unavailable for longer than the configurable "[hyperstore.repair.proactive.retryQueue.timeout](#)" (page 774) (default = 4 hours). In this case then the metadata required for implementing [proactive repair](#) on the node will stop being written to Cassandra, and an alert will display in the CMC's **Alerts** page (**Alerts -> Alerts**). When the node comes back online you will need to:

1. Monitor the automatic proactive repair that initiates on the node when the node starts up, until it completes. You can check the CMC's **Repair Status** page (**Cluster -> Repair Status**) periodically to see whether proactive repair is still running on the node that you've brought back online. This proactive repair will repair the objects from during the period when proactive repair metadata was still being written to the Cassandra for the node.
2. After proactive repair on the node completes, manually initiate a full repair of the node (using `hsstool repair` and, if appropriate for your environment, `hsstool repairec`). This will repair objects that were written after the proactive repair queueing time maximum was reached.

Note The *repair* operation will fail if the HyperStore service is down on any of the nodes affected by the operation (the nodes storing the affected token ranges). The operation will also fail if any disk storing affected token ranges is disabled or more than 95% full.

Problems Remedied by *repair* and *repair computedigest*

The table below lists data problem cases and shows whether or not they are remedied by regular *repair* and by *repair* that uses the *computedigest* option. Although regular *repair* can handle some cases of corruption, if corruption is suspected on a node and you're not certain exactly which data is corrupted, it's best to use *repair computedigest*.

Case	Will <i>repair</i> fix it?	Will <i>repair computedigest</i> fix it?
Missing blob file	yes	yes
Missing digest	yes	yes
Missing blob file and digest	yes	yes
Corrupted blob file	no	yes
Corrupted digest	yes	yes
Corrupted blob file and digest	yes	yes

CMC Support for This Command

As an alternative to running `hsstool repair` on the command line, you can run it through the CMC UI:

The screenshot shows the Cloudfire CMC interface. The top navigation bar includes 'Cluster' (1), 'Alerts', 'Admin', and 'Help'. The main navigation bar includes 'Data Centers', 'Nodes' (2), 'Cluster Config', 'Storage Policies', 'KMIP Profiles', 'Repair Status', and 'Operation Status'. The 'Nodes' page has tabs for 'Node Status', 'Node Activity', and 'Advanced' (3). The 'Advanced' section contains a 'Command Type' dropdown set to 'Maintenance', an 'hsstool Command' dropdown set to 'repair', and a 'Target Node' dropdown set to 'Jarvis'. Below these are 'Options' with checkboxes for 'default' (selected), 'allkeyspaces', 'nokeyspaces', 'pr', 'l', 'm', 'computedigest', 'stop', 'resume', and 'rebuild'. There are also fields for 'Mount Point', 'Token Range', 'start', and 'end'. A description at the bottom reads: 'Repair HyperStore node data. For Cassandra Repair please choose allkeyspaces option.' An 'EXECUTE' button is located at the bottom right.

Note If you launch the operation through the CMC UI, you can track the operation progress through the CMC's [Operation Status](#) page. This way of tracking operation progress is not supported if you launch the operation on the command line. However, regardless of how you launch the operation you can periodically check on its progress by using the [hsstool opstatus](#) command.

Repairing Objects Specified in a File

When run on the command line, the *hsstool repair* command supports an option for repairing one or more specific objects that are listed in an input file:

```
# hsstool -h <host> repair -f <input-file> [-opid <operationId>]
```

For *<input-file>*, specify the full absolute path to the input file (including the file name). Both the input file and the directory in which it is located must be readable by the 'cloudian' user or else the repair command will immediately fail and report an error.

The target *<host>* can be any node that is utilized by the storage policies used by the objects that are listed in the input file.

You have two options for obtaining or creating the input file that lists the objects to repair:

- Use the dedicated repair failure log file that the system generates automatically when *hsstool repair* is run and repair fails for some objects.
- Create the input file yourself, to target a specific object or small set of objects.

These methods are described in the sections below.

Using the Repair Failure Log as the Input File

Each time *hsstool repair* is run, if repair fails for any object -- or more precisely, if repair fails for any object "chunk" (see ["Creating an Input File"](#) (page 914) for background information about chunks) -- the failures are written to a dedicated repair failure log, on the node(s) on which the failures occur. In this log file there is an entry for each chunk for which the repair fails. If an *hsstool repair* run results in chunk repair failures on multiple nodes, then a repair failure log will be written on each node on which failures occurred, and on each such node the repair failure log entries will identify the chunks for which repair failed on that node.

On each node, the location and name of the current repair failure log is:

```
/var/log/cloudian/cloudian-hyperstore-repair-failure.log
```

The format of entries to the repair failure log is:

```
ChunkName|Path|OperationId|yyyy-mm-dd HH:mm:ss,SSS|FailReason[|TaskType]
```

Note For more information about this log, including sample entries and log rotation policy, see ["Hyper-Store Service Logs"](#) (page 671).

Within the log entries, the "OperationId" field indicates a system-generated unique ID for the *hsstool repair* run that resulted in the chunk repair failure.

You can use the repair failure log file as input to the *hsstool repair -f <input-file>* command, and if you wish you can use the *-opid <operationId>* option to limit the repair to the chunk repair failures that resulted from a particular *hsstool repair* run. If you do not use the *-opid <operationId>* option, then the repair will be performed for all chunks listed in the log file.

Here is an example of the command syntax, including use of the *-opid* option:

```
hsstool -h 10.50.200.161 repair -f /var/log/cloudian/cloudian-hyperstore-repair-failure.log -opid
a2e2b92a-78a0-1206-983d-000c29b774a0
```

Note that:

- Since *repair* is a long-running operation, and the *cloudian-hyperstore-repair-failure.log* is rotated at the end of each day (or when the log file reaches 10MB in size, whichever occurs first), the log entries associated with a particular *repair* operation may be spread across multiple files -- for example, the entries may be in the current, live *cloudian-hyperstore-repair-failure.log* file and also in one or more rotated log files. You can *grep* for the operation ID of interest to see if entries associated with the operation appear in multiple log files. If so, you will need to run *hsstool repair -f <input-file> [-opid <operationId>]* separately for each file. Wait for the *hsstool repair -f <input-file> [-opid <operationId>]* run to complete for one file before moving on to run it again for the next file (do not run multiple such operations concurrently).
- Rotated *cloudian-hyperstore-repair-failure.log* files are Gzipped. You must unzip them before using them as input to an *hsstool repair -f <input-file>* run.
- The *-opid <operationId>* option is supported only if the repair failure log is the input file. The *-opid <operationId>* option is not supported if you use an input file that you've created (as described below).

Creating an Input File

In the HyperStore File System on each of your nodes, objects are stored as "chunks". Objects smaller than or equal to the chunk size threshold (10MB) are stored as a single chunk. Objects larger than the chunk size threshold are broken into and stored as multiple chunks, with no chunk exceeding the threshold size. In the case of large objects that S3 client applications upload to HyperStore by the Multipart Upload method, HyperStore breaks the individual parts into chunks if the parts exceed the chunk size threshold.

In the context of replication storage policies, after objects are broken into chunks, each object chunk is replicated on to multiple nodes.

The *hsstool repair -f <input-file>* operation acts on individual chunks, and so in the input file each line must specify a chunk name and the full path to the chunk file, in the following format:

```
ChunkName | Path
```

There is no limit on the number of lines that you can include in the file (no limit on the number of chunks that you can specify in the file).

Here is an example in which the object is smaller than the chunk size threshold and so the object is stored as just one chunk. In this case the chunk name is simply in the form of *<bucketname>/<objectname>*. (For background information about chunk **file paths** within the HyperStore File System see "[HyperStore Service and the HSFS](#)" (page 66).)

```
bucket1/HyperFileInstallGuide_v-3.6.pdf|/cloudian1/hsfs/std8ZdRJDskcPvmOg4/
0bb5332b429ccb76466e05bee2915d34/074/156/90721763863541208072539249099911078458.
1554130786616795163-0A3232C9
```

Note Although the example above and those that follow below break to multiple lines in this documentation, in the actual input file each *ChunkName|Path* combination constitutes just one line in the file.

Here is a second example, for an object that was uploaded by a regular S3 PUT operation (not a Multipart Upload) but which is larger than the chunk size threshold and so has been broken into multiple chunks. The

example shows an input file entry for one of those chunks. Note that the chunk name here includes a chunk number suffix (shown in **bold**).

```
bucket1/HyperFileAdminGuide_v-3.6.pdf..0001|/cloudian1/hsfs/std8ZdRJDskcPvmOg4/
0bb5332b429ccb76466e05bee2915d34/087/073/13080395222414127681573583484873262519.
1554124662019670529-0A3232C9
```

In this third example, the object has been uploaded via an S3 Multipart Upload operation. In this example that specifies one of the object's chunks, the chunk name includes a prefix based on the upload ID, as well as a chunk number suffix and part number suffix (all shown in **bold**).

```
m-MDA1NTE5NjEzNTU0MTIzODU3Mjg1/bucket1/cloudian-hyperfile_v-3.6.tar.gz..0001.2|
/cloudian1/hsfs/std8ZdRJDskcPvmOg4/0bb5332b429ccb76466e05bee2915d34/082/100/
39535768889303436494640495599026926454.1554123857285865726-0A3232C9
```

You have two options for obtaining the chunk name and chunk path for chunks that you want to target for repair. One option is to use the [hsstool whereis](#) command for each object that you want to repair. The *whereis* response includes the chunk name(s) and chunk path(s) for the object that you specify. You can copy the chunk name and path from the *whereis* response into your input file.

The other option is to copy one or more entries from the log file *cloudian-hyperstore-repair-failure.log* (described in ["Using the Repair Failure Log as the Input File" \(page 913\)](#)) into a separate file, and use that separate file as the input file. You could do this if you wanted to target a particular object's chunks for another repair attempt, rather than targeting all failures from a preceding repair run.

14.1.10.3. Checking Operation Status

The *repair* operation is a long-running operation. Its duration will depend on multiple factors including the amount of data in your system and the number of nodes in your system. For information about checking the status of an in-progress or recently completed *repair* operation see ["hsstool opstatus" \(page 882\)](#).

14.1.11. hsstool repaircassandra

Subjects covered in this section:

- ["Command Syntax" \(page 915\)](#)
- ["Usage Notes" \(page 917\)](#)
- ["Checking Operation Status" \(page 918\)](#)

14.1.11.1. Command Syntax

Synopsis

```
# hsstool -h <host> repaircassandra [-keyspace <keyspacename> [<columnfamily> <columnfamily>...]]
[-pr] [-local] [-dc <dcname>] [-stop [enforce]]
```

Options

-keyspace <keyspacename> [<columnfamily> <columnfamily>...] *Set scope of metadata repair*

(Optional) Use the *-keyspace* option if you want to repair only a specific keyspace in Cassandra (rather than repairing all keyspace, which is the default behavior). You can also optionally specify one or more column families within that keyspace, if you want to repair just that column family or those column families. If you do not specify a column family then all column families in the specified keyspace will be repaired.

You can only specify one *-keyspace* option per *hsstool repaircassandra* run. Specifying multiple keyspaces is not supported. Note again that the default behavior of *hsstool repaircassandra* -- if you omit the *-keyspace* option -- is to repair all of the Cassandra keyspaces.

Example of using the *-keyspace* option together with a target column family name:

```
... -keyspace UserData_a36d2c44fbfcc12222e9587b3a42997f CLOUDIAN_OBJMETADATA
```

Note The *-keyspace* option is supported only on the command line, not in the CMC UI.

-pr *Repair only the node's primary ranges*

(Optional) Only repair data that falls within the target node's primary ranges (data for which the hash of the data row key falls into one of the token ranges assigned to the node). Do not repair data that falls into other nodes' primary ranges and that is replicated on to the target node.

If each node in the cluster is being repaired in succession, using this option makes the successive repair operations less duplicative and more efficient.

-local *Repair only the nodes in the local data center*

(Optional) Only repair data on nodes in the same data center as the target node.

If you use neither the *-local* option nor the *-dc <dcname>* option, then Cassandra data in all of your HyperStore system's data centers will be repaired.

Note The *-local* option is supported only on the command line, not in the CMC UI.

-dc <dcname> *Repair only the nodes in the specified data center*

(Optional) Only repair data on nodes in the specified data center. For example if you specify *-dc boston* then only the Cassandra data on the nodes in the data center named *boston* will be repaired. The *<dcname>* must be a valid data center name from your HyperStore system configuration.

If you use neither the *-local* option nor the *-dc <dcname>* option, then Cassandra data in all of your HyperStore system's data centers will be repaired.

Note The *-dc <dcname>* option is supported only on the command line, not in the CMC UI.

-stop [-enforce] *Stop an in-progress repair*

(Optional) Use *hsstool -h <host> repaircassandra -stop* to terminate an in-progress Cassandra repair that was initiated via *hsstool* (whether by you or automatically by the system, such as with auto-repair).

If you need to terminate an in-progress Cassandra repair that was initiated via the native Cassandra utility *nodetool*, use *hsstool -h <host> repaircassandra -stop -enforce*. Note that using *nodetool* to initiate a Cassandra repair is not recommended.

Note In the CMC UI the *-stop -enforce* option is called "force stop".

The `hsstool repaircassandra` command does not support a 'resume' option. If you stop an in-progress Cassandra repair, to do the repair again use the `hsstool repaircassandra` command again and the repair will start over.

Note The `-stop` option stops a single in-progress Cassandra repair on a single node. It does **not** disable the HyperStore [scheduled auto-repair](#) feature.

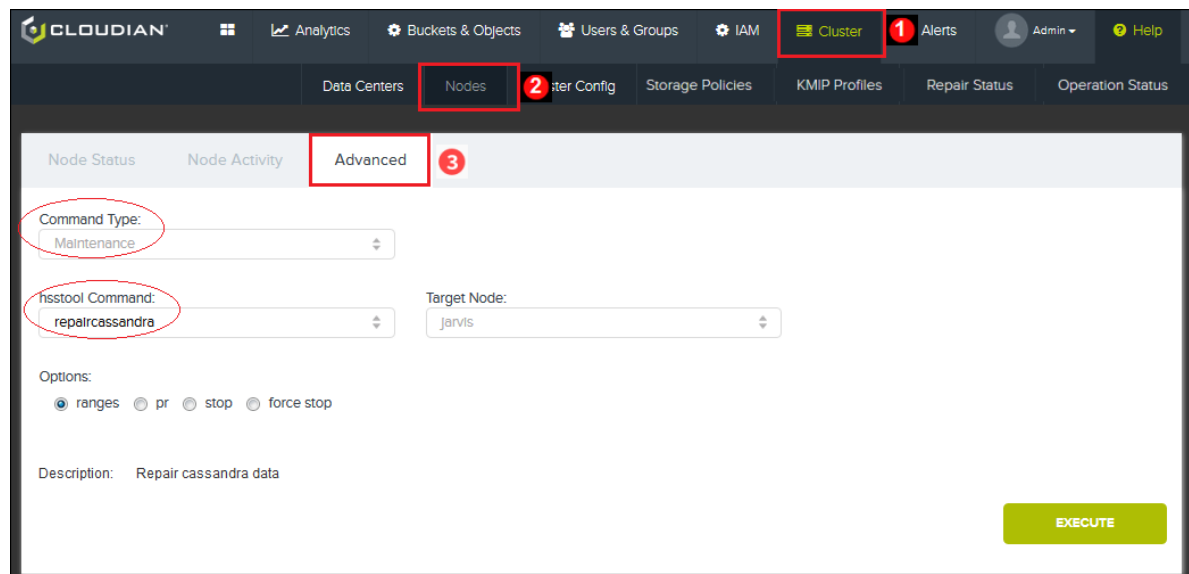
14.1.11.2. Usage Notes

Use this [hsstool](#) command to repair only the Cassandra data on a node (the system metadata and object metadata stored in Cassandra) and **not** the object data on the node. Under normal circumstances you should not need to use this command, but you might use it when in a troubleshooting or recovery situation.

IMPORTANT ! If you do need to initiate a Cassandra-only repair (with no repair of the object data in the HyperStore File System), use this command rather than using the native Cassandra utility `nodetool` to initiate the repair. Using `hsstool repaircassandra` has multiple advantages over using `nodetool repair`, including that with `hsstool repaircassandra` you can track the repair's progress with [hsstool opstatus](#) and you can stop the repair if you need to for some reason.

CMC Support for This Command

As an alternative to running `hsstool repaircassandra` on the command line, you can run it through the CMC UI:



Note As is shown in the CMC interface for `hsstool repaircassandra` and in the status response when you run the command, the command applies a "`ranges=true`" argument by default (the status response includes "`ranges=true`" in the list of arguments). With this method of Cassandra repair, each impacted token range is repaired one range at a time, sequentially. This approach improves the performance for Cassandra repair. Prior to HyperStore release 7.1.5 using this method was optional, but starting with release 7.1.5 it became the default repair behavior.

14.1.11.3. Checking Operation Status

The *repaircassandra* operation is a long-running operation. Its duration will depend on multiple factors including the amount of metadata in your system and the number of nodes in your system. For information about checking the status of an in-progress or recently completed *repaircassandra* operation see ["hsstool opstatus"](#) (page 882).

14.1.12. hsstool repairc

Subjects covered in this section:

- ["Command Syntax"](#) (page 918)
- ["Usage Notes"](#) (page 920)
- ["Checking Operation Status"](#) (page 925)

14.1.12.1. Command Syntax

Synopsis

```
# hsstool -h <host> repairc [-computedigest] [-d <mountpoint> [-reb]]  
[-range <start-token,end-token>] [-b <bucket>] [-f <input-file> [-opid <operationId>]]  
[-l <true|false>] [-stop] [-resume]
```

Options

-computedigest Repair corrupted fragments as well as missing fragments

(Optional) Use this option if you want to check for and repair not only missing erasure coded fragments but also any fragments that are present but corrupted. When doing the repair with the *-computedigest* option, the system computes a fresh digest for each fragment rather than using cached digests. The re-computed digest of each fragment is compared to the original digest of the fragment (stored alongside the fragment data) and if there is a mismatch the fragment is considered to be corrupted. The object is then decoded from healthy fragments and then re-encoded, and the corrupted fragment is replaced by a new and correct fragment.

This way of running *repairc* is resource-intensive.

Note If you wish, you can have some or all of the [scheduled auto-repairs](#) of erasure coded data use the *"-computedigest"* option to combat bit rot. This aspect of auto-repair is controlled by the configurable setting ["hyperstore.repair.scheduled.computeDigest.frequency"](#) (page 776). By default the *"-computedigest"* option is not used in auto-repair runs.

-d <mountpoint> [-reb] Repair only a specified mountpoint

(Optional) If you use this option the repair is performed only for object fragments mapped to the token ranges that are assigned to the specified HyperStore data mount point on the target node -- for example *hsstool -h <host> repairc -d /cloudian1* . This option may be useful in circumstances where data is known or suspected to be missing or incorrect on a particular disk, or when you need to rebuild all the data on a particular disk.

If you are performing the disk repair after an "Add Node" operation has successfully completed and before rebalance has completed for all the new nodes , use the *-reb* option together with *-d <mountpoint>* -- for example *hsstool -h <host> repairc -d /cloudian1 -reb* . In all other circumstances, omit the *-reb* option.

You can repair disks on multiple nodes concurrently, by running the `hsstool -h <host> repairec -d <mountpoint>` command separately for each target host and mountpoint. However you cannot repair multiple disks on the same node concurrently. You can only repair multiple disks concurrently if the disks are on different nodes, so that on any individual node only one targeted disk (mountpoint) is being repaired at a time.

Note

- * If you use the `-d <mountpoint>` option do not use the `-range <start-token,end-token>` option or the `-f <input-file>`. The system does not support combining mountpoint repair with those other repair options.
- * The HyperStore `replaceDisk` function (see ["Replacing a HyperStore Data Disk" \(page 642\)](#)) automatically invokes the `hsstool repairec -d <mountpoint>` command, after automatically performing other tasks required for bringing a replacement disk back into service. If you use the `replaceDisk` function after adding nodes to your cluster and before rebalancing has been completed, the function automatically invokes the `hsstool repairec -d <mountpoint> -reb` command.

`-range <start-token,end-token>` *Repair only a specified token range*

(Optional) If you use this option the repair is performed only for objects mapped to your specified token range. You specify the range by indicating the start token and end token (an example of a token is 18315119863185730105557340630830311535). The repair operation will repair all objects that fall within the token range bounded by the start and end tokens.

The **end-token must be a token that is assigned to the target host**. To see what tokens are assigned to each node you can use the ["hsstool ring" \(page 930\)](#) command. The **start-token must be the next-lower token in the ring** from the end-token.

This option may be useful if a previous full node repair reported failures for a particular token range.

`-b <bucket>` *Repair only a specified bucket*

(Optional) Use the `-b <bucket>` option if you want to repair only the data in a particular bucket that is using a EC or EC Hybrid storage policy. For example, `hsstool -h <host> repairec -b bucket10`. If the bucket is using an EC Hybrid storage policy, the repair operation will repair any replicated objects from the specified bucket as well as the erasure coded objects.

Note The CMC does not support the `-b <bucket>` option. This option is only supported if you run `hsstool repairec` on the command line.

`-f <input-file> [-opid <operationId>]` *Repair only the objects listed in an input file*

(Optional) See ["Repairing Objects Specified in a File" \(page 922\)](#).

Note The CMC does not support the `-f <input-file>` option. This option is only supported if you run `hsstool repairec` on the command line.

`-l <true|false>` *Log a list of repaired objects*

(Optional, defaults to true) If this option is true, write to a log file a list of all the objects that were repaired. Defaults to true, so you only need to specify an `-l` option if you do **not** want repair object logging (in which case you'd specify `-l false`).

The log is named *cloudian-hyperstore-repair.log* and is written into the Clodian HyperStore log directory of the target node.

Failures to repair particular erasure coded object chunks are logged in a separate log file, named *cloudian-hyperstore-repair-failure.log*.

For more information about these logs see ["HyperStore Service Logs"](#) (page 671).

-stop *Stop an in-progress repair*

(Optional) Use *hsstool -h <host> repair -stop* to stop an in-progress erasure coded data repair. This stops the repair immediately. You can subsequently use the ["hsstool opstatus"](#) (page 882) command to confirm that the repair has been stopped (status = TERMINATED) and to see how much repair progress had been made before the stop.

If you subsequently want to resume a repair that you stopped, you can use the *repair -resume* option (but see the first bullet point below for exceptions).

Note that:

- If you want to stop a repair of a particular mountpoint, or a repair of a specified token range, or a repair of objects listed in a file, you can stop the repair operation with simply *hsstool -h <host> repair -stop*. However, be aware that resuming a mountpoint repair (by running the *repair -resume* option) is supported, but resuming a token range repair or a repair of objects listed in a file is not supported.
- The *-stop* option stops a single in-progress repair. It does not disable the HyperStore [scheduled auto-repair](#) feature.

-resume <options> *Resume a stopped or interrupted repair*

(Optional) Use *hsstool -h <host> repair -resume <options>* to resume the most recent *repair* run, if you had stopped that run (with the *-stop* command option) or if the run was interrupted (by the target host going down during the repair, for instance). The incomplete repair run will resume from the point of progress at which it was when the repair run was stopped or interrupted.

Note that:

- If you want to resume a repair of a particular mountpoint, you must specify the same mountpoint option along with the *-resume* option. For example, *hsstool -h <host> repair -d /cloudian1 -resume*

Note This is different than the behavior of the *-resume* option for the ["hsstool repair"](#) (page 907) and ["hsstool rebalance"](#) (page 904) operations. With those operations you do not need to specify the options from the original operation run when resuming an operation.

- Resuming a repair of the objects in a particular token range is not supported.
- Resuming a repair of the objects listed in an input file is not supported.

14.1.12.2. Usage Notes

Use this [hsstool](#) command to evaluate and repair erasure coded object data. When you run *hsstool repair* on a target node, the scope of the repair depends on the [storage policy or policies](#) that you are using in your system:

- For an **erasure coding storage policy confined to a single data center**, the *hsstool repair* operation repairs all erasure coded data on all nodes in the data center in which the target node resides. So, to

repair all the data associated with this policy you only need to run *hsstool repair* on any one node in the data center.

- For a **distributed erasure coding storage policy spanning multiple data centers**, the *hsstool repair* operation repairs all erasure coded data on all nodes in all of the data centers included in the storage policy. To repair all the data associated with this type of storage policy you only need to run *hsstool repair* on one node in any one of the data centers included in the storage policy.
- For a **replicated erasure coding storage policy spanning multiple data centers**, the *hsstool repair* operation repairs all erasure coded data on all nodes in the data center in which the target node resides. To repair all the data associated with this type of storage policy you must run *hsstool repair* on one node in each of the data centers included in the storage policy.

The repair process entails replacing erasure coded object fragments that are missing, outdated, or corrupted. Replacement of a missing or bad object fragment is implemented by using the object's good fragments to decode the object, re-encoding the object, and then re-writing the missing or bad fragment to the correct end-point node. To repair an erasure coded object in this manner, there must be at least "k" good fragments present for the object within the system.

Note that HyperStore 7.5.1 and later supports EC Hybrid storage policies whereby objects larger than a [configurable size threshold](#) are erasure coded and objects at or smaller than that threshold are replicated. The *hsstool repair* operation will repair the erasure coded objects associated with EC Hybrid storage policies and **will also repair the replicated objects associated with EC Hybrid storage policies**. For data in such storage policies, the repair scope for the replicated objects is the same as the repair scope for the erasure coded objects (for example, in the case of an EC Hybrid storage policy confined to a single data center, you only need to run *hsstool repair* against one target node in the data center to repair all the erasure coded data and all the replica data associated with that storage policy).

The system will not allow you to run a full *hsstool repair*:

- On any node if a full *hsstool repair* is already running on a different node in the same data center. You cannot run a full *hsstool repair* on more than one node in the same data center concurrently. Note however that you can run a mountpoint-specific repair (with the *-d <mountpoint>* option) on multiple hosts concurrently, if multiple hosts have disks that are in need of repair.
- On a node on which [hsstool cleanupec](#) is currently running.
- On any node if there is a [disabled disk](#) on any node in the same service region.

If the HyperStore Service or Cassandra Service is down on a node in the same data center as the target node, the system does allow you to run *hsstool repair* but the repair will skip all objects that have a fragment on the node on which the HyperStore Service or Cassandra Service is down. Those objects will not be added to the proactive repair queue -- instead, those objects will remain un-repaired until you subsequently run *hsstool repair* again with the HyperStore Service and Cassandra Service both up on that node.

Note In a large cluster with high data volume *hsstool repair* is a long-running operation that may take multiple weeks to complete.

When to Use *hsstool repair*

The HyperStore system automatically uses a combination of [read repair, proactive repair, and scheduled auto-repair](#) to keep the erasure coded data on each node complete and current. Consequently, you should rarely need to manually initiate a full *repair* operation.

However, if you use erasure coding in your system, there are these uncommon circumstances when you should manually initiate a *repair* operation:

- If you are removing a "dead" node from your cluster. In this circumstance, after removing the dead node you will run *repairec* on one node in each of your data centers. See ["Removing a Node" \(page 603\)](#) for details.
- If a node is unavailable for longer than the configurable **"hyper-store.repair.proactive.retryQueue.timeout"** (page 774) (default = 4 hours). In this case then the metadata required for implementing [proactive repair](#) on the node will stop being written to Cassandra, and an alert will display in the CMC's [Alerts](#) page. When the node comes back online you will need to:
 1. Monitor the automatic proactive repair that initiates on the node when the node starts up, until it completes. You can check the CMC's [Repair Status](#) page periodically to see whether proactive repair is still running on the node that you've brought back online. This proactive repair will repair the objects from during the period when proactive repair metadata was still being written to the Cassandra for the node.
 2. After proactive repair on the node completes, manually initiate a full repair of the node (using *hsstool repairec* and [hsstool repair](#)). This will repair objects that were written after the proactive repair queueing time maximum was reached.

CMC Support for This Command

You can also run the *hsstool repairec* command through the CMC UI:

Note If you launch the operation through the CMC UI, you can track the operation progress through the CMC's [Operation Status](#) page. This way of tracking operation progress is not supported if you launch the operation on the command line. However, regardless of how you launch the operation you can periodically check on its progress by using the [hsstool opstatus](#) command.

Repairing Objects Specified in a File

When run on the command line, the *hsstool repairec* command supports an option for repairing one or more specific objects that are listed in an input file:

```
# hsstool -h <host> repairec -f <input-file> [-opid <operationId>]
```

For *<input-file>*, specify the full absolute path to the input file (including the file name). Both the input file and the directory in which it is located must be readable by the 'cloudian' user or else the repair command will immediately fail and report an error.

The target *<host>* can be any node that is utilized by the storage policies used by the objects that are listed in the input file.

You have two options for obtaining or creating the input file that lists the objects to repair:

- Use the dedicated repair failure log file that the system generates automatically when *hsstool repair* is run and repair fails for some objects.
- Create the input file yourself, to target a specific object or small set of objects.

These methods are described in the sections below.

Using the Repair Failure Log as the Input File

Each time *hsstool repair* is run, if repair fails for any object -- or more precisely, if repair fails for any object "chunk" (see ["Creating an Input File" \(page 924\)](#) for background information about chunks) -- the failures are written to a dedicated repair failure log, on the node(s) on which the failures occur. In this log file there is an entry for each chunk for which the repair fails. If an *hsstool repair* run results in chunk repair failures on multiple nodes, then a repair failure log will be written on each node on which failures occurred, and on each such node the repair failure log entries will identify the chunks for which repair failed on that node.

On each node, the location and name of the current repair failure log is:

```
/var/log/cloudian/cloudian-hyperstore-repair-failure.log
```

The format of entries to the repair failure log is:

```
ChunkName|Path|OperationId|yyyy-mm-dd HH:mm:ss,SSS|FailReason[|TaskType]
```

Note For more information about this log, including sample entries and log rotation policy, see ["Hyper-Store Service Logs" \(page 671\)](#).

Within the log entries, the "OperationId" field indicates a system-generated unique ID for the *hsstool repair* run that resulted in the chunk repair failure.

You can use the repair failure log file as input to the *hsstool repair -f <input-file>* command, and if you wish you can use the *-opid <operationId>* option to limit the repair to the chunk repair failures that resulted from a particular *hsstool repair* run. If you do not use the *-opid <operationId>* option, then the repair will be performed for all chunks listed in the log file.

Here is an example of the command syntax, including use of the *-opid* option:

```
hsstool -h 10.50.200.161 repair -f /var/log/cloudian/cloudian-hyperstore-repair-failure.log -opid a2e2b92a-78a0-1206-983d-000c29b774a0
```

Note that:

- Since *repair* is a long-running operation, and the *cloudian-hyperstore-repair-failure.log* is rotated at the end of each day (or when the log file reaches 10MB in size, whichever occurs first), the log entries associated with a particular *repair* operation may be spread across multiple files -- for example, the entries may be in the current, live *cloudian-hyperstore-repair-failure.log* file and also in one or more rotated log files. You can *grep* for the operation ID of interest to see if entries associated with the operation appear in multiple log files. If so, you will need to run *hsstool repair -f <input-file> [-opid*

`<operationId>]` separately for each file. Wait for the `repair` run to complete for one file before moving on to run it again for the next file (do not run multiple `repair` operations concurrently).

- Rotated `cloudian-hyperstore-repair-failure.log` files are Gzipped. You must unzip them before using them as input to an `hsstool repair -f <input-file> [-opid <operationId>]` run.
- The `-opid <operationId>` option is supported only if the repair failure log is the input file. The `-opid <operationId>` option is not supported if you use an input file that you've created (as described below).

Creating an Input File

In the HyperStore File System on each of your nodes, objects are stored as "chunks". Objects smaller than or equal to the chunk size threshold (10MB) are stored as a single chunk. Objects larger than the chunk size threshold are broken into and stored as multiple chunks, with no chunk exceeding the threshold size. In the case of large objects that S3 client applications upload to HyperStore by the Multipart Upload method, HyperStore breaks the individual parts into chunks if the parts exceed the chunk size threshold.

In the context of erasure coding storage policies, after objects are broken into chunks, each object chunk is erasure coded.

The `hsstool repair -f <input-file>` operation acts on individual chunks, and so in the input file each line must specify a chunk name and the full path to the chunk file, in the following format:

```
ChunkName|Path
```

There is no limit on the number of lines that you can include in the file (no limit on the number of chunks that you can specify in the file).

Here is an example in which the object is smaller than the chunk size threshold and so the object is stored as just one chunk. In this case the chunk name is simply in the form of `<bucketname>/<objectname>`. (For background information about chunk **file paths** within the HyperStore File System see ["HyperStore Service and the HSFS" \(page 66\)](#).)

```
bucket1/HyperFileInstallGuide_v-3.6.pdf|/cloudian1/ec/std8ZdRJDskcPvmOg4/
0bb5332b429ccb76466e05bee2915d34/074/156/90721763863541208072539249099911078458.
1554130786616795163-0A3232C9
```

Note Although the example above and those that follow below break to multiple lines in this documentation, in the actual input file each `ChunkName|Path` combination constitutes just one line in the file.

Here is a second example, for an object that was uploaded by a regular S3 PUT operation (not a Multipart Upload) but which is larger than the chunk size threshold and so has been broken into multiple chunks. The example shows an input file entry for one of those chunks. Note that the chunk name here includes a chunk number suffix (shown in bold).

```
bucket1/HyperFileAdminGuide_v-3.6.pdf..0001|/cloudian1/ec/std8ZdRJDskcPvmOg4/
0bb5332b429ccb76466e05bee2915d34/087/073/13080395222414127681573583484873262519.
1554124662019670529-0A3232C9
```

In this third example, the object has been uploaded via an S3 Multipart Upload operation. In this example that specifies one of the object's chunks, the chunk name includes a prefix based on the upload ID, as well as a chunk number suffix (all shown in bold).

```
m-MDA1NTE5NjExNTU0MTIzODU3Mjg1/bucket1/cloudian-hyperfile_v-3.6.tar.gz..0001.2|
/cloudian1/ec/std8ZdRJDskcPvmOg4/0bb5332b429ccb76466e05bee2915d34/082/100/
39535768889303436494640495599026926454.1554123857285865726-0A3232C9
```


You have two options for obtaining the chunk name and chunk path for chunks that you want to target for repair. One option is to use the [hsstool whereis](#) command for each object that you want to repair. The *whereis* response includes the chunk name(s) and chunk path(s) for the object that you specify. You can copy the chunk name and path from the *whereis* response into your input file.

Note The *whereis* response has information for each erasure coded fragment, on each node on which the fragments reside. For a given object chunk, each erasure coded fragment has the same chunk name and chunk path, so you can get this information from any of the fragments, regardless of which node a particular fragment is stored on.

The other option is to copy one or more entries from the log file *cloudian-hyperstore-repair-failure.log* (described in ["Using the Repair Failure Log as the Input File"](#) (page 923)) into a separate file, and use that separate file as the input file. You could do this if you wanted to target a particular object's chunks for another repair attempt, rather than targeting all failures from a preceding repair run.

14.1.12.3. Checking Operation Status

The *repair* operation is a long-running operation. Its duration will depend on multiple factors including the amount of data in your system and the number of nodes in your system. For information about checking the status of an in-progress or recently completed *repair* operation see ["hsstool opstatus"](#) (page 882).

14.1.13. hsstool repairqueue

Subjects covered in this section:

- ["Command Syntax"](#) (page 925)
- ["Usage Notes"](#) (page 926)
- ["Command/Response Example "](#) (page 928)

14.1.13.1. Command Syntax

Synopsis

```
# hsstool [-h <host>] repairqueue [-t replicas|ec|cassandra|readreplicas|readec] [-inc]
[-enable true|false]
```

Options

-t replicas|ec|cassandra|readreplicas|readec Apply command only to a specified repair type

(Optional) The *hsstool repairqueue* command either:

- Returns basic status information for automated repairs, if the *[-enable true|false]* option is omitted.
- Enables or disables automated repairs, if the *[-enable true|false]* option is used.

If you do not use the *-t <repair-type>* option, then the *hsstool repairqueue* command (either for retrieving status or for enabling or disabling) applies to **all** automated repair types. If you do use the *-t <repair-type>* option, then the command applies only to the specified automated repair type.

The supported automated repair types for the *-t* option are:

- *replicas* -- Scheduled auto-repair of replicated object data.
- *ec* -- Scheduled auto-repair of erasure coded object data.
- *cassandra* -- Scheduled auto-repair of the Metadata DB (metadata stored in Cassandra).
- *readreplicas* -- Read repair (repair-on-read) for replicated object data.
- *readec* -- Read repair (repair-on-read) for erasure coded object data.

For more information about these automated repair types see ["Automated Data Repair Feature Overview" \(page 130\)](#).

Note that:

- If you run *hsstool repairqueue -enable true|false*, without using the *-t <repair-type>*, then the enabling or disabling operation applies also to proactive repair. If you want to enable or disable only proactive repair without impacting the scheduled auto-repair or repair-on-read features you can do so with the ["hsstool proactiverepairq" \(page 899\)](#) command.
- Disabling the schedule auto-repair feature (the *replicas*, *ec*, or *cassandra* automated repair types) **does not abort in-progress auto-repairs**. Rather, it prevents any additional scheduled auto-repairs from launching. (For information about stopping in-progress repairs, see ["hsstool repair" \(page 907\)](#) and ["hsstool repairec" \(page 918\)](#)).
- If you use the *-enable true|false* option this enables or disables automated repair operations on **all nodes in the service region of the specified host** (*-h <host>*).

IMPORTANT ! The automated repair features are essential for maintaining data integrity in your system. Do not leave them disabled permanently.

-inc *Restrict '-t cassandra' option to incremental repair*

(Optional) You can use *-t cassandra -inc -enable true|false* options if you want the enabling or disabling action to apply only to Cassandra incremental auto-repair. For Cassandra, two types of auto-repair are executed on schedule for each node: incremental repair (once a day) and full repair (once a week). For further information see ["Auto-Repair Schedule Settings" \(page 507\)](#).

If you use *-t cassandra -enable true|false* without the *-inc* option then your enabling or disabling action applies to both types of Cassandra auto-repair (incremental and full).

-enable true|false *Enable or disable auto-repair*

(Optional) For more information see the description of the *-t replicas|ec|cassandra|readreplicas|readec* option.

14.1.13.2. Usage Notes

The HyperStore scheduled auto-repair feature implements a periodic automatic repair of replicated object data, erasure coded object data, and Cassandra metadata on each node in your system. The HyperStore repair-on-read feature performs repairs on objects after they are read by client applications, for objects that are readable but that do not have all of the replicas or erasure coded fragments that they are supposed to have. With the *hsstool repairqueue* command you can:

- **Check on the status of each of these automated repair types.** For scheduled auto-repair types you can check the upcoming repair schedule as well as the status from the most recent auto-repair runs. For read repair types you can check whether the feature is currently enabled or disabled.
- **Temporarily disable** a particular automated repair type or all automated repair types.

- **Re-enable** a particular automated repair type or all automated repair types, if you previously disabled the automated repair type(s).

For background information on automated data repair in HyperStore see "[Automated Data Repair Feature Overview](#)" (page 130).

Note You cannot enable or disable automated repair operations for just one particular node. These features are either enabled or disabled for a service region as a whole. Therefore when running the `hsstool repairqueue` command the target `<host>` can be any node.

When to Use `hsstool repairqueue`

With the `hsstool repairqueue` command you can disable (and subsequently re-enable) the HyperStore automated repair features. You should disable the automated repair features before performing the following cluster operation:

- "[Removing a Node](#)" (page 603)

IMPORTANT ! If you disable automated repairs in order to perform an operation, be sure to re-enable them afterward.

Note The system automatically disables the automated repair features when you upgrade your HyperStore software version or when you add nodes to your cluster; and the system automatically re-enables the automated repair features after these operations are completed. You do not need to use `hsstool repairqueue` when you perform those operations.

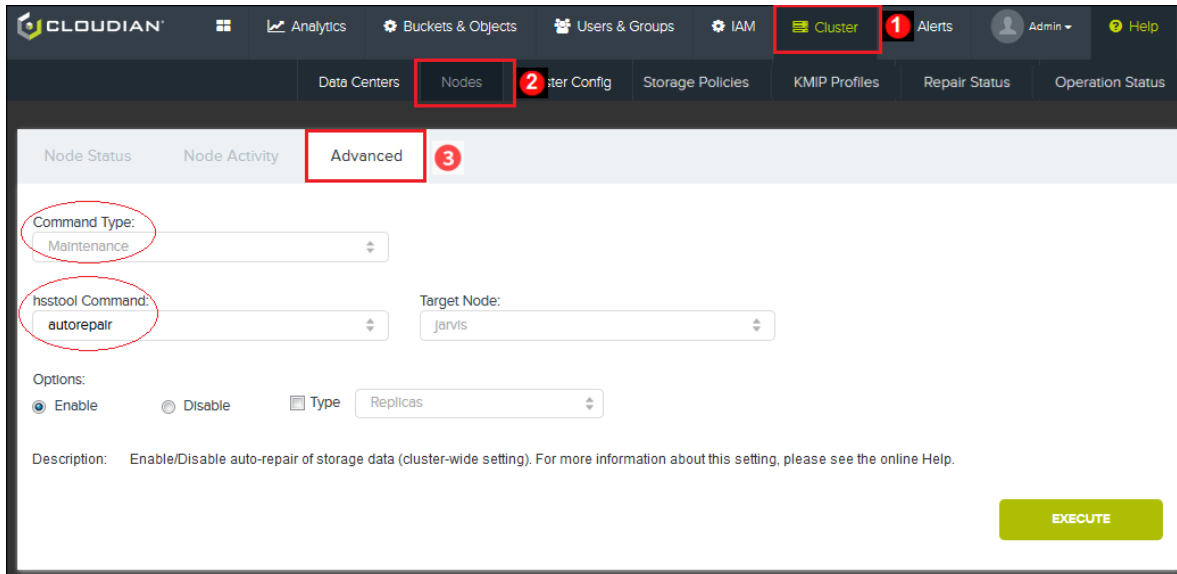
CMC Support for this Command

In the CMC UI you can run the `hsstool repairqueue` command's status check for automated repair features through this interface:

The screenshot shows the Cloudian CMC interface. The top navigation bar includes 'Cluster' (highlighted with a red box and '1'), 'Alerts', 'Admin', and 'Help'. The main navigation bar includes 'Data Centers', 'Nodes' (highlighted with a red box and '2'), 'Cluster Config', 'Storage Policies', 'KMIP Profiles', 'Repair Status', and 'Operation Status'. The 'Nodes' page has three tabs: 'Node Status', 'Node Activity', and 'Advanced' (highlighted with a red box and '3'). The 'Advanced' tab contains a 'Command Type' dropdown set to 'Info', an 'hsstool Command' dropdown set to 'repairqueue', and a 'Target Node' dropdown set to 'jarvis'. A description 'Display repair queues' is shown below. An 'EXECUTE' button is located at the bottom right.

You cannot specify a repair type; instead, the CMC will show status for all automated repair types.

The function for disabling and re-enabling scheduled auto-repairs has its own separate CMC interface and the command is there renamed as "autorepair" (although *hsstool repairqueue* is being invoked behind the scenes):



Disabling read repairs is not supported through the CMC.

14.1.13.3. Command/Response Example

The first *repairqueue* example below shows the "replicas" auto-repair queue status for a recently installed six-node cluster.

```
# hsstool -h cloudbian-node1 repairqueue -t replicas

Queue: replicas Auto-repair: enabled #endpoints: 6
1: endpoint: 192.168.204.201 next repair at: Tue Mar 19 22:32:57 PDT 2024 last repair status:
INIT interval: 43200 mins count: 0
2: endpoint: 192.168.204.202 next repair at: Tue Mar 19 22:32:58 PDT 2024 last repair status:
INIT interval: 43200 mins count: 0
3: endpoint: 192.168.204.203 next repair at: Tue Mar 19 22:32:59 PDT 2024 last repair status:
INIT interval: 43200 mins count: 0
4: endpoint: 192.168.204.204 next repair at: Tue Mar 19 22:33:01 PDT 2024 last repair status:
INIT interval: 43200 mins count: 0
5: endpoint: 192.168.204.205 next repair at: Tue Mar 19 22:33:02 PDT 2024 last repair status:
INIT interval: 43200 mins count: 0
6: endpoint: 192.168.204.206 next repair at: Tue Mar 19 22:33:03 PDT 2024 last repair status:
INIT interval: 43200 mins count: 0
```

Note The response attributes for the "ec" and "cassandra" auto-repair queues would be the same as for the "replicas" queue ("next repair at", "last repair status", and so on) -- except that for the "cassandra" repair queue the response also includes a "repairScope" attribute which distinguishes between "INCREMENTAL" (for Cassandra incremental repairs) and "DEFAULT" (for Cassandra full repairs).

The next example command disables "replicas" auto-repair. Note that this disables replicated object data auto-repair for the **whole service region**. It does not matter which node you submit the command to.

```
# hsstool -h cloudian-node1 repairqueue -enable false -t replicas
Auto replicas repair disabled
```

Response Items

When you use the *repairqueue* command to retrieve scheduled auto-repair queue information, the command results have three sections — one for each repair type. Each section consists of the following items:

Queue

Auto-repair type — either "replicas", "ec", or "cassandra"

Auto-repair

Enabled or disabled

#endpoints

Number of nodes in the cluster. Each node is separately scheduled for repair, for each repair type.

<Queue position>

This is an integer that indicates the position of this node within the cluster-wide queue for auto-repairs of this type. The node at the head of the queue has queue position "1" and is listed first in the command results.

endpoint

IP address of a node

next repair at

For each repair type, each node's *next repair at* value is determined by adding the configurable auto-repair *interval* for that repair type to the start-time of the last repair of that type done on that node. It's important to note that *next repair at* values are used to **order** the cluster-wide queues for each repair type, but the next repair of that type on that node won't necessarily start at that exact time. This is because the queue processing logic takes into account several other considerations along with the scheduled repair time.

Note For erasure coded (EC) object repair, the "next repair at" values are not relevant. Ignore these values. This is because auto-repair for erasure coded objects is run against just one randomly selected target host in each data center each 29-day auto-repair period (and this results in repair of all EC objects in the whole data center).

last repair status

This can be INIT (meaning no repair has been performed on that node yet), INPROGRESS, COMPLETED, TERMINATED (meaning that an in-progress repair was aborted by the operator using the "stop" option for ["hsstool repair"](#) (page 907) or ["hsstool repairec"](#) (page 918)), or FAILED.

If a node restart interrupts a repair, that repair job is considered FAILED and it goes to the head of the queue.

interval

The [configurable interval](#) at which this type of repair is automatically initiated on each node, in number of minutes. (Note though the qualifiers indicated in the "next repair at" description above).

count

The number of repairs of this type that have been executed on this node since HyperStore was installed on the

node.

14.1.14. hsstool ring

Subjects covered in this section:

- ["Command Syntax" \(page 930\)](#)
- ["Usage Notes" \(page 930\)](#)
- ["Command/Response Example" \(page 930\)](#)

14.1.14.1. Command Syntax

Synopsis

```
# hsstool [-h <host>] ring
```

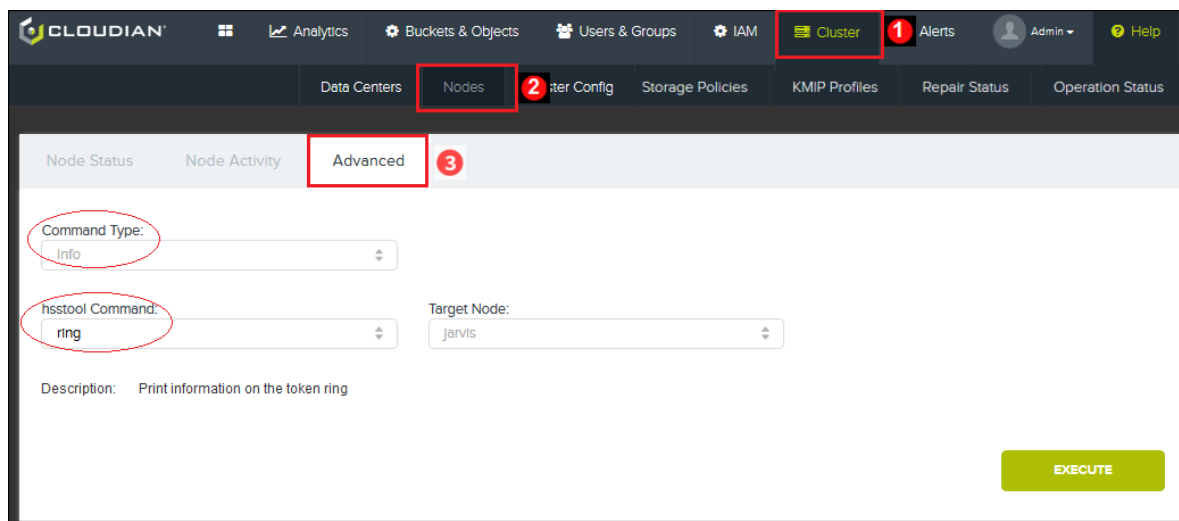
14.1.14.2. Usage Notes

This [hsstool](#) command provides status information for each of the dozens or hundreds of [virtual nodes](#) (vNodes) in your storage cluster. It is very granular and verbose. In most circumstances you will find more value in using the [hsstool info](#) or [hsstool status](#) commands rather than *ring*.

Since this command retrieves information for the whole cluster, the target *<host>* can be any node.

CMC Support for This Command

As an alternative to running *hsstool ring* on the command line, you can run it through the CMC UI:



14.1.14.3. Command/Response Example

The *ring* command results display a status line for each virtual node (vNode) in the storage cluster. In a typical cluster the *ring* command may return hundreds of lines of information. The returned information is sorted by ascending vNode token number.

The example below is an excerpt from a *ring* command response for a four node HyperStore system that spans two data centers. Each of the four physical nodes has 32 vNodes, so the full response has 128 data lines. The list is sorted by ascending vNode token number. Note that although the command is submitted to a particular node ("cloudian-node1"), it returns information for the whole cluster. It doesn't matter which node you submit the command to.

```
# hsstool -h cloudian-node1 ring
Address      DC    Rack  Cassandra  Cassandra-Load  Hss  State  Token
105.236.130.70 DC1   RAC1   Up         2.05 MB         Up   Normal 2146203117201265879150333284323068618
105.236.130.70 DC1   RAC1   Up         2.05 MB         Up   Normal 5542328528654630725776532927863868383
105.236.218.176 DC2   RAC1   Up         1.35 MB         Up   Normal
12000523287982171803377514999547780254
105.236.218.176 DC2   RAC1   Up         1.35 MB         Up   Normal
13878365488241145156735727847865047180
198.199.106.194 DC1   RAC1   Up         1.9 MB          Up   Normal
18315119863185730105557340630830311535
...
...
```

Response Items

Address

IP address of the physical node on which the vNode resides.

DC

Data center in which the vNode resides.

Rack

Rack in which the vNode resides.

Cassandra

Cassandra Service status of the vNode. Will be one of: "Up", "Down", "Joining" (in the process of joining the cluster), "Leaving" (in the process of decommissioning or being removed from the cluster), or "?" (physical host cannot be reached). All vNodes on a physical node will have the same Cassandra status.

Cassandra-Load

Cassandra load (quantity of data stored in Cassandra) for the physical host on which the vNode resides. There will be some Cassandra load even if all S3 objects are stored in the HyperStore File System or the erasure coding file system. For example, Cassandra is used for storage of object metadata and service usage data, among other things. Note that Cassandra load information is available only for the physical node as a whole; it is not available on a per-vNode basis.

HSS

HyperStore Service status for the vNode. Will be one of: "Up", "Down", or "?" (physical host cannot be reached). All vNodes on a physical node will have the same HSS status.

State

HyperStore Service state for the vNode. Will be one of: "Normal" or "Decommissioning". All vNodes on a given physical node will have the same HSS state.

Token

The vNode's token (from an integer token space ranging from 0 to $2^{127} - 1$). This token is the top of the token range that constitutes the vNode. Each vNode's token range spans from the next-lower token (exclusive) in the cluster up to its own token (inclusive).

14.1.15. hsstool status

Subjects covered in this section:

- ["Command Syntax" \(page 932\)](#)
- ["Usage Notes" \(page 932\)](#)
- ["Command/Response Example" \(page 932\)](#)

14.1.15.1. Command Syntax

Synopsis

```
# hsstool [-h <host>] status
```

14.1.15.2. Usage Notes

This [hsstool](#) command returns status information for the storage cluster as a whole.

Since this command retrieves information for the whole cluster, the target *<host>* can be any node.

CMC Support for This Command

As an alternative to running *hsstool status* on the command line, you can run it through the CMC UI:

The screenshot shows the Cloudian CMC interface. The top navigation bar includes 'Cluster' (highlighted with a red box and a red circle with '1'), 'Alerts', 'Admin', and 'Help'. Below this, the 'Nodes' tab is selected (highlighted with a red box and a red circle with '2'). The 'Advanced' sub-tab is also selected (highlighted with a red box and a red circle with '3'). In the 'Advanced' section, the 'Command Type' dropdown is set to 'Info' (circled in red), and the 'hsstool Command' dropdown is set to 'status' (circled in red). The 'Target Node' dropdown is set to 'jarvis'. A description reads 'Print cluster information'. An 'EXECUTE' button is located at the bottom right.

14.1.15.3. Command/Response Example

The *status* command example below retrieves the status of a four-node cluster.

```
# hsstool -h cld05-02 status
Datacenter: DC1
=====
Address      DC    Rack    Cassandra Tokens  Cassandra-Load  Hss State      Hyperstore-Disk  Host-ID
```


Hostname											
10.20.2.54	DC1	RAC1	Up	32	0.24118415	Up	Normal	427.66MB/7.75GB (5.39%)	cfa851df-4170-4006-a516-d5b56f99a820	c1d05-04	
10.20.2.57	DC1	RAC1	Down	32	0.2239672	Up	Normal	79.45MB/11.17GB (0.69%)	6bf98091-5318-4ae6-acc3-f1524f5e7da3	c1d05-07	
10.20.2.55	DC1	RAC1	Up	32	0.23458646	Up	Normal	1.1GB/20.48GB (5.36%)	c77ff93a-5579-4f9f-b5a3-9e22d68fab84	c1d05-05	
10.20.2.52	DC1	RAC1	Up	32	0.3002622	Up	Normal	692.7MB/12.61GB (5.37%)	914117cb-3ec5-4c79-9dff-5a9f33002ce3	c1d05-02	

Response Items

Address

IP address of the physical node on which the vNode resides.

DC

Data center in which the node resides.

Rack

Rack in which the node resides.

Cassandra

Cassandra Service status of the node. Will be one of: "Up", "Down", "Joining" (in the process of joining the cluster), "Leaving" (in the process of decommissioning or being removed from the cluster), or "?" (physical host cannot be reached).

Cassandra-Load

Of the Cassandra data in the system as a whole, the portion (as a decimal) that is stored on each node.

HSS

HyperStore Service status for the node. Will be one of: "Up", "Down", or "?" (physical host cannot be reached).

State

HyperStore Service state for the node. Will be one of: "Normal" or "Decommissioning".

HyperStore-Disk

HyperStore data disk utilization on the node

Host-ID

System-generated unique ID for the node

Hostname

Hostname of the node

14.1.16. hsstool trmap

Subjects covered in this section:

- ["Command Syntax" \(page 934\)](#)
- ["Usage Notes" \(page 934\)](#)
- ["Command/Response Examples" \(page 935\)](#)

14.1.16.1. Command Syntax

Synopsis

```
# /opt/cloudian/bin/hsstool -h <host> trmap [list [-a]] [show <snapshotId>]
[set <snapshotId> <active/disabled>] [delete <snapshotId>]
```

Options

list [-a] *List token range map snapshot IDs*

(Optional) Using *list* **without** the optional *-a* flag returns only a list of active token range map snapshot IDs. These are token range map snapshots for which the associated rebalancing operation has not yet completed.

If you use the *-a* flag -- that is, *trmap list -a* -- then the command returns the IDs of **all** snapshots -- the active snapshots and also the disabled snapshots (snapshots for which the associated rebalancing operation has completed). When you use the *-a* flag the return includes a status field for each snapshot, to distinguish active snapshots from disabled snapshots. For example:

```
[root]# /opt/cloudian/bin/hsstool -h hyperstore1 trmap list -a
59af5410-b303-41bd-94a0-31ce92058e11 Tue Jun 26 09:03:56 EDT 2023 DISABLED
061e9190-6d62-429d-85dc-e4baec07f49e Wed Jun 19 15:26:07 EDT 2024 ACTIVE
22351e62-e921-4130-afaf-1ca3183046e1 Thu Jun 20 14:17:04 EDT 2024 ACTIVE
1c84d57e-dccb-47b0-af6d-015e823eab07 Wed Jun 19 21:42:42 EDT 2024 ACTIVE
9e096b8f-5914-42fc-8dc7-063de85deb4a Thu Jun 20 06:56:17 EDT 2024 ACTIVE
3bf72d7e-a2a1-4551-972c-1223a4485335 Mon Jun 25 17:23:32 EDT 2023 DISABLED
dbf98852-2557-4cfd-8708-0398009fc6d7 Mon Jun 25 18:38:47 EDT 2023 DISABLED
```

show <snapshotId> *Show snapshot content*

(Optional) This returns the content of the specified token range map snapshot. This option is only supported on the command line, not in the CMC interface.

set <snapshotId> <active/disabled> *Change snapshot status*

(Optional) **Do not use this option unless instructed to do so by Cloudian Support.** This sets the status of the specified token range map snapshot to *active* (rebalancing in connection with this snapshot still needs to be completed) or *disabled* (rebalancing in connection with this snapshot has been completed). This option is only supported on the command line, not in the CMC interface.

delete <snapshotId> *Delete snapshot*

(Optional) **Do not use this option unless instructed to do so by Cloudian Support.** This deletes the specified token range map snapshot. This option is only supported on the command line, not in the CMC interface.

14.1.16.2. Usage Notes

This [hsstool](#) command returns a list of token range map snapshot IDs along with information about each snapshot such as the snapshot creation time. You can also use the command to return the contents of a specified token range map snapshot.

The system creates a token range map snapshot each time you [add a new node to your cluster](#). The token range map identifies, for each storage policy in your system, the nodes (endpoints) that store data from each token range. The data from each token range will be stored on multiple nodes, with the number of nodes depending on the storage policy (for example, in a 3X replication storage policy each token range would be mapped to three storage endpoints). When you've added new nodes to your cluster, the system uses token range maps to manage the rebalancing of S3 object data from existing nodes to the new nodes.

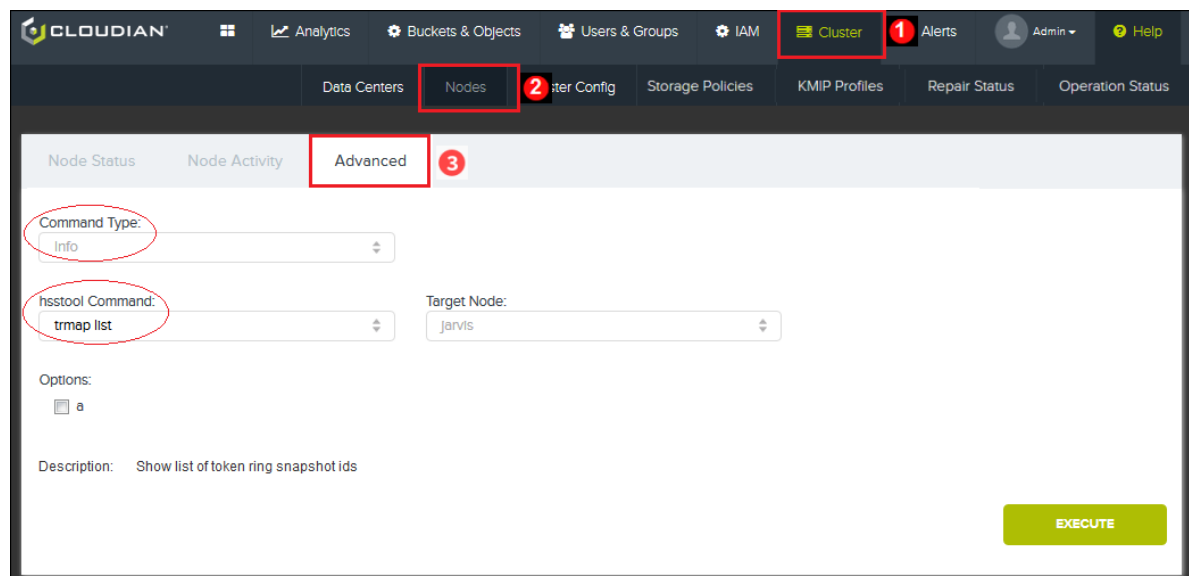
Typically you should not need to use this command unless you are working with Cloudian Support to troubleshoot a failed attempt to add nodes to your HyperStore cluster or a failed attempt to rebalance the cluster after adding nodes.

As the target `<host>` you can specify the hostname or IP address of any node in the cluster. The command retrieves cluster-wide information that is available from any node that belongs to the cluster.

IMPORTANT ! Do not use the `set` or `delete` options unless instructed to do so by Cloudian Support.

CMC Support for This Command

As an alternative to running `hsstool tmap` on the command line, you can run it through the CMC UI:



14.1.16.3. Command/Response Examples

In this first example a list of active token range map snapshot IDs is retrieved. These are token range map snapshots for which the associated rebalancing operation has not yet completed.

```
# /opt/cloudian/bin/hsstool -h cloudian-node1 tmap list
061e9190-6d62-429d-85dc-e4baec07f49e Wed Jun 19 15:26:07 EDT 2024
22351e62-e921-4130-afaf-1ca3183046e1 Thu Jun 20 14:17:04 EDT 2024
1c84d57e-dccb-47b0-af6d-015e823eab07 Wed Jun 19 21:42:42 EDT 2024
9e096b8f-5914-42fc-8dc7-063de85deb4a Thu Jun 20 06:56:17 EDT 2024
```

In this next example, a token range map snapshot is retrieved (this is from a different system and is not one of the snapshots listed in the first example). The map is in JSON format. The response is very large, and is truncated below. In practice, if you use this command you should redirect the output to a text file.

```
# /opt/cloudian/bin/hsstool -h cloudian-node1 trmap show fbcd5be-7c15-40a4-be59-955df70e7fb2
{
  "id" : "fbcd5be-7c15-40a4-be59-955df70e7fb2",
  "version" : "1",
  "timestamp" : 1477266709616,
  "rebalance" : "REQUIRED",
  "policies" : {
    "5871e62dae4ccfd618112ca3403b3251" : {
      "policyId" : "5871e62dae4ccfd618112ca3403b3251",
      "keyspaceName" : "UserData_5871e62dae4ccfd618112ca3403b3251",
      "replicationScheme" : {
        "DC2" : "1",
        "DC1" : "1",
        "DC3" : "1"
      },
      "ecScheme" : null,
      "ecMap" : null,
      "replicasMap" : [ {
        "left" : 163766745962987615139826681359528839019,
        "right" : 164582023203604297970082254372849331112,
        "endPointDetails" : [ {
          "endpoint" : "10.10.10.114",
          "datacenter" : "DC3",
          "rack" : "RAC1"
        }, {
          "endpoint" : "10.10.10.115",
          "datacenter" : "DC2",
          "rack" : "RAC1"
        }, {
          "endpoint" : "10.10.10.111",
          "datacenter" : "DC1",
          "rack" : "RAC1"
        } ]
      }, {
        "left" : 6341660663762096831290541188712444913,
        "right" : 6449737778660216877727472971404857143,
        "endPointDetails" : [ {
          ...
          ...
        } ]
      }
    ]
  }
}
```

Response Items

id

System-generated unique identifier of this token range map snapshot.

version

Version of the token range map snapshot. This integer is incremented each time a new snapshot is created.

timestamp

Timestamp indicating when the token range map snapshot was created.

rebalance

Status of the *hsstool rebalance* operation in regard to this token range map snapshot, such as **REQUIRED** or **COMPLETED**. Each time you add a node to your system (using the CMC's function for adding a node, in the **Data Centers** page), the system automatically generates a token range snapshot. After adding a node, you then must run *hsstool rebalance* on the new node (which you can do from the CMC's **Nodes Advanced** page). The rebalance operation utilizes the token range map snapshot. For complete instructions on adding nodes, see ["Adding Nodes" \(page 579\)](#).

policies

This marks the beginning of the per-policy token range map information. The token range map will have separate token range map information for each of your storage policies.

policyId

System-generated unique identifier of a storage policy. Note that this ID appears three times: at the outset of the policy block, then again as the *policyId* attribute, then again within the *keyspaceName*.

keyspaceName

Name of the Cassandra keyspace in which object metadata is stored for this storage policy. The name is in format *UserData_<policyId>*.

replicationScheme

Specification of the replication scheme, if this policy block is for a replication storage policy. In the example the policy's replication scheme calls for one replica in each of three data centers.

ecScheme

Specification of the erasure coding scheme, if this policy block is for an erasure coding storage policy. In the example, the policy block is for a replication policy so the *ecScheme* value is null.

ecMap

Content of the token range map for the policy scheme, if this policy block is for an erasure coding storage policy. In the example, the policy block is for a replication policy so the *ecMap* value is null.

replicasMap

Content of the token range map for the policy scheme, if this policy block is for a replication storage policy. The map consists of lists of endpoints per token range.

left

Token at the low end of the token range (exclusive). This is from the consistent hashing space of 0 to 2^{127} from which HyperStore generates tokens for the purpose of allocating data across the cluster.

right

Token at the high end of the token range (inclusive). This is from the consistent hashing space of 0 to 2^{127} from which HyperStore generates tokens for the purpose of allocating data across the cluster.

endPointDetails

Endpoint mapping information for this particular token range, for this storage policy. This is a list of endpoints (nodes), with each endpoint identified by IP address as well as data center name and rack name. The number of endpoints per token range will depend on the storage policy scheme. In the example the policy is a 3X replication policy, so there are three endpoints listed for each token range. Objects for which the object token

(based on a hash of the bucket name / object name combination) falls into this token range will have replicas placed on each of these nodes.

14.1.17. hsstool whereis

Subjects covered in this section:

- ["Command Syntax" \(page 938\)](#)
- ["Usage Notes" \(page 939\)](#)
- ["Command/Response Examples" \(page 940\)](#)

14.1.17.1. Command Syntax

Synopsis

```
# hsstool [-h <host>] whereis <bucket>/<object> [-v <version>] [-ck] [-r] [-a]
```

Options

<bucket>/<object> *Bucket and object name*

(Mandatory unless using the `-a` option) Bucket name, followed by a forward slash, followed by the full object name (including "folder path", if any). For example, `mybucket/file1.txt` or `mybucket/Videos/Vacation/Italy_2024-06-27.mpg`.

If the object name has spaces in it, enclose the *bucket/object* name pair in quotes. For example, `"mybucket/big document.doc"`.

The *bucket/object* name is case-sensitive.

Note In the CMC UI implementation of this command, you enter the bucket name and the full object name (including folder path) in separate fields. For example, bucket name `mybucket` and full object name `Videos/Vacation/Italy_2024-06-27.mpg`.

-v <version> *Object version*

(Optional) If versioning is enabled on the bucket that contains the object, you can optionally specify the version ID of a particular version of the object. Version IDs are system-generated hexadecimal values (for example, `fe1be647-5f3b-e87f-b433-180373cf31f5`). If versioning has been used for the object but you do not specify a version ID, location information will be retrieved for the most recent version of the object.

-ck *Check for object corruption*

(Optional) Use the `-ck` option if you want the *whereis* results to indicate whether any of the target object's replicas or fragments are corrupted. When you use this option, the *whereis* operation detects corruption by comparing a freshly computed MD5 hash of each replica or fragment to what the MD5 hash of each replica or fragment should be, based on the object's stored metadata.

-r *Show replicas or fragments to be rebalanced*

(Optional) Use the `-r` option if you have added new nodes to your cluster, and your rebalance operations are not yet completed. When you use the `-r` option, in the location detail section of the *whereis* command response, a replica or erasure coded fragment that will be relocated during the rebalance operations will be indicated as:

```
[new host] <- [old host]
```

where the new host will be one of your newly added nodes and the old host will be one of your pre-existing nodes. For example, in the case of an erasure coded fragment that will be rebalanced:

```
[dc-1 null 10.50.40.18 rocky08] <- [dc-1 10.50.40.11 rocky01] fragment detail...
```

Note The CMC does not support the `-r` option. To use this option you need to use `hsstool whereis` on the command line.

`-a` Write location info for all objects to a log file

(Optional) Use the `-a` option if you want a full list of **all replicas and fragments of all objects in the entire service region**. This information will be written to a log file. For more information on the log file, see the section that follows the command/response example.

If you use the `-a` option do not use the `<bucket/object>` parameter or the `-v <version>` parameter. Run the command simply as `hsstool -h <host> whereis -a`

Note The CMC does not support the `-a` option. To use this option you need to use `hsstool whereis` on the command line.

14.1.17.2. Usage Notes

This [hsstool](#) command returns the current storage location of each replica of a specified S3 object (or in the case of erasure coded objects, the location of each of the object's fragments). The command response also shows the specified object's metadata such as last modified timestamp and object digest.

Since this command returns information from across the cluster, you can specify any node as the target `<host>`.

CMC Support for This Command

As an alternative to running `hsstool whereis` on the command line, you can run it through the CMC UI:

The screenshot shows the Cloudian CMC interface. The top navigation bar includes 'Cluster' (highlighted with a red box and a red circle with '1'), 'Alerts', 'Admin', and 'Help'. The sub-navigation bar includes 'Data Centers', 'Nodes' (highlighted with a red box and a red circle with '2'), 'Cluster Config', 'Storage Policies', 'KMIP Profiles', 'Repair Status', and 'Operation Status'. The 'Nodes' tab is active, and the 'Advanced' sub-tab is selected (highlighted with a red box and a red circle with '3'). The 'Command Type' dropdown is set to 'Info' (circled in red). The 'hsstool Command' dropdown is set to 'whereis' (circled in red). The 'Target Node' dropdown is set to 'jarvis'. There are empty input fields for 'Bucket', 'Object', and 'Version (optional)'. A description at the bottom reads: 'Locate all of a given object's replicas or erasure coded fragments.' An 'EXECUTE' button is at the bottom right.

Log File for "whereis -a"

When you use the *whereis -a* command, information about all replicas and erasure coded fragments of all objects in the entire service region is written to a log file. The log file is written on the HyperStore host that you connect to when you run *whereis -a*, and by default the log file path is:

```
/var/log/cloudian/whereis.log
```

In the *whereis.log* file, the start and completion of the output from a single run of *whereis -a* is marked by "START" and "END" timestamps. Within those timestamps, the output is organized by user. The start of output for a particular user is marked by "#user:<canonical UID>". This line is then followed by lines for the user's buckets and objects, with the same object detail information as described in the *whereis* command results documentation above. Users who do not have any buckets will not be included in the log file.

The output of multiple runs of *whereis -a* may be written to the same log file, depending on the size of the output. Because the output of *whereis -a* may be very large, it's also possible that the output of a single run may be spread across multiple log files, if maximum file size is reached and log rotation occurs.

By default this log is rotated if it reaches 10MB in size or at the end of the day, whichever occurs first. The oldest rotated *whereis* log file is automatically deleted if it reaches 180 days in age or if the aggregate size of all rotated *whereis* log files (after compression) reaches 100MB.

14.1.17.3. Command/Response Examples

The first *whereis* command example below retrieves location information for an object named "Guide.pdf". This is from a single-node HyperStore system, so there is just one replica of the object. The location detail information for the object replica is truncated in this example.

```
# hsstool -h sirius whereis bucket2/Guide.pdf
Key: bucket2/Guide.pdf
Policy ID: c4a276180b0c99346e2285946f60e59c
Version: null
Compression: NONE
Create Time: 2017-02-20T16:38:09.783Z
Last Modified: 2017-02-20T16:38:09.783Z
Last Access Time: 2017-02-20T16:38:10.273Z
Size: 7428524
Type: REPLICAS
Region: region1
[DC1 10.50.10.21 sirius] bucket2/Guide.pdf file://10.50.10.21:/var/lib/cloudian/hsfs/1L1t...
```

The second *whereis* command example retrieves location information for an object named "obj1.txt". This is from a two data center HyperStore system, using replicated 2+1 erasure coding. The location detail information for each found fragment is truncated in this example.

```
# hsstool -h cloudian-node3 whereis bucket1/obj1.txt
Key: bucket1/obj1
Policy ID: 1d140a90b17285cf2d1502cbd424d621
Version: null
Compression: NONE
Create Time: 2024-01-29T23:07:52.626Z
Last Modified: 2024-01-29T23:07:52.626Z
Last Access Time: 2024-01-29T23:07:52.626Z
Size: 11231
Type: EC
```



```

Region: region1
6 out of 6 fragments were found
[DC1 2 10.100.186.47 cloudian-node3] bucket1/obj1.txt ec://10.100.186.47:/var/lib/cloudian/ec/...
[DC1 3 10.100.76.90 cloudian-node4] bucket1/obj1.txt ec://10.100.76.90:/var/lib/cloudian/ec/...
[DC2 1 10.100.61.17 cloudian-node6] bucket1/obj1.txt ec://10.100.61.17:/var/lib/cloudian/ec/...
[DC2 2 10.100.80.35 cloudian-node7] bucket1/obj1.txt ec://10.100.80.35:/var/lib/cloudian/ec/...
[DC2 3 10.100.218.23 cloudian-node8] bucket1/obj1.txt ec://10.100.218.23:/var/lib/cloudian/ec/...
[DC1 1 10.100.186.42 cloudian-node1] bucket1/obj1.txt ec://10.100.186.42:/var/lib/cloudian/ec/...

```

Note For objects for which the Type is "TRANSITIONED" (auto-tiered), the response will also include a *URL* field.

Response Items

Key

Key that uniquely identifies the S3 object, in format *<bucketname>/<objectname>*. For example, *bucket1/Documents/Meetings_2024-06-27.docx*.

PolicyID

System-generated identifier of the storage policy that applies to the bucket in which this object is stored.

Version

Object version, if versioning has been used for the object. Versions are identified by timeuuid values in hexadecimal format. If versioning has not been used for the object, the Version field displays "null".

Compression

Type of server-side compression applied to the object, if any. Possible values are NONE, SNAPPY, ZLIB, or LZ4. The type of compression applied depends on the storage policy used by the bucket. Each storage policy has its own configuration as to whether compression is used and the compression type.

Create Time

Timestamp for the original creation of the object. Format is ISO 8601 and the time is in Coordinated Universal Time (UTC).

Last Modified

Timestamp for last modification of the object. Format is ISO 8601 and time is in UTC.

Last Access Time

Timestamp for last access of the object. 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). Format is ISO 8601 and time is in UTC.

Size

The object's size in bytes.

Type

One of:

- REPLICAS — The object is replicated in the HyperStore File System (HSFS).
- EC — The object is erasure coded in the HSFS.
- TRANSITIONED — The object has been transitioned ([auto-tiered](#)) to a different storage system such as Amazon S3.

Region

The HyperStore service region in which the object is stored.

URL (transitioned objects only)

This field appears only in the case of TRANSITIONED objects. For such objects, this field shows the URL that identifies the location of the object in the tiering destination system. For example:

```
http://s3.amazonaws.com/bucket2.mdazyjgxnjyxndu2ody4mji1nty3/notes.txt
```

In this example, the tiering destination is Amazon S3; the bucket name in the destination system is *bucket2.mdazyjgxnjyxndu2ody4mji1nty3* (which is the HyperStore source bucket name — *bucket2* in this case — appended by a 28 character random string); and the object name is *notes.txt*. Note that the URL field will specify the transfer protocol as *http*, whereas to actually access the object in the destination system the protocol would typically be *https*.

Fragment count summary (erasure coded objects only)

For erasure coded objects, the location detail section of the *hsstool whereis* response starts with a line stating the number of fragment locations that have been identified for the object -- for example "6 out of 6 fragments were found" for an object for which there are supposed to be 6 fragments. This statement does not necessarily mean that all the fragments were found in their expected locations -- it means only that 6 expected fragment locations were identified. If the fragment itself is found at the expected location, in the location detail section the last field of the entry for that location will show the actual size of the found fragment. **If the fragment is missing from its expected location, the location entry's last field will show a size of "-1".**

Location detail (including indicator of whether the replica or fragment is found at the expected location)

For objects stored locally (objects that are not of type TRANSITIONED), the lower part of the response shows the location of **each object replica** (for replicated objects) or of **each erasure coded object fragment** (for EC objects).

For HSFS replicated objects each location is specified as:

```
[<datacenter> <IP-address> <hostname>] <bucket>/<objectname>  
file://<IP-address>:<mountpoint>/hsfs/<base62-encoded-vNode-token>/<policyid>/  
<000-255>/<000-255>/<filename> <last-modified> <version> <digest> <size>
```

For erasure coded object fragments each location is specified as:

```
[<datacenter> <key-suffix-digit> <IP-address> <hostname>] <bucket>/<objectname>  
ec://<IP_address>:<mountpoint>/ec/<base62-encoded-vNode-token>/<policyid>/  
<000-255>/<000-255>/<filename> <last-modified> <version> <digest> <size>
```

- For EC objects only, the *<key_suffix_digit>* at the beginning of each location is a digit that the system generates and uses to ensure that each fragment goes to a different node.
- The *<base62-encoded-vNode-token>* is a base-62 encoding of the token belonging to the vNode to which the object instance or fragment is assigned.
- The *<policyid>* segment is the unique identifier of the storage policy applied to the bucket in which the object is stored.

- The two `<000-255>` segments of the path are based on a hash of the `<filename>`, normalized to a 255*255 number.
- The `<filename>` is a dot-separated concatenation of the object's hash token and the object's Last Modified Time timestamp. The timestamp is formatted as `<UnixTimeMillis><6digitAtomicCounter>-<nodeIPAddrHex>` (the last element is the IP address -- in hexadecimal format -- of the S3 Service node that processed the object upload request). Note: For objects last modified prior to HyperStore version 6.1, the timestamp is simply Unix time in milliseconds. This was the timestamp format used in HyperStore 6.0.x and older..
- If the replica or fragment is found at the expected location, the `<size>` field shows the size of the replica or fragment . If the replica or fragment is not found at the expected location, the size field shows "-1". **A size of "-1" indicates that the replica or fragment is missing from a location where it is supposed to be.** If the digest is also missing, the digest will be absent from the location detail entry.

Note For multipart objects (large objects uploaded via the S3 multipart upload method), storage location detail is shown **for each part**.

14.2. `hsctl`

Using the versatile command line tool `hsctl` is one of the primary ways in which you will manage your HyperStore system. When logged into the Config Controller node either as `root` or as a HyperStore Shell user, you can run this tool from any directory with no need to specify a path. For details on some of the `hsctl` tool's major functions see:

- ["Using `hsctl` to Control HyperStore Configuration"](#) (page 693)
- ["Using `hsctl` to Start, Stop, or Check Status of Services \(Scope: Node, Data Center, Region, or System\)"](#) (page 91)
- TLS Certificate management:
 - ["Generating a New Self-Signed Certificate for a Service"](#) (page 178)
 - ["Generating a Certificate Signing Request \(CSR\)"](#) (page 180)
 - ["Importing a CA-Signed Certificate"](#) (page 182)
 - ["Adding a Private Root CA's Certificate to HyperStore's Truststore"](#) (page 185)
- File Service installation and operations:
 - "Installing the HyperStore File Service" in the *Clouddian HyperStore Installation Guide*
 - ["Replacing a Node in a Three-Node File Service Pool"](#) (page 647)
 - ["Backing Up and Restoring a Single-Node File Service Pool"](#) (page 649)

For information about additional functionality provided by `hsctl` you can view the `hsctl` top-level inline help:

```
hsctl --help
```

And also inline help for particular `hsctl` commands, for example:

```
hsctl logs --help
```

```
hsctl patch --help
```

```
hsctl cmd --help
```

Note The [HyperStore Shell](#) does not support the *hsctl cmd* command. It does support all other *hsctl* commands.

14.3. cloudianInstall.sh

The *cloudianInstall.sh* tool (also known as "the installer") serves several purposes including:

- Installation of a HyperStore cluster (for detail see [Installing a New HyperStore System](#))
- Implementing advanced, semi-automated system configuration changes (for detail see [Installer Advanced Configuration Options](#))
- Pushing configuration file edits to the cluster and restarting services to apply the changes (for detail see "Pushing Configuration File Edits to the Cluster and Restarting Services" in the **Cloudian HyperStore Administrator's Guide**)

The *cloudianInstall.sh* tool is in your installation staging directory on your Config Controller node. To perform advanced configurations, or to push configuration file changes to the system and restart services, you would launch the tool simply like this, without using additional command line options:

```
# ./cloudianInstall.sh
```

14.3.1. Command Line Options When Using cloudianInstall.sh for HyperStore Cluster Installation

To perform a HyperStore cluster installation you typically would launch the script either like this:

```
# ./cloudianInstall.sh
```

Or like this if you are not using your DNS environment to resolve HyperStore service endpoints and you want to use the bundled tool *hsDNS* instead (which is not appropriate for production systems):

```
# ./cloudianInstall.sh configure-hsdns
```

However the script does support additional command line options. The syntax is as follows:

```
# ./cloudianInstall.sh [-k <ssh-private-key-filename>] [-d] [-h]
[configure-hsdns] [configure-qlc-rdb] [no-hosts] [no-firewall] [force] [uninstall]
```

Note If you use multiple options, on the command line place options that start with a "-" (such as *-k <ssh-private-key-filename>* or *-d*) before options that do not (such as *no-hosts* or *configure-hsdns*).

If you are using the HyperStore Shell

If you are using the HyperStore Shell (HSH) as a Trusted user, from any directory on the Config Controller node you can launch the installer with this command:

```
$ hspkg install
```

The installer's options are the same regardless of whether it is launched from the HSH command line or the OS command line.

Note After using the installer, exit the installer when you're done. Do not leave it running. Certain automated system tasks invoke the installer and cannot do so if it is already running.

The supported command line options are:

- `-k <ssh-private-key-filename>` — The Config Controller node uses SSH for secure communication with the rest of your HyperStore installation nodes. By default the install script automatically creates an SSH key pair for this purpose. But if instead you would prefer to use your own existing SSH key pair for this purpose, you can use the installer's `-k <ssh-private-key-filename>` option to specify the name of the private key file (including the full path to the file). When you run the install script it will copy the private key and corresponding public key to the installation staging directory, and in the staging directory the key file will be renamed to `cloudian-installation-key`. Then from the staging directory, the public key file `cloudian-installation-key.pub` will be copied to each node on which you are installing HyperStore.

Note Keys created with default `ssh-keygen` behavior will not work with *JSch* (a Java implementation of SSH2 that HyperStore uses). Instead use these options for key generation:

```
ssh-keygen -t ecdsa -m pem -b 256
```

- `-d` — Turn on debugging output.
- `-h` — Display usage information for the install tool. This option causes the tool to print a usage message and exit.

Note This usage information mentions more command line options than are described here in this Help topic. This is because the usage information includes installer options that are meant for HyperStore internal system use, such as options that are invoked by the CMC when you use the CMC to add nodes to your cluster or remove nodes from your cluster. You should perform such operations through the CMC, not directly through the installer. The CMC implements automations and sanity checks beyond what is provided by the install script alone.

- `configure-hsdns` — Include this command-line option if you want to use HyperStore's hsDNS component to resolve HyperStore service endpoints within the cluster (and also externally if you set up DNS delegation). For more information about hsDNS, in the "Service Features" section of your *Cloudian HyperStore Administrator's Guide*, see "hsDNS".
- `configure-qlc-rdb` — Use this command-line option if you are using Quad-Level Cell (QLC) SSDs as your object data drives. Using this option results in the commit logs from RocksDB -- used for object digests -- being stored on the Cassandra (Metadata DB) drives rather than on the data drives (where they are stored in traditional non-QLC deployments). This helps to reduce frequent-write, small-file activity on your QLC drives.
- `no-hosts` — Use this option if you do not want the install tool to append entries for each HyperStore host on to the `/etc/hosts` file of each of the other HyperStore hosts. By default the tool appends to these files so that each host is resolvable to the other hosts by way of the `/etc/hosts` files.
- `no-firewall` — If this option is used, the HyperStore firewall will **not** be enabled upon HyperStore installation. By default the HyperStore firewall will be enabled upon completion of a fresh HyperStore installation. For more information about the HyperStore firewall see the "HyperStore Firewall" section in the *Cloudian HyperStore Administrator's Guide*.

- *force* — By default the installer performs certain prerequisite checks on each node on which you are installing HyperStore and aborts the installation if any of your nodes fails a check. By contrast, if you use the *force* option when you launch the installer, the installer will output warning messages to the terminal if one or more nodes fails a prerequisite check but the installation will continue rather than aborting. The prerequisite checks that this feature applies to are:
 - CPU has minimum of 8 cores
 - RAM is at least 128GB
 - System Architecture is x86 64-bit
 - SELinux is disabled
 - firewalld is disabled
 - iptables is not running

Note If you specify the *force* option when running the installer, the *force* option will "stick" and will be used automatically for any subsequent times the installer is run to install additional nodes (such as when you do an "Add Node" operation via the Cloudian Management Console, which invokes the installer in the background). To turn the *force* option off so that it is no longer automatically used when the installer is run to add more nodes, launch the installer and go to the Advanced Configuration Options. Then choose option **t** for **Configure force behavior** and follow the prompts.

Note Even if the *force* option is used the installer will abort if it detects an error condition on the host that will prevent successful installation.

- *uninstall* — If you use this option when launching the installer, the installer main menu will include an additional menu item -- "Uninstall Cloudian HyperStore".

```
Cloudian HyperStore(R) 8.0 Installation/Configuration
-----
0 ) Run Pre-Installation checks
1 ) Install Cloudian HyperStore
2 ) Cluster Management
3 ) Upgrade From a Previous Version
4 ) Advanced Configuration Options
5 ) Uninstall Cloudian HyperStore
6 ) Help
x ) Exit

Choice: █
```

Use this menu option only if you want to **delete the entire HyperStore system, on all nodes, including any metadata and object data** stored in the system. You may want to use this Uninstall Cloudian HyperStore option, for example, after completing a test of HyperStore -- if you do not want to retain the test system.

IMPORTANT ! Do not use this option to uninstall a single node from a HyperStore system that you want to retain (such as a live production system). For instructions on removing a node from a HyperStore system see the "Removing a Node" section in the *Cloudian HyperStore Administrator's Guide*.

14.4. system_setup.sh

The `system_setup.sh` tool is for setting up nodes on which you will install HyperStore software, either during initial cluster installation or during cluster expansion. For basic information about using `system_setup.sh`, on the Config Controller node change into the installation staging directory and run the following command:

```
# ./system_setup.sh --help
```

14.5. Redis Monitor Commands

The HyperStore Redis Monitor Service monitors Redis Credentials DB and Redis QOS DB cluster health and implements automatic failover of the master node role in each cluster. The Redis Monitor runs on two nodes, with one instance being the primary and the other being the backup. When you submit Redis Monitor commands **you must submit them to the primary node**, not the backup. To check which of your nodes is running the primary Redis Monitor, go to the CMC's [Cluster Information](#) page.

You can submit commands to the Redis Monitor primary host through the Redis Monitor CLI. A couple of the more useful commands can also be executed through the CMC.

- **To initiate a Redis Monitor CLI session**, use `netcat` to connect to port 9078 on the node on which the primary Redis Monitor is running:

```
# nc <redismon_primary_host> 9078
```

Specify the hostname or IP address (do not use 'localhost'). Once connected, you can then use any of the Redis Monitor CLI commands listed below. When you're done using Redis Monitor commands, enter **quit** to end your Redis Monitor CLI session and then enter **<ctrl>-d** to end your `netcat` session and return to the terminal prompt.

*If you are using the **HyperStore Shell***

Because 'nc' is not a supported command for the HyperStore Shell, you cannot initiate a Redis Monitor CLI session through the HyperStore Shell. Your only access to Redis Monitor commands is through the CMC as noted below.

- **To access Redis Monitor commands in the CMC**, go to the [Node Advanced](#) page and from the "Command Type" drop-down list select "Redis Monitor Operations". Note that only a small number of Redis Commands are available through the CMC (specifically `get cluster` and `set master`).

The Redis Monitor supports the following commands:

14.5.1. get cluster

get cluster

Use this command to retrieve basic status information that the Redis Monitor currently has for a specified Redis DB cluster. The cluster status information includes:

- The identity of the master node within the cluster
- Whether monitoring of the cluster by the Redis Monitor is enabled
- Whether the sending of cluster status notifications to clients of the cluster is enabled
- A list of cluster member nodes, with an indication of the status (UP/DOWN) of each node
- A list of cluster clients (which write to and/or read from this Redis database), with an indication of the status (UP/DOWN) of each client. The clients will include S3 Service instances (identified by JMX listening socket <host>:19080), IAM Service instances (<host>:19084), Admin Service instances (<host>:19081), and HyperStore Service instances (<host>:19082). These are the clients to which the Redis Monitor sends notifications regarding the Redis cluster's status.
- "state" information that includes a timestamp indicating the last date and time that the master role status was updated (if there have been any fail-overs of the master role, the timestamp is the time of the last fail-over -- otherwise, it's the time of the last start-up of the Redis Monitor)

14.5.1.1. Command Line Syntax

```
get cluster redis.credentials|redis.qos.<region>
```

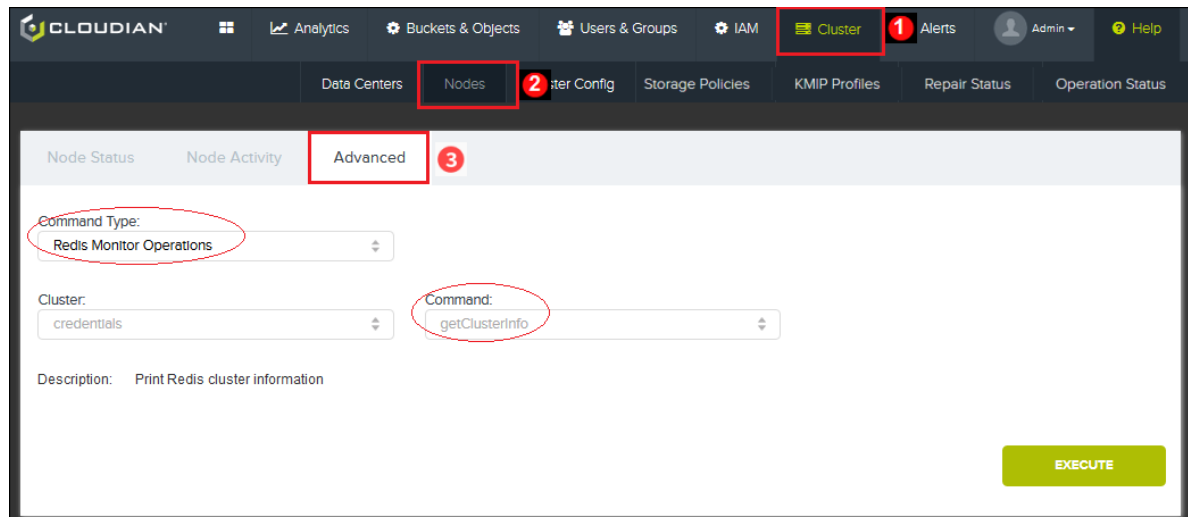
Note For this and all other Redis Monitor commands, the Redis QOS cluster identifier includes the name of the service region in which the cluster resides. This is necessary since in a multi-region HyperStore system each region has its own Redis QoS cluster. By contrast the Redis Credentials cluster, since it is global (extending across all service regions), does not include a region name in its identifier.

Example:

```
# nc 10.50.20.12 9078
get cluster redis.credentials
OK master: store1(10.50.20.1):6379, monitoring: enabled, notifications: enabled
nodes: [[store1(10.50.20.1):6379,UP,master], [store4(10.50.20.4):6379,UP,slave],
[store5(10.50.20.5):6379,UP,slave]]
clients: [[store1(10.50.20.1):19080,UP,store1], [store2(10.50.20.2):19080,UP,store1],
[store3(10.50.20.3):19080,UP,store1], [store4(10.50.20.4):19080,UP,store1],
[store5(10.50.20.5):19080,UP,store1], [store6(10.50.20.6):19080,UP,store1],
[store1(10.50.20.1):19081,UP,store1], [store2(10.50.20.2):19081,UP,store1],
[store3(10.50.20.3):19081,UP,store1], [store4(10.50.20.4):19081,UP,store1],
[store5(10.50.20.5):19081,UP,store1], [store6(10.50.20.6):19081,UP,store1],
[store1(10.50.20.1):19082,UP,store1], [store2(10.50.20.2):19082,UP,store1],
[store3(10.50.20.3):19082,UP,store1], [store4(10.50.20.4):19082,UP,store1],
[store5(10.50.20.5):19082,UP,store1], [store6(10.50.20.6):19082,UP,store1]]
state: redis.credentials: master= 10.50.20.1 updatetime= Fri Sep 20 16:17:23 PDT 2024
```

14.5.1.2. CMC UI

Path: **Cluster** → **Nodes** → **Advanced**



14.5.2. get master

get master

Use this command to retrieve from the Redis Monitor the identity of the current master node within a specified Redis cluster.

14.5.2.1. Command Line Syntax

```
get master redis.credentials|redis.qos.<region>
```

Example:

```
# nc 10.50.20.12 9078
get master redis.qos.region1
OK store2(10.50.20.2):6380
```

14.5.3. get nodes

get nodes

Use this command to retrieve from the Redis Monitor a list of all current members of a specified Redis cluster. The command response also indicates the status (UP/DOWN) and role (master/slave) of each member node.

14.5.3.1. Command Line Syntax

```
get nodes redis.credentials|redis.qos.<region>
```

Example:

```
$ nc 10.50.20.12 9078
get nodes redis.credentials
OK [[store1(10.50.20.1):6379,UP,master], [store4(10.50.20.4):6379,UP,slave],
[store5(10.50.20.5):6379,UP,slave]]]
```

14.5.4. get clients

get clients

Use this command to retrieve from the Redis Monitor a list of clients of a specified Redis cluster (client nodes that write to and/or read from the Redis database). These are the clients to which the Redis Monitor sends notifications regarding the Redis cluster's status. For example, if the cluster's master role changes from one node to another, the Redis Monitor will notify these clients of the change.

The clients will include S3 Service instances (identified by JMX listening socket <host>:19080), IAM Service instances (<host>:19084), Admin Service instances (<host>:19081), and HyperStore Service instances (<host>:19082). The command response also indicates the status (UP/DOWN) of each client, and for each client it shows which node the client thinks is the Redis master.

14.5.4.1. Command Line Syntax

```
get clients redis.credentials|redis.qos.<region>
```

Example:

```
# nc 10.50.20.12 9078
get clients redis.credentials
OK [[[store1(10.50.20.1):19080,UP,store1], [store2(10.50.20.2):19080,UP,store1],
[store3(10.50.20.3):19080,UP,store1], [store4(10.50.20.4):19080,UP,store1],
[store5(10.50.20.5):19080,UP,store1], [store6(10.50.20.6):19080,UP,store1],
[store1(10.50.20.1):19081,UP,store1], [store2(10.50.20.2):19081,UP,store1],
[store3(10.50.20.3):19081,UP,store1], [store4(10.50.20.4):19081,UP,store1],
[store5(10.50.20.5):19081,UP,store1], [store6(10.50.20.6):19081,UP,store1],
[store1(10.50.20.1):19082,UP,store1], [store2(10.50.20.2):19082,UP,store1],
[store3(10.50.20.3):19082,UP,store1], [store4(10.50.20.4):19082,UP,store1],
[store5(10.50.20.5):19082,UP,store1], [store6(10.50.20.6):19082,UP,store1]]]
```

In the above example, "store1" is the current Redis Credentials master node. All the clients correctly have this information.

14.5.5. enable monitoring

enable monitoring

Use this command to enable monitoring of a specified Redis cluster by the Redis Monitor.

Note Monitoring is enabled by default. This command is relevant only if you have previously disabled monitoring.

14.5.5.1. Command Line Syntax

```
enable monitoring redis.credentials|redis.qos.<region>
```

Example:

```
# nc 10.50.20.12 9078
enable monitoring redis.credentials
OK enabled
```

14.5.6. disable monitoring

disable monitoring

Use this command if you want to temporarily disable Redis Monitor's monitoring of a specified Redis cluster — for example if you are performing maintenance work on the Redis cluster. (You can subsequently use the *enable monitor* command re-enable monitoring of that cluster.)

14.5.6.1. Command Line Syntax

```
disable monitoring redis.credentials|redis.qos.<region>
```

Example:

```
# nc 10.50.20.12 9078
disable monitoring redis.qos.region1
OK disabled
```

14.5.7. enable notifications

enable notifications

Use this command to enable Redis Monitor's sending of notifications to the clients of a specified Redis cluster. (The clients are the S3 Service instances, IAM Service instances, Admin Service instances, and HyperStore Service instances that write to and/or read from that Redis cluster).

The Redis Monitor sends notifications to inform clients of the identity of the Redis cluster's master node, in either of these circumstances:

- The Redis master role has switched from one host to another. (This could happen if the original master goes down and Redis Monitor detects this and fails the master role over to one of the slave nodes; or if an operator uses the Redis Monitor CLI to move the master role from one node to another).
- The Redis Monitor in its regular polling of cluster clients' status detects that one of the clients has incorrect information about the identity of the Redis cluster master node. In this case the Redis Monitor notifies the client to give it the correct information.

Note Notifications are enabled by default. This operation is relevant only if you have previously disabled notifications using the *disable notifications* command.

14.5.7.1. Command Line Syntax

```
enable notifications redis.credentials|redis.qos.<region>
```

Example:

```
# nc 10.50.20.12 9078
enable notifications redis.credentials
OK enabled
```

14.5.8. disable notifications

disable notifications

Use this command to temporarily disable Redis Monitor's sending of Redis cluster status notifications to the clients of that cluster. For more information on the notification feature see ["enable notifications"](#) (page 951).

14.5.8.1. Command Line Syntax

```
disable notifications redis.credentials|redis.qos.<region>
```

Example:

```
# nc 10.50.20.12 9078
disable notifications redis.qos.region1
OK disabled
```

14.5.9. set master

set master

Use this command to assign the Redis master role to a different node within a specified Redis cluster. **The node to which you assign the master role must be one of the current slaves within the same Redis cluster.**

The Redis master node within a cluster is the node to which Redis clients submit writes. The writes are asynchronously replicated to the slave(s) within that cluster. Redis clients read from the slave(s).

An example of when you would move the Redis master role is if you want to remove the current Redis master host from your cluster.

Using this command is part of a broader procedure for moving a Redis master role to a slave. For the full procedure including the use of this command within the procedure, see ["Move the Credentials DB Master Role or QoS DB Master Role"](#) (page 627).

14.5.9.1. Command Line Syntax

```
set master redis.credentials|redis.qos.<region> [<host:redisPort>]
```

Example:

```
# nc 10.50.20.12 9078
set master redis.credentials store5:6379
OK set new master store5(10.50.20.5):6379
```

Note If you do not specify a *<host:redisPort>* value, the Redis Monitor chooses a slave node at random (from within the cluster) to elevate to the master role.

14.5.9.2. CMC UI

Path: **Cluster** → **Nodes** → **Advanced**

In the CMC UI, use the "Hostname" field to specify the host to which you want to move the Redis master role.

14.5.10. add node

add node

Use this command to add a Redis node to the list of nodes that Redis Monitor is monitoring, for a specified Redis cluster. This would be if you have used the installer (*cloudianInstall.sh* in the installation directory on your Config Controller node) to activate Redis on a HyperStore node that wasn't previously running Redis. In this circumstances you have two options to make Redis Monitor aware of the new member of the Redis cluster:

- Restart Redis Monitor
- OR
- Use this command.

14.5.10.1. Command Line Syntax

```
add node redis.credentials|redis.qos.<region> <host:redisPort>
```

Example:

```
# nc 10.50.20.12 9078
add node redis.qos.region1 store3:6380
OK added node store3(10.50.20.3):6380 to redis
```

14.5.11. add client

add client

This command can be used to add a new S3 Service, IAM Service, Admin Service, and/or HyperStore Service node to the list of clients to which Redis Monitor will send notifications regarding the status of a specified Redis cluster.

In normal circumstances you should not have to use this command. If you add a new node to your HyperStore cluster (as described in ["Adding Nodes" \(page 579\)](#)), the system automatically makes Redis Monitor aware of the new clients of the Redis Credentials and Redis QoS clusters.

If you do use this command, add only one client per command run. A client is identified by its JMX socket (for example "cloudian12:19080" for an S3 Service instance running on host cloudian12, or "cloudian12:19081" for an Admin Service instance running on host cloudian12).

14.5.11.1. Command Line Syntax

```
add client redis.credentials|redis.qos.<region> <host:JMXport>
```

Example:

```
# nc 10.50.20.12 9078
add client redis.credentials store7:19080
OK added client store7(10.50.20.7):19080 to redis
```

14.5.12. test dc partition

test dc partition

If your HyperStore system includes multiple data centers (DCs), you can use this command to check whether or not there is a DC partition in the specified Redis cluster. A Redis cluster is considered to have a DC partition if the Redis Monitor -- from its location within one of the DCs -- cannot reach any of that cluster's Redis nodes or any of that cluster's Redis clients (S3, IAM, Admin, HyperStore) in one of the other DCs.

Note The Redis Monitor **automatically** checks for a DC partition once every five seconds, and if a partition is detected an alert is logged in *cloudian-redismon.log* on the Redis Monitor node and is displayed in the CMC's **Alerts** page. So under normal circumstances you should not need to manually trigger a DC partition check by using this command.

14.5.12.1. Command Line Syntax

```
test dc partition redis.credentials|redis.qos.<region>
```

Example:

```
# nc 10.50.20.12 9078
test dc partition redis.credentials
OK
Redis cluster, DC: us-east-2a # of Nodes: 1 Failure nodes: 0
Client side, DC: us-east-2a # of Clients: 9 Failure Clients: 0
Redis cluster, DC: us-east-1a # of Nodes: 3 Failure nodes: 0
Client side, DC: us-east-1a # of Clients: 16 Failure Clients: 0
Redis cluster, DC: us-east-1b # of Nodes: 2 Failure nodes: 0
Client side, DC: us-east-1b # of Clients: 12 Failure Clients: 0
Has Partition: false
```

14.5.13. test split brain

test split brain

You can use this command to check whether or not there is a "split brain" condition in the specified Redis cluster. A Redis cluster is considered to have a "split brain" if two different Redis nodes within the cluster have the master role at the same time. For proper cluster operation and metadata integrity, each Redis cluster should have just one master node at any given point in time. A split brain can occur, for instance, if the master role fails over from one node to another, and then the original master node comes back up and -- due to connectivity failures or some other system problem -- starts acting as master again rather than rejoining the cluster as a slave.

Note The Redis Monitor **automatically** checks for a "split brain" condition once every five seconds, and if a split brain condition is detected an alert is logged in *cloudian-redismon.log* on the Redis Monitor node and is displayed in the CMC's **Alerts** page. So under normal circumstances you should not need to manually trigger a split brain check by using this command.

14.5.13.1. Command Line Syntax

```
test split brain redis.credentials|redis.qos.<region>
```

Example #1:

```
# nc 10.50.20.12 9078
test split brain redis.credentials
OK
Number of Master in cluster redis.credentials: 1
Has No Brain: false
Has Split Brain: false
```

Example #2:

```
# nc 10.50.20.12 9078
test split brain redis.credentials
OK
Number of Master in cluster redis.credentials: 2
Has No Brain: false
Has Split Brain: true
```

14.5.14. disable dc partition monitoring

disable dc partition monitoring

This command affects how the Redis Monitor behaves **after it has detected a data center partition** in a Redis cluster. (For a description of how the Redis Monitor determines that a Redis cluster is in a DC partition condition, see **"test dc partition"** (page 954)).

If DC partition monitoring is **enabled**, then in the circumstance where DC partition has been detected and the Redis master node is in the unreachable DC, the Redis Monitor will continue with its normal master role monitoring and managing behavior by **promoting a reachable slave in a different DC to the master role** (in other words, failover of the master role will be executed).

If DC partition monitoring is **disabled**, then in the circumstance where DC partition has been detected and the Redis master node is in the unreachable DC, the Redis Monitor will discontinue its normal master role monitoring behavior and will **not promote a reachable slave in a different DC to the master role** (in other words, failover of the master role will **not** be executed). Once the unreachable DC becomes reachable again -- which will be detected by the Redis Monitor -- then the Redis Monitor will resume its normal monitoring behavior and will execute failover of the master role if the existing master node goes down.

By default DC partition monitoring/failover is disabled.

Note If a Redis cluster DC partition occurs an alert will be written to *cloudian-redismon.log* on the Redis Monitor node and an alert will display in the CMC. You can also confirm that the condition exists by using the [test dc partition](#) command. For additional guidance on managing and recovering from a Redis cluster DC partition condition consult with Clodian Support.

14.5.14.1. Command Line Syntax

```
disable dc partition monitoring redis.credentials|redis.qos.<region>
```

Example:

```
# nc 10.50.20.12 9078
disable dc partition monitoring redis.credentials
OK
Skip Monitoring when DC Partition detected
```

14.5.15. enable dc partition monitoring

enable dc partition monitoring

This command affects how the Redis Monitor behaves **after it has detected a data center partition** in a Redis cluster. (For a description of how the Redis Monitor determines that a Redis cluster is in a DC partition condition, see ["test dc partition"](#) (page 954)).

If DC partition monitoring is **enabled**, then in the circumstance where DC partition has been detected and the Redis master node is in the unreachable DC, the Redis Monitor will continue with its normal master role monitoring and managing behavior by **promoting a reachable slave in a different DC to the master role** (in other words, failover of the master role will be executed).

If DC partition monitoring is **disabled**, then in the circumstance where DC partition has been detected and the Redis master node is in the unreachable DC, the Redis Monitor will discontinue its normal master role monitoring behavior and will **not promote a reachable slave in a different DC to the master role** (in other words, failover of the master role will **not** be executed). Once the unreachable DC becomes reachable again -- which will be detected by the Redis Monitor -- then the Redis Monitor will resume its normal monitoring behavior and will execute failover of the master role if the existing master node goes down.

By default DC partition monitoring/failover is disabled.

Note If a Redis cluster DC partition occurs an alert will be written to *cloudian-redismon.log* on the Redis Monitor node and an alert will display in the CMC. You can also confirm that the condition exists by using the [test dc partition](#) command. For additional guidance on managing and recovering from a Redis cluster DC partition condition consult with Clodian Support.

14.5.15.1. Command Line Syntax

```
enable dc partition monitoring redis.credentials|redis.qos.<region>
```

Example:


```
# nc 10.50.20.12 9078
enable dc partition monitoring redis.credentials
OK
Keep Monitoring when DC Partition detected
```

14.5.16. disable split brain monitoring

disable split brain monitoring

This command affects how the Redis Monitor behaves **after it has detected a "split brain" condition** (two simultaneous masters) in a Redis cluster.

If split brain monitoring is **enabled**, then in the circumstance where split brain has been detected the Redis Monitor will continue with its normal master role monitoring and managing behavior by **demoting one of the masters to a slave role**. Of the two masters that constitute the "split brain", Redis Monitor will demote the one that most recently became a master. The node that had been master for a longer period of time will be left as the one master.

If split brain monitoring is **disabled**, then in the circumstance where split brain has been detected the Redis Monitor will discontinue its normal master role monitoring behavior and will **not automatically demote one of the masters to a slave role**. Instead it will be left to you to resolve the split brain condition by using the [resolve split brain](#) command (which will let you choose which node should remain as the one master).

By default split brain monitoring/resolution is enabled.

Note If a Redis cluster "split brain" condition occurs an alert will be written to *cloudian-redismon.log* on the Redis Monitor node and an alert will display in the CMC. For additional guidance on managing and recovering from a Redis cluster split brain condition consult with Clodian Support.

14.5.16.1. Command Line Syntax

```
disable split brain monitoring redis.credentials|redis.qos.<region>
```

Example:

```
# nc 10.50.20.12 9078
disable split brain monitoring redis.credentials
OK
Skip Monitoring when Split Brain detected
```

14.5.17. enable split brain monitoring

enable split brain monitoring

This command affects how the Redis Monitor behaves **after it has detected a "split brain" condition** (two simultaneous masters) in a Redis cluster.

If split brain monitoring is **enabled**, then in the circumstance where split brain has been detected the Redis Monitor will continue with its normal master role monitoring and managing behavior by **demoting one of the masters to a slave role**. Of the two masters that constitute the "split brain", Redis Monitor will demote the one that most recently became a master. The node that had been master for a longer period of time will be left as the one master.

If split brain monitoring is **disabled**, then in the circumstance where split brain has been detected the Redis Monitor will discontinue its normal master role monitoring behavior and will **not automatically demote one of the masters to a slave role**. Instead it will be left to you to resolve the split brain condition by using the [resolve split brain](#) command (which will let you choose which node should remain as the one master).

By default split brain monitoring/resolution is enabled.

Note If a Redis cluster "split brain" condition occurs an alert will be written to *cloudian-redismon.log* on the Redis Monitor node and an alert will display in the CMC. For additional guidance on managing and recovering from a Redis cluster split brain condition consult with Cloudian Support.

14.5.17.1. Command Line Syntax

```
enable split brain monitoring redis.credentials|redis.qos.<region>
```

Example:

```
# nc 10.50.20.12 9078
enable split brain monitoring redis.credentials
OK
Keep Monitoring when Split Brain detected
```

14.5.18. resolve split brain

resolve split brain

This command is applicable **only if you have previously disabled automatic split brain resolution**.

If a Redis cluster is in a "split brain" condition and automatic split brain resolution is disabled, you can use the *resolve split brain* command to resolve the condition. When you run the command, it will show you how long each of the current masters has been acting as a master, and you will be prompted to choose one of the masters to continue as master. The other master will be demoted to slave.

Note If a Redis cluster "split brain" condition occurs an alert will be written to *cloudian-redismon.log* on the Redis Monitor node and an alert will display in the CMC. You can also confirm that the condition exists by using the [test split brain](#) command. For additional guidance on managing and recovering from a Redis cluster split brain condition consult with Cloudian Support.

14.5.18.1. Command Line Syntax

```
resolve split brain redis.credentials|redis.qos.<region>
```

Example:

```
# nc 10.112.2.12 9078
resolve split brain redis.credentials
OK
1. ch-us-east-1-us-east-1a-2-251(10.112.2.251):6379
ch-us-east-1-us-east-1a-2-251(10.112.2.251):6379: Consecutive time being master: 2.51 min
2. ch-us-east-1-us-east-1b-2-33(10.112.2.33):6379
ch-us-east-1-us-east-1b-2-33(10.112.2.33):6379: Consecutive time being master: 8.73 min
```

```
Please select which redis instance as master:
```

```
1
```

```
Forced new Master: ch-us-east-1-us-east-1a-2-251(10.112.2.251):6379
```