

# **Clouddian HyperStore Administration Guide**

**Version 7.2.3**

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 7.2 .....	1
AWS API Support -- New Features and Enhancements .....	1
Admin API -- New Features and Enhancements .....	3
System Operations -- New Features and Enhancements .....	4
Documentation -- New Features and Enhancements .....	12
Chapter 1. Introduction .....	13
1.1. HyperStore Documentation .....	13
1.2. HyperStore Overview .....	14
1.3. Licensing and Auditing .....	15
1.3.1. License Expiration .....	16
1.3.2. Licensed Maximum On-Premise Storage Usage .....	17
1.3.3. Licensed Maximum Tiered Storage Usage .....	18
1.3.4. WORM (Object Lock) License .....	20
1.3.5. HyperIQ License .....	20
1.3.6. License Updating .....	20
1.3.7. Auditing .....	21
1.4. HyperStore Services .....	21
1.4.1. S3 Service .....	22
1.4.2. HyperStore Service and the HSFS .....	23
1.4.3. Cassandra Service .....	26
1.4.4. Redis Credentials and Redis QoS Services .....	26
1.4.5. Redis Monitor Service .....	27
1.4.6. Admin Service .....	28
1.4.7. IAM, STS, and SQS Services .....	28
1.4.8. Cloudian Management Console (CMC) Service .....	28
1.4.9. Supporting Services .....	29
1.5. System Diagrams .....	30
1.5.1. System Levels .....	30
1.5.2. Service Interconnections .....	30
1.5.3. Services Distribution -- 3 Nodes, Single DC .....	31
1.5.4. Services Distribution -- Multi-DC, Single Region .....	32

1.5.5. Services Distribution -- Multi-Region .....	33
1.5.6. Specialized Services Availability .....	34
1.5.7. S3 PUT Processing Flow .....	35
1.5.8. S3 GET Processing Flow .....	37
1.5.9. Data Freshness for Replicated Object Reads .....	38
1.5.10. Dynamic Consistency Levels .....	39
1.5.11. How vNodes Work .....	42
<b>Chapter 2. Getting Started with a New HyperStore System .....</b>	<b>51</b>
<b>Chapter 3. Upgrading Your HyperStore Software Version .....</b>	<b>55</b>
3.1. Preparing to Upgrade Your System .....	55
3.1.1. Additional Upgrade Preparation If Your System Currently Has Failed Disks .....	58
3.1.2. Additional Upgrade Preparation If You Are Using Elasticsearch .....	58
3.2. Upgrading Your System .....	58
3.2.1. Upgrade Failure and Roll-Back .....	60
3.3. Verifying Your System Upgrade .....	61
3.4. Installing a Patch .....	62
3.4.1. Reapplying the Patch in the Case of Installation Errors .....	63
3.4.2. Reverting a Patch .....	63
3.4.3. Adding Nodes to a Patched System .....	64
<b>Chapter 4. Working with HyperStore Major Features .....</b>	<b>66</b>
4.1. Management Interfaces and Tools .....	66
4.1.1. HyperStore Management Interfaces and Tools -- Feature Overview .....	66
4.2. Support for AWS APIs .....	67
4.2.1. Support for AWS APIs -- Feature Overview .....	67
4.3. Nodes, Data Centers, and Regions .....	68
4.3.1. Nodes, Data Centers, and Regions Feature Overview .....	68
4.3.2. Capacity Monitoring and Expansion .....	71
4.3.3. Using the CMC with Multiple DCs or Regions .....	74
4.4. Storage Policies .....	76
4.4.1. Storage Policies Feature Overview .....	76
4.4.2. Consistency Levels .....	79

4.4.3. Object Metadata Replication .....	80
4.4.4. System Metadata Replication .....	81
4.4.5. Creating and Managing Storage Policies .....	83
4.4.6. Assigning a Storage Policy to a Bucket .....	83
4.4.7. Finding an Object's Replicas or EC Fragments .....	84
4.4.8. Storage Policy Resilience to Downed Nodes .....	84
4.5. Security Features .....	89
4.5.1. HyperStore Shell (HSH) .....	89
4.5.2. HyperStore Firewall .....	100
4.5.3. Server-Side Encryption .....	105
4.5.4. FIPS Support .....	113
4.5.5. HTTPS Support (TLS/SSL) .....	114
4.5.6. Secure Delete .....	120
4.5.7. WORM (Object Lock) .....	121
4.6. User Provisioning and LDAP Integration .....	129
4.6.1. User Provisioning and LDAP Integration Feature Overview .....	129
4.6.2. Provisioning Groups .....	130
4.6.3. Provisioning Users .....	130
4.6.4. LDAP Integration .....	131
4.7. Quality of Service Controls .....	135
4.7.1. Quality of Service (QoS) Feature Overview .....	135
4.7.2. Enabling QoS Enforcement .....	136
4.7.3. Setting QoS Limits for Groups .....	137
4.7.4. Setting QoS Limits for Users .....	137
4.8. Usage Reporting and Billing .....	138
4.8.1. Usage Reporting and Billing Feature Overview .....	138
4.8.2. Enabling Advanced Usage Reporting Features .....	142
4.8.3. Validating Storage Usage Data .....	143
4.8.4. Setting Usage Data Retention Periods .....	144
4.8.5. Generating a Usage Report .....	144

4.8.6. Creating Rating Plans for Billing .....	145
4.8.7. Assigning Rating Plans to Users .....	147
4.8.8. Creating a "Whitelist" for Free Traffic .....	148
4.8.9. Generating Billing Data for a User or Group .....	149
4.9. Automated Data Repair .....	150
4.9.1. Automated Data Repair Feature Overview .....	150
4.9.2. Configuring Automatic Data Repair .....	152
4.9.3. Checking Data Repair Status .....	153
4.9.4. Disabling or Stopping Data Repairs .....	154
4.10. Automated Disk Management .....	157
4.10.1. Automated Disk Management Feature Overview .....	157
4.10.2. Configuring Disk Usage Balancing .....	161
4.10.3. Triggering a Disk Usage Balance Check .....	162
4.10.4. Configuring Disk Failure Handling .....	162
4.10.5. Checking Disk Usage and Health Status .....	163
4.10.6. Disk Error Alerts .....	163
4.10.7. Responding to a Disabled Disk .....	163
4.11. Object Metadata .....	164
4.11.1. Object Metadata Feature Overview .....	164
4.11.2. Creating Object Metadata and Tags .....	166
4.11.3. Retrieving Object Metadata and Tags .....	167
4.11.4. Object Metadata Structure in Cassandra .....	167
4.11.5. Elasticsearch Integration for Object Metadata .....	171
4.12. Auto-Tiering .....	176
4.12.1. Auto-Tiering Feature Overview .....	176
4.12.2. Setting Up Auto-Tiering .....	180
4.12.3. Accessing Auto-Tiered Objects .....	184
4.13. Cross-Region Replication .....	186
4.13.1. Cross-Region Replication Feature Overview .....	186
4.13.2. Configuring Cross-Region Replication for a Bucket .....	189

4.14. Smart Support .....	190
4.14.1. Smart Support and Diagnostics Feature Overview .....	190
4.14.2. Configuring Smart Support and Node Diagnostics .....	192
4.14.3. Executing Node Diagnostics Collection .....	193
<b>Chapter 5. Cloudfan Management Console (CMC) .....</b>	<b>196</b>
5.1. Dashboard .....	197
5.1.1. Dashboard .....	197
5.2. Analytics .....	206
5.2.1. Cluster Usage .....	206
5.2.2. Capacity Explorer .....	209
5.2.3. Usage By Users & Groups .....	211
5.2.4. Object Locator .....	217
5.3. Buckets .....	218
5.3.1. Add a Bucket .....	218
5.3.2. Set Bucket Properties .....	221
5.3.3. Delete a Bucket .....	244
5.4. Objects .....	244
5.4.1. Create or Delete a "Folder" .....	245
5.4.2. Upload an Object .....	246
5.4.3. Set Object Properties .....	248
5.4.4. List or Search for Objects .....	257
5.4.5. Download an Object .....	258
5.4.6. Restore an Auto-Tiered Object .....	258
5.4.7. Delete an Object .....	261
5.5. Users & Groups .....	262
5.5.1. Manage Users .....	262
5.5.2. Manage Groups .....	270
5.5.3. Rating Plan .....	278
5.5.4. Account Activity .....	282
5.5.5. Whitelist .....	283

5.5.6. Set Quality of Service (QoS) Controls .....	285
5.6. IAM .....	289
5.6.1. Manage IAM User .....	289
5.6.2. Manage IAM Group .....	297
5.6.3. Manage IAM Policy .....	303
5.7. Cluster .....	309
5.7.1. Data Centers .....	309
5.7.2. Node Status .....	313
5.7.3. Node Activity .....	323
5.7.4. Node Advanced .....	326
5.7.5. Cluster Information .....	330
5.7.6. Configuration Settings .....	337
5.7.7. Storage Policies .....	353
5.7.8. Repair Status .....	380
5.7.9. Operation Status .....	384
5.8. Alerts .....	385
5.8.1. Alerts .....	385
5.8.2. Alert Rules .....	390
5.8.3. How HyperStore Implements Alerts .....	398
5.9. My Account .....	399
5.9.1. Profile .....	399
5.9.2. Security Credentials .....	400
5.10. Customizing the CMC .....	402
5.10.1. Showing/Hiding CMC UI Functions .....	402
5.10.2. Rebranding the CMC UI .....	404
5.10.3. Configuring a Login Page Banner .....	407
5.10.4. Configuring a Login Page Acknowledgment Gate .....	408
5.10.5. Implementing Single Sign-On for the CMC .....	410
<b>Chapter 6. Node and Cluster Operations .....</b>	<b>415</b>
6.1. Starting and Stopping Services .....	415

6.1.1. Start or Stop Services on All Nodes in the Cluster .....	415
6.1.2. Start or Stop Services on One Node .....	417
6.1.3. Shutting Down or Rebooting a Node .....	419
6.1.4. Automatic Service Start on Node Boot-Up .....	419
6.2. Adding Nodes .....	420
6.2.1. Special Requirements if an Existing Node is Down .....	420
6.2.2. Preparing to Add Nodes .....	420
6.2.3. Adding Nodes .....	423
6.3. Adding a Data Center .....	430
6.3.1. Special Requirements if an Existing Node is Down or Unreachable .....	430
6.3.2. Preparing to Add a Data Center .....	430
6.3.3. Adding a Data Center .....	432
6.4. Adding a Region .....	437
6.4.1. Preparing to Add a Region .....	437
6.4.2. Adding a Region .....	439
6.5. Removing a Node .....	443
6.5.1. Preparing to Remove a Node .....	443
6.5.2. Removing a Node .....	448
6.6. Replacing a Node .....	452
6.7. Restoring a Node That Has Been Offline .....	453
6.7.1. 6.7.1 Repairing a Node That's Been Down for Longer than the Proactive Repair Limit .....	454
6.8. Changing a Node's IP Address .....	455
6.9. Backing Up and Restoring a Cluster .....	455
6.9.1. Backing Up an Entire Cluster .....	455
6.9.2. Restoring an Entire Cluster .....	456
6.10. Change Node Role Assignments .....	457
6.10.1. Move the Redis Credentials Master or QoS Master Role .....	458
6.10.2. Move or Add a Redis Credentials Slave or Redis QoS Slave .....	461
6.10.3. Move the Cassandra Seed Role .....	462
6.10.4. Reduce or Change the List of CMC Hosts .....	464
6.10.5. Move the Redis Monitor Primary or Backup Role .....	465

6.10.6. Move the Cron Job Primary or Backup Role .....	466
6.10.7. Move the Puppet Master Primary or Backup Role .....	468
6.10.8. Change Internal NTP Servers or External NTP Servers .....	471
6.11. Cron Jobs and Automated System Maintenance .....	472
6.11.1. System cron Jobs .....	473
6.11.2. Scheduled Auto-Repair .....	478
6.11.3. Cassandra Data Compaction .....	478
<b>Chapter 7. Disk Operations .....</b>	<b>479</b>
7.1. Disabling a HyperStore Data Disk .....	479
7.1.1. The Impact of Disabling a Disk .....	479
7.1.2. Disabling a Disk .....	479
7.2. Enabling a HyperStore Data Disk .....	480
7.2.1. The Impact of Enabling a Disk .....	481
7.2.2. Enabling a Disabled Disk .....	481
7.3. Replacing a HyperStore Data Disk .....	482
7.3.1. The Impact of Replacing a Disk .....	482
7.3.2. Replacing a Disk .....	483
7.4. Replacing a Cassandra Disk .....	484
7.5. Responding to Data Disks Nearing Capacity .....	487
7.6. Responding to Cassandra Disks Nearing Capacity .....	488
7.7. Adding Disks is Not Supported .....	489
<b>Chapter 8. System Monitoring .....</b>	<b>491</b>
8.1. Using the CMC to Monitor Your HyperStore System .....	491
8.2. Cloudbian HyperIQ .....	491
8.3. Additional Monitoring Tools .....	492
8.3.1. Using the Admin API to Monitor HyperStore .....	492
8.3.2. Doing an HTTP Health Check .....	492
8.3.3. Using JMX to Monitor Java-Based HyperStore Services .....	493
8.3.4. Using Native Linux Utilities for System Resource Monitoring .....	498
8.3.5. Using nodetool to Monitor Cassandra .....	498
8.3.6. Using the Redis CLI to Monitor Redis .....	499
<b>Chapter 9. System Configuration .....</b>	<b>501</b>

9.1. CMC's Configuration Settings Page .....	501
9.2. Installer Advanced Configuration Options .....	501
9.3. Pushing Configuration File Edits to the Cluster and Restarting Services .....	506
9.3.1. Puppet Overview .....	506
9.3.2. Installation Staging Directory .....	507
9.3.3. Using the Installer to Push Configuration Changes and Restart Services .....	507
9.3.4. Option for Triggering a Puppet Sync-Up from the Command Line .....	509
9.3.5. Excluding a Down Node from an Installer-Driven Configuration Push .....	509
9.3.6. Automatic Puppet Sync-Up on an Interval .....	510
9.4. Using the HSH to Manage Configuration Files .....	510
9.5. HyperStore Configuration Files .....	511
9.5.1. common.csv .....	512
9.5.2. hyperstore-server.properties.erb .....	544
9.5.3. mts.properties.erb .....	553
9.5.4. mts-ui.properties.erb .....	580
9.5.5. Other Configuration Files .....	594
9.5.6. Using JMX to Dynamically Change Configuration Settings .....	597
9.6. Configuration Special Topics .....	598
9.6.1. Anti-Virus Software .....	598
9.6.2. NTP Automatic Set-Up .....	598
9.6.3. Changing S3, Admin, or CMC Listening Ports .....	599
9.6.4. Changing S3, Admin, CMC, or IAM Service Endpoints .....	600
9.6.5. Tuning HyperStore Performance Parameters .....	602
9.6.6. Vanity Domains for S3 Buckets .....	603
<b>Chapter 10. Logging .....</b>	<b>605</b>
10.1. HyperStore Logs .....	605
10.1.1. Admin Service Logs .....	605
10.1.2. Cassandra Logs .....	606
10.1.3. CMC Log .....	608
10.1.4. HyperStore Firewall Log .....	609
10.1.5. HyperStore Service Logs .....	609

10.1.6. HyperStore Shell Log .....	613
10.1.7. IAM Service Logs .....	613
10.1.8. Monitoring Agent and Collector Logs .....	615
10.1.9. Phone Home (Smart Support) Log .....	616
10.1.10. Redis and Redis Monitor Logs .....	617
10.1.11. S3 Service Logs (including Auto-Tiering, CRR, and WORM) .....	619
10.1.12. SQS Service Logs .....	625
10.2. Log Configuration Settings .....	626
10.3. Aggregating Logs to a Central Server .....	627
10.4. Setting Up Elastic Stack for S3 Request Traffic Analysis .....	631
10.4.1. Installing Elasticsearch, Kibana, and Logstash .....	631
10.4.2. Installing Filebeat .....	634
10.4.3. Configuring Kibana for Custom Metrics Visualizations .....	635
10.5. Using the HSH to View Logs .....	640
<b>Chapter 11. Commands .....</b>	<b>643</b>
11.1. hsstool .....	643
11.1.1. hsstool cleanup .....	644
11.1.2. hsstool cleanupec .....	651
11.1.3. hsstool info .....	658
11.1.4. hsstool ls .....	660
11.1.5. hsstool metadata .....	662
11.1.6. hsstool opctl .....	665
11.1.7. hsstool opstatus .....	666
11.1.8. hsstool proactiverepairq .....	674
11.1.9. hsstool rebalance .....	679
11.1.10. hsstool repair .....	686
11.1.11. hsstool repaircassandra .....	695
11.1.12. hsstool repairec .....	697
11.1.13. hsstool repairqueue .....	707
11.1.14. hsstool ring .....	712
11.1.15. hsstool status .....	714

11.1.16. hsstool trmap .....	717
11.1.17. hsstool whereis .....	721
<b>11.2. Redis Monitor Commands .....</b>	<b>726</b>
11.2.1. get cluster .....	727
11.2.2. get master .....	728
11.2.3. get nodes .....	729
11.2.4. get clients .....	729
11.2.5. enable monitoring .....	730
11.2.6. disable monitoring .....	730
11.2.7. enable notifications .....	730
11.2.8. disable notifications .....	731
11.2.9. set master .....	731
11.2.10. add node .....	732
11.2.11. add client .....	733
11.2.12. test dc partition .....	733
11.2.13. test split brain .....	734
11.2.14. disable dc partition monitoring .....	735
11.2.15. enable dc partition monitoring .....	736
11.2.16. disable split brain monitoring .....	737
11.2.17. enable split brain monitoring .....	738
11.2.18. resolve split brain .....	739
<b>Chapter 12. Admin API .....</b>	<b>741</b>
12.1. Introduction .....	741
12.1.1. HyperStore Admin API Introduction .....	741
12.1.2. Admin API Methods List .....	742
12.1.3. Common Request and Response Headers .....	745
12.1.4. Common Response Status Codes .....	746
12.1.5. cURL Examples .....	746
12.1.6. HTTP and HTTPS for Admin API Access .....	747
12.1.7. HTTP/S Basic Authentication for Admin API Access .....	748

12.1.8. Admin API Logging .....	750
12.2. billing .....	751
12.2.1. GET /billing .....	751
12.2.2. POST /billing .....	753
12.2.3. billing Query Parameters .....	755
12.2.4. billing Objects .....	755
12.3. bppolicy .....	759
12.3.1. GET /bppolicy/bucketsperpolicy .....	759
12.3.2. GET /bppolicy/listpolicy .....	760
12.3.3. bppolicy Query Parameters .....	761
12.3.4. bppolicy Objects .....	761
12.4. bucketops .....	762
12.4.1. GET /bucketops/id .....	762
12.4.2. GET /bucketops/gettags .....	763
12.4.3. POST /bucketops/purge .....	764
12.4.4. bucketops Query Parameters .....	765
12.4.5. bucketops Objects .....	766
12.5. group .....	767
12.5.1. DELETE /group .....	767
12.5.2. GET /group .....	768
12.5.3. GET /group/list .....	770
12.5.4. GET /group/ratingPlanId .....	772
12.5.5. POST /group .....	773
12.5.6. POST /group/ratingPlanId .....	774
12.5.7. PUT /group .....	774
12.5.8. group Query Parameters .....	776
12.5.9. group Objects .....	776
12.6. monitor .....	781
12.6.1. DELETE /monitor/notificationrule .....	781
12.6.2. GET /monitor/events .....	781

12.6.3. GET /monitor/nodelist .....	784
12.6.4. GET /monitor/host .....	785
12.6.5. GET /monitor .....	790
12.6.6. GET /monitor/history .....	793
12.6.7. GET /monitor/notificationrules .....	794
12.6.8. POST /monitor/acknowledgeevents .....	796
12.6.9. POST /monitor/notificationruleenable .....	796
12.6.10. POST /monitor/notificationrule .....	797
12.6.11. PUT /monitor/notificationrule .....	798
12.6.12. monitor Query Parameters .....	799
12.6.13. monitor Objects .....	801
12.7. permissions .....	819
12.7.1. GET /permissions/publicUrl .....	819
12.7.2. POST /permissions/publicUrl .....	820
12.7.3. permissions Query Parameters .....	821
12.7.4. permissions Objects .....	822
12.8. qos .....	823
12.8.1. DELETE /qos/limits .....	823
12.8.2. GET /qos/limits .....	824
12.8.3. POST /qos/limits .....	826
12.8.4. qos Query Parameters .....	827
12.8.5. qos Objects .....	829
12.9. ratingPlan .....	832
12.9.1. DELETE /ratingPlan .....	833
12.9.2. GET /ratingPlan .....	833
12.9.3. GET /ratingPlan/list .....	835
12.9.4. POST /ratingPlan .....	836
12.9.5. PUT /ratingPlan .....	837
12.9.6. ratingPlan Query Parameters .....	839
12.9.7. ratingPlan Objects .....	839

12.10. system .....	841
12.10.1. GET /system/audit .....	841
12.10.2. GET /system/bucketcount .....	842
12.10.3. GET /system/bucketlist .....	843
12.10.4. GET /system/bytecount .....	844
12.10.5. GET /system/bytestiered .....	845
12.10.6. GET /system/groupbytecount .....	846
12.10.7. GET /system/groupobjectcount .....	847
12.10.8. GET /system/license .....	848
12.10.9. GET system/objectcount .....	850
12.10.10. GET /system/objectlockenabled .....	851
12.10.11. GET /system/version .....	852
12.10.12. POST /system/processProtectionPolicy .....	853
12.10.13. POST /system/repairusercount .....	854
12.10.14. system Query Parameters .....	854
12.10.15. system Objects .....	855
12.11. tiering .....	861
12.11.1. DELETE /tiering/credentials .....	861
12.11.2. DELETE /tiering/azure/credentials .....	861
12.11.3. DELETE /tiering/spectra/credentials .....	862
12.11.4. GET /tiering/credentials .....	862
12.11.5. GET /tiering/credentials/src .....	863
12.11.6. GET /tiering/azure/credentials .....	864
12.11.7. GET /tiering/spectra/credentials .....	864
12.11.8. POST /tiering/credentials .....	865
12.11.9. POST /tiering/azure/credentials .....	866
12.11.10. POST /tiering/spectra/credentials .....	867
12.11.11. tiering Query Parameters .....	868
12.12. usage .....	868
12.12.1. DELETE /usage .....	868

12.12.2. GET /usage .....	869
12.12.3. POST /usage/bucket .....	873
12.12.4. POST /usage/repair .....	875
12.12.5. POST /usage/repair/bucket .....	875
12.12.6. POST /usage/repair/dirtyusers .....	876
12.12.7. POST /usage/repair/user .....	877
12.12.8. POST /usage/rollup .....	878
12.12.9. POST /usage/storage .....	879
12.12.10. POST /usage/storageall .....	880
12.12.11. usage Query Parameters .....	880
12.12.12. usage Objects .....	885
12.13. user .....	895
12.13.1. DELETE /user .....	895
12.13.2. DELETE /user/credentials .....	896
12.13.3. DELETE /user/deleted .....	897
12.13.4. GET /user .....	898
12.13.5. GET /user/credentials .....	900
12.13.6. GET /user/credentials/list .....	902
12.13.7. GET /user/credentials/list/active .....	904
12.13.8. GET /user/list .....	906
12.13.9. GET /user/password/verify .....	909
12.13.10. GET /user/ratingPlan .....	910
12.13.11. GET /user/ratingPlanId .....	912
12.13.12. POST /user .....	912
12.13.13. POST /user/credentials .....	913
12.13.14. POST /user/credentials/status .....	914
12.13.15. POST /user/password .....	914
12.13.16. POST /user/ratingPlanId .....	915
12.13.17. PUT /user .....	916
12.13.18. PUT /user/credentials .....	918

12.13.19. user Query Parameters .....	919
12.13.20. user Objects .....	921
12.14. whitelist .....	925
12.14.1. GET /whitelist .....	925
12.14.2. POST /whitelist .....	926
12.14.3. POST /whitelist/list .....	927
12.14.4. whitelist Query Parameters .....	928
12.14.5. whitelist Objects .....	928
<b>Chapter 13. S3 API .....</b>	<b>931</b>
13.1. Introduction .....	931
13.1.1. HyperStore Support for the AWS S3 API .....	931
13.1.2. S3 Client Application Options .....	932
13.1.3. Authenticating Requests (AWS Signature Version 4) .....	933
13.1.4. Access Control List (ACL) Support .....	934
13.1.5. S3 Common Request and Response Headers .....	935
13.1.6. S3 Error Responses .....	936
13.1.7. HyperStore Extensions to the S3 API .....	937
13.2. Supported S3 Operations .....	938
13.2.1. AbortMultipartUpload .....	938
13.2.2. CompleteMultipartUpload .....	938
13.2.3. CopyObject .....	939
13.2.4. CreateBucket .....	941
13.2.5. CreateMultipartUpload .....	942
13.2.6. DeleteBucket .....	944
13.2.7. DeleteBucketCors .....	944
13.2.8. DeleteBucketEncryption .....	944
13.2.9. DeleteBucketLifecycle .....	944
13.2.10. DeleteBucketPolicy .....	945
13.2.11. DeleteBucketReplication .....	945
13.2.12. DeleteBucketTagging .....	945

13.2.13. DeleteBucketWebsite .....	945
13.2.14. DeleteObject .....	945
13.2.15. DeleteObjects .....	946
13.2.16. DeleteObjectTagging .....	947
13.2.17. GetBucketAcl .....	948
13.2.18. GetBucketCors .....	948
13.2.19. GetBucketEncryption .....	948
13.2.20. GetBucketLifecycle .....	949
13.2.21. GetBucketLocation .....	949
13.2.22. GetBucketLogging .....	950
13.2.23. GetBucketNotificationConfiguration .....	950
13.2.24. GetBucketPolicy .....	951
13.2.25. GetBucketReplication .....	951
13.2.26. GetBucketTagging .....	951
13.2.27. GetBucketVersioning .....	952
13.2.28. GetBucketWebsite .....	952
13.2.29. GetObject .....	952
13.2.30. GetObjectAcl .....	954
13.2.31. GetObjectLegalHold .....	954
13.2.32. GetObjectLockConfiguration .....	955
13.2.33. GetObjectRetention .....	955
13.2.34. GetObjectTagging .....	955
13.2.35. GetObjectTorrent .....	956
13.2.36. HeadBucket .....	956
13.2.37. HeadObject .....	957
13.2.38. ListBuckets .....	958
13.2.39. ListMultipartUploads .....	959
13.2.40. ListObjects .....	960
13.2.41. ListObjectsV2 .....	962
13.2.42. ListObjectVersions .....	963

13.2.43. ListParts .....	964
13.2.44. OPTIONS Object .....	965
13.2.45. POST Object .....	966
13.2.46. PutBucketAcl .....	967
13.2.47. PutBucketCors .....	968
13.2.48. PutBucketEncryption .....	969
13.2.49. PutBucketLifecycle .....	969
13.2.50. PutBucketLogging .....	975
13.2.51. PutBucketNotificationConfiguration .....	976
13.2.52. PutBucketPolicy .....	977
13.2.53. PutBucketReplication .....	981
13.2.54. PutBucketTagging .....	983
13.2.55. PutBucketVersioning .....	983
13.2.56. PutBucketWebsite .....	983
13.2.57. PutObject .....	984
13.2.58. PutObjectAcl .....	986
13.2.59. PutObjectLegalHold .....	986
13.2.60. PutObjectLockConfiguration .....	987
13.2.61. PutObjectRetention .....	987
13.2.62. PutObjectTagging .....	988
13.2.63. RestoreObject .....	988
13.2.64. UploadPart .....	989
13.2.65. UploadPartCopy .....	990
<b>Chapter 14. IAM API .....</b>	<b>991</b>
14.1. Introduction .....	991
14.1.1. HyperStore Support for the AWS IAM API .....	991
14.1.2. IAM Client Application Options .....	992
14.1.3. IAM Common Request Parameters .....	994
14.1.4. IAM Common Errors .....	994
14.2. Supported IAM Actions .....	995

14.2.1. AddUserToGroup .....	995
14.2.2. AttachGroupPolicy .....	995
14.2.3. AttachRolePolicy .....	996
14.2.4. AttachUserPolicy .....	996
14.2.5. CreateAccessKey .....	997
14.2.6. CreateGroup .....	997
14.2.7. CreatePolicy .....	998
14.2.8. CreateRole .....	999
14.2.9. CreateSAMLProvider .....	1000
14.2.10. CreateUser .....	1000
14.2.11. DeleteAccessKey .....	1001
14.2.12. DeleteGroup .....	1002
14.2.13. DeleteGroupPolicy .....	1002
14.2.14. DeletePolicy .....	1002
14.2.15. DeleteRole .....	1003
14.2.16. DeleteRolePolicy .....	1003
14.2.17. DeleteSAMLProvider .....	1004
14.2.18. DeleteUser .....	1004
14.2.19. DeleteUserPolicy .....	1004
14.2.20. DetachGroupPolicy .....	1005
14.2.21. DetachRolePolicy .....	1005
14.2.22. DetachUserPolicy .....	1006
14.2.23. GetGroup .....	1006
14.2.24. GetGroupPolicy .....	1007
14.2.25. GetPolicy .....	1008
14.2.26. GetPolicyVersion .....	1008
14.2.27. GetRole .....	1009
14.2.28. GetRolePolicy .....	1009
14.2.29. GetSAMLProvider .....	1010
14.2.30. GetUser .....	1010

14.2.31. GetUserPolicy .....	1011
14.2.32. ListAccessKeys .....	1011
14.2.33. ListAttachedGroupPolicies .....	1012
14.2.34. ListAttachedRolePolicies .....	1012
14.2.35. ListAttachedUserPolicies .....	1013
14.2.36. ListEntitiesForPolicy .....	1013
14.2.37. ListGroupPolicies .....	1014
14.2.38. ListGroups .....	1014
14.2.39. ListGroupsForUser .....	1015
14.2.40. ListPolicies .....	1015
14.2.41. ListPolicyVersions .....	1016
14.2.42. ListRolePolicies .....	1016
14.2.43. ListRoles .....	1017
14.2.44. ListSAMLProviders .....	1017
14.2.45. ListUserPolicies .....	1017
14.2.46. ListUsers .....	1018
14.2.47. PutGroupPolicy .....	1018
14.2.48. PutRolePolicy .....	1019
14.2.49. PutUserPolicy .....	1019
14.2.50. RemoveUserFromGroup .....	1020
14.2.51. UpdateAccessKey .....	1020
14.2.52. UpdateAssumeRolePolicy .....	1021
14.2.53. UpdateGroup .....	1021
14.2.54. UpdateRole .....	1022
14.2.55. UpdateRoleDescription .....	1022
14.2.56. UpdateSAMLProvider .....	1022
14.2.57. UpdateUser .....	1023
14.3. Supported IAM Policy Elements .....	1023
14.3.1. Policy Document Content for Granting S3 or IAM Permissions .....	1024
14.3.2. Policy Document Content for Granting HyperStore Administrative Permissions .....	1025

14.4. IAM Extensions for Role-Based Access to HyperStore Admin Functions .....	1027
14.4.1. Comparing the Admin API to the IAM API with RBAC Extensions .....	1027
14.4.2. Administrative Actions Supported by the IAM API .....	1028
14.4.3. Giving Administrative Action Privileges to IAM Users .....	1030
14.4.4. Using admin_client.py to Call the IAM Service Extensions for Administrative Actions .....	1031
14.5. SAML Support .....	1032
14.5.1. Downloading the HyperStore SAML Metadata Document for IdP Setup .....	1033
14.5.2. Using the IAM Service to Create and Manage SAML Provider Resources .....	1033
14.5.3. Using the IAM Service to Create and Manage Roles .....	1034
14.5.4. Using the STS Service to Assume a Role .....	1034
<b>Chapter 15. STS API .....</b>	<b>1037</b>
15.1. Introduction .....	1037
15.1.1. HyperStore Support for the AWS STS API .....	1037
15.1.2. STS Common Request Parameters .....	1037
15.1.3. STS Common Errors .....	1037
15.2. Supported STS Actions .....	1038
15.2.1. AssumeRole .....	1038
15.2.2. AssumeRoleWithSAML .....	1039
15.2.3. GetCallerIdentity .....	1039
<b>Chapter 16. SQS API .....</b>	<b>1041</b>
16.1. HyperStore Support for the AWS SQS API .....	1041
16.1.1. Enabling the Bucket Notification Feature and the SQS Service .....	1041
16.2. SQS Supported Actions .....	1042
<b>Chapter 17. Open Source License Agreements .....</b>	<b>1045</b>

This page left intentionally blank.

# What's New in HyperStore 7.2

This section introduces the main new features and enhancements for Cloudian HyperStore version 7.2. [Click an item for a summary of the change and links to further information.](#)

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

## AWS API Support -- New Features and Enhancements

### *S3 API support for Object Lock (WORM)*

HyperStore's S3 Service now supports the S3 API calls that implement Object Lock. Object Lock provides WORM (write once, read many) protection that prevents objects from being deleted or altered before the completion of a specified time period.

More information, including the requirements that your HyperStore deployment must meet to use Object Lock:

- **"WORM (Object Lock)"** (page 121)
- **"Setting Up Object Lock"** (page 123)

**Note** Support for the standard AWS S3 Object Lock APIs **replaces** the Cloudian-proprietary "Bucket Lock" feature that had been introduced in HyperStore 7.1. That proprietary "Bucket Lock" feature has now been removed from HyperStore in favor of supporting the standard AWS S3 Object Lock calls.

### *Support for Bucket Notifications and Simple Queue Service (SQS)*

HyperStore's S3 Service now supports the S3 API calls that implement bucket notifications. In connection with supporting these S3 API calls, HyperStore now implements its own Simple Queue Service (SQS) that is compatible with the AWS Simple Queue Service API.

The CMC does not yet support this functionality. To use this HyperStore functionality requires that you have an S3 client application that supports the standard AWS S3 API call that configures bucket notification; and an SQS client application that supports the standard AWS SQS API calls for operations such as creating queues and receiving notification messages.

HyperStore's bucket notification feature and its SQS Service are **disabled by default**. For instructions for enabling these features see **"HyperStore Support for the AWS SQS API"** (page 1041).

More information:

- **"PutBucketNotificationConfiguration"** (page 976)
- **"GetBucketNotificationConfiguration"** (page 950)
- **"HyperStore Support for the AWS SQS API"** (page 1041)
- **"SQS Supported Actions"** (page 1042)

### *IAM Service now enabled by default*

HyperStore's IAM Service is now enabled by default. Previously this service was disabled by default, and you

had to modify the system configuration if you wanted to activate the service.

More information:

- **"HyperStore Support for the AWS IAM API"** (page 991)

*IAM Service has its own dedicated service endpoint*

HyperStore's IAM Service now has its own configurable service endpoint. Previously the IAM Service was accessed through the Admin Service endpoint.

More information:

- **"Changing S3, Admin, CMC, or IAM Service Endpoints"** (page 600)
- DNS Set-Up
- Load Balancing

*IAM Service supports 'Role' actions*

HyperStore's IAM Service now supports all the Actions related to Roles.

The CMC does not yet support this functionality. To use this HyperStore functionality requires that you have an IAM client application that supports the standard AWS IAM API actions relating to Roles.

More information:

- **"Supported IAM Actions"** (page 995)

*CMC's IAM client has improved support for IAM policy creation and assignment*

The CMC's IAM client now supports creating IAM "managed" policies as well as creating IAM "inline" policies for groups and for users. Previously the CMC's IAM client only supported creating inline policies, and only for groups.

Additionally, the CMC now includes a GUI-driven method (a "visual editor") for creating policies, as well as a JSON editor. Previously it only had the JSON editor.

More information:

- **"Manage IAM Policy"** (page 303)
- **"Manage IAM Group"** (page 297)
- **"Manage IAM User"** (page 289)

*CMC support for Object Lock (7.2.2)*

Starting in HyperStore version 7.2.2, the CMC's S3 client supports Object Lock. In HyperStore 7.2.0 and 7.2.1, the S3 Service (S3 API server) supported Object Lock but the CMC's S3 client application did not.

More information:

- **"Add a Bucket"** (page 218)
- **"Configure Object Lock Properties for a Bucket"** (page 242)
- **"Set Object Lock Attributes on an Object"** (page 255)

*S3 Service support for partNumber query parameter (7.2.3)*

HyperStore's S3 Service now supports the *partNumber* query parameter for the *GetObject* and *HeadObject* operations, in regard to objects that were uploaded using the multipart upload method.

More information:

- **"GetObject"** (page 952)
- **"HeadObject"** (page 957)

#### *S3 Service no longer prohibits HTTPS access to static websites (7.2.3)*

HyperStore's S3 Service now allows static website requests that use HTTPS (TLS). HyperStore does not provide a mechanism for setting up TLS for static website requests -- you would need to do so outside of HyperStore, by appropriately configuring your TLS certificates. But the HyperStore S3 Service no longer prohibits HTTPS access to static websites.

More information:

- **"GetObject"** (page 952)
- **"HeadObject"** (page 957)

#### *Support for IAM and STS calls to enable SAML-based access to S3 services (7.2.3)*

HyperStore now supports SAML-based access to HyperStore S3 storage services. This includes support for all the IAM "Role" calls and the Security Token Service (STS) calls that are required in order to implement SAML-based access.

More information:

- **"SAML Support"** (page 1032)
- **"HyperStore Support for the AWS STS API"** (page 1037)

## Admin API -- New Features and Enhancements

#### *Group attributes for filtering S3 endpoint displays*

The *GroupInfo* object now includes attributes that give you the option to restrict which S3 endpoints display for the group's users when they log into the CMC and go to the **Security Credentials** page. By default in that CMC page users can view all the S3 endpoints that are configured in the system.

You can specify the S3 endpoint display filtering for a group when you create or edit the group through the Admin API or through the CMC.

More information:

- **"group"** (page 767)
- **"GroupInfo Object"** (page 777)
- **"Add a Group"** (page 271) (CMC)
- **"Edit a Group"** (page 276) (CMC)

#### *New API calls for retrieving current storage usage for each user in a specified group*

The Admin API now allows you, in a single call, to retrieve current stored byte counts for each user in a specified group; or, in a single call, to retrieve current stored object counts for each user in a specified group.

More information:

- **"GET /system/groupbytecount Get stored byte counts for all of a group's users"** (page 846)
- **"GET /system/groupobjectcount Get stored object counts for all of a group's users"** (page 847)

#### *Randomly generated Admin API HTTP(S) Basic Authentication password (7.2.2)*

In new installations of HyperStore version 7.2.2 or newer, the system now generates a random password as

the default password to be used for Admin API HTTP(S) Basic Authentication. In older versions of HyperStore the default password for this purpose was "public".

This feature change applies only to new installations. Upgrading from an older version to 7.2.2 or newer does not change your system's Admin API HTTP(S) Basic Authentication password.

More information:

- **"HTTP/S Basic Authentication for Admin API Access"** (page 748)

*API call to retrieve enabled/disabled status of the Object Lock feature (7.2.3)*

The Admin API now supports a call for checking whether or not the Object Lock feature is enabled in your HyperStore system.

More information:

- **"GET /system/objectlockenabled Get Object Lock feature status"** (page 851)

## System Operations -- New Features and Enhancements

*HyperStore shell (HSH)*

HyperStore nodes now include a restrictive user login and command shell that enables administrators to log into HyperStore nodes and execute common HyperStore and OS commands without requiring root access or using the more open-ended Bash shell. The HyperStore shell is disabled by default.

More information:

- **"Security Features"** (page 89)

*HyperStore firewall*

HyperStore nodes now include a built-in firewall that protects HyperStore internal services while allowing access to HyperStore public services. For fresh installations of HyperStore 7.2, the HyperStore firewall is enabled by default. For HyperStore systems originally installed as an older version and then upgraded to 7.2, the HyperStore is available but is disabled by default. You can enable or disable the firewall using the installer's Advanced Configuration Options.

More information:

- **"HyperStore Firewall"** (page 100)

*FIPS support*

HyperStore's core cryptographic module now complies with the requirements of Federal Information Processing Standard (FIPS) Publication 140-2. Optionally, you can configure the SSH server on HyperStore nodes to also comply with those requirements.

More information:

- **"FIPS Support"** (page 113)

*Improved support for managing HTTPS and TLS/SSL certificate keystores*

The HyperStore installer's functions for managing HTTPS and the associated TLS/SSL certificate keystores have been simplified and better automated. This allows you to more easily manage HTTPS and keystores for the S3 Service, the IAM Service, and the CMC.

More information:

- **"HTTPS Support (TLS/SSL)"** (page 114)

*All HyperStore HTTPS listeners now require clients to use TLS 1.2*

HTTPS listeners for the Admin Service, CMC, IAM Service, and S3 Service will not accept client connections that use TLS versions older than 1.2. Previously only the S3 Service's HTTPS listener required clients to use TLS 1.2.

More information:

- **"HTTPS Support (TLS/SSL)"** (page 114)

*Faster and easier freeing of existing storage space when adding new nodes*

The `hsstool rebalance` command now includes an option to perform cleanup of existing nodes as part of the same operation that rebalances data to the newly added node(s). Previously you had to run separate cleanup operations on each of the older nodes, after the rebalance operation completed for the newly added node(s). The new method is simpler, and more quickly frees up storage space on your older nodes.

More information:

- **"hsstool rebalance"** (page 679)
- **"Adding Nodes"** (page 420)

*Automatic creation of installation staging directory*

Previously you needed to create an installation directory when you first install HyperStore and then create a different staging directory each time that you upgrade HyperStore to a new version. Now, when you extract the HyperStore package (`.bin` file) for a HyperStore version that you are installing or upgrading to, a staging directory named `/opt/cloudian-staging/<version-number>` is automatically created and the package contents are extracted into that directory.

More information:

- **"Upgrading Your HyperStore Software Version"** (page 55)

*Run hsstool from any directory*

In previous HyperStore versions `hsstool` had to be run from the `/opt/cloudian/bin` directory. Now HyperStore automatically adds `/opt/cloudian/bin` to each host's `$PATH` variable, so you can run `hsstool` from any directory.

More information:

- **"hsstool"** (page 643)

*Improved monitoring and management for multi-DC Redis clusters*

The Redis Monitor has been enhanced to provide monitoring and management for abnormal conditions that can occur in Redis clusters if your HyperStore system spans multiple data centers -- specifically, monitoring and management for "data center partition" conditions and "split brain" conditions.

More information:

- **"Redis Monitor Commands"** (page 726) (see the commands in sections 11.2.12 through 11.2.18, pertaining to DC partition and split brain)

*CMC's Operation Status page now includes region and DC info*

In the CMC's **Operation Status** page, the information for each operation now includes the region and data center in which the target node resides. Previously this page only displayed the hostname of the target node

without identifying the region and data center.

More information:

- **"Operation Status"** (page 384)

#### *Improved disk status display in CMC*

In the CMC's **Node Status** page, the scheme of color-coded icons for indicating disk status has been made simpler and more clear.

More information:

- **"View a Node's Disk Detail"** (page 317)

#### *Improved logging of CMC user logins*

All user logins to the CMC are now recorded in the CMC application log *cloudian-ui.log*. The log entries include the user ID, group ID, and source IP address.

More information:

- **"HyperStore Logs"** (page 605)

#### *Puppet Master and Cronjob primary and backup roles now allocated for high availability in multi-DC install*

For a new multi-DC installation of HyperStore, the Puppet Master backup role is now allocated to a node in a different DC as the node hosting the Puppet Master primary role; and the Cronjobs backup role is allocated to a node in a different DC as the node hosting the Cronjob primary role.

More information:

- **"Services Distribution -- Multi-DC, Single Region"** (page 32)

#### *hsstool repairec improvements*

The *hsstool repairec* operation now makes greater use of parallel processing and distribution of work across the cluster in order to improve operation performance and reduce the time the operation takes to complete. It continues to be a long-running operation -- particularly in environments with high data volumes -- but not so long-running as it was in prior versions of HyperStore.

Also, more metrics are available to help with troubleshooting operation performance issues or repair failures. For complete metrics on a *repairec* run you can use the command *hsstool opstatus repairec -a*.

More information:

- **"hsstool repairec"** (page 697)
- **"hsstool opstatus"** (page 666)

#### *hsstool proactiverepairq output is now clearer*

The output of the *hsstool proactiverepairq* command now makes clear that this command returns only an estimate of the number of objects in a node's proactive repair queue (whereas the more resource-intensive *hsstool proactiverepairq -a* option returns an exact count).

More information:

- **"hsstool proactiverepairq"** (page 674)

#### */etc/init.d service scripts superseded by systemctl*

The */etc/init.d* scripts are no longer supported as a way to stop or start services on individual HyperStore

nodes. Instead use *systemctl*, which is the preferred method on CentOS 7.

More information:

- **"Start or Stop Services on One Node"** (page 417)

#### *G1GC garbage collection*

HyperStore's Java-based services -- such as the S3 Service, the Admin Service, and Cassandra -- now use the Garbage First Garbage Collector (G1GC), for improved memory management. Previously these services used the Concurrent Mark Sweep (CMS) garbage collector.

Among other benefits, the switch to G1GC allows for larger default heap size limits for HyperStore's Java-based services. For example, the S3 Service's heap size limit is now set to 30gb by default, whereas it previously was set to 8gb.

More information:

- common.csv: **"cloudian\_s3\_heap\_limit "** (page 531) (and subsequent heap size related settings)
- **"View a Node's Memory Usage"** (page 320)

#### *OpenJDK Java platform*

HyperStore now uses OpenJDK as its Java platform. Previously it used Oracle JDK.

There is nothing that you need to do for this change. The HyperStore installer installs the needed OpenJDK versions on your host machines, whether you are doing a fresh install of HyperStore 7.2 or upgrading to it.

#### *Configurable CMC session timeout*

The CMC's session timeout is now configurable. The default is 30 minutes. After a logged-in user has been inactive for this long, the CMC terminates the user's session.

More information:

- common.csv: **"cmc\_session\_timeout"** (page 541)

#### *Configurable connection timeout from CMC to Admin Service*

The connection timeout for when the CMC connects to the Admin Service is now configurable. The default is 10 seconds.

To provide any of its functions for any type of user, the CMC must successfully connect to the Admin Service.

More information:

- mts-ui.properties.erb: **"admin.conn.timeout"** (page 581)

#### *Configurable maximum concurrent cleanup operations per DC*

The maximum number of *hsstool cleanup* or *hsstool cleanupc* operations that can be run concurrently within a data center is now configurable. The default is 1.

More information:

- hyperstore-server.properties.erb: **"max.cleanup.operations.perdc"** (page 550)

#### *Batch processing of object data deletes is now distributed and configurable*

The batch processing job that deletes data from disk after objects have been marked for deletion is now distributed around the cluster, running on each node. Previously this batch process was centered on the cron job host node.

Also, the frequency of the execution of this batch processing job is now configurable. By default it runs hourly, on each node.

More information:

- `mts.properties.erb: "cloudian.delete.queue.poll.interval"` (page 566)

### *JMX now uses fixed ports and internal interface binding*

When JMX communications takes place between HyperStore nodes, it now uses fixed ports and is bound to the internal interface (for nodes that have dual interfaces for internal and public networks).

More information:

- "HyperStore Listening Ports" in the *HyperStore Installation Guide*

### *Health check logging in the S3 request log*

The S3 request log now includes a special type of log entry that identifies health check requests, so that these may be easily distinguished from regular S3 requests.

More information:

- "S3 Service Logs (including Auto-Tiering, CRR, and WORM)" (page 619)
- "Doing an HTTP Health Check" (page 492)

### *Requirement for 128GB RAM for HyperStore host machines (7.2.2)*

Installing or upgrading to HyperStore 7.2.2 requires that all of your HyperStore hosts have at least 128GB RAM. If you are currently running an older version of HyperStore and all or some of your hosts have less than 128GB RAM, contact Cloudian Support.

### *Mandatory admin user password change on first CMC login (7.2.2)*

For fresh installations of HyperStore version 7.2.2 and newer, the first time you try to log in to the CMC as the *admin* user with the default password *public*, you will be required to create a new password.

This does not impact HyperStore systems that are upgraded to version 7.2.2 from an older version.

More information:

- "Cloudian Management Console (CMC)" (page 196)

### *Improved patch process (7.2.2)*

The process for applying a HyperStore patch upgrade has been improved and simplified.

More information:

- "Installing a Patch" (page 62)

### *Redis QoS automatic clean (7.2.2)*

The system now automatically deletes old and no longer needed metadata in the Redis QoS database -- in particular, metadata relating to the automated mapping and cleanup of tombstones in Cassandra.

### *Configurable CMC password restrictions (7.2.2)*

The system now supports configurable restrictions on users' CMC passwords, including minimum password length (which is a minimum of nine characters by default). The other new configurable restrictions (which are disabled by default) are 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.

More information:

- In [common.csv](#), see **"user\_password\_min\_length"** (page 527) and the subsequent settings.

#### *Alerts for SSD failures (Appliance only) (7.2.2)*

For HyperStore Appliances only, an alert is now triggered if an SSD fails, and an error is displayed in the CMC's **Node Status** page.

More information:

- **"View a Node's Disk Detail"** (page 317)
- **"Alerts"** (page 385)

#### *HyperIQ license information (7.2.2)*

The CMC's **Cluster Information** page and the Admin API call `GET /system/license` now indicate your HyperStore system license's level of support for Cloudian HyperIQ integration.

Cloudian HyperIQ is a solution for dynamic visualization and analysis of HyperStore monitoring 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.

#### *Administrator access to users' data via the CMC is disabled by default (7.2.2)*

The ability of system administrators to access and manage all users' data via the CMC, and the ability of group administrators to access and manage the data of users within their group via the CMC, is now disabled by default system configuration. Previously this capability was enabled by default.

More information:

- In *common.csv*: **"cmc\_view\_user\_data"** (page 541)

#### *Read-on-repair improvements (7.2.2)*

The system's "read-on-repair" functionality has been enhanced to be more comprehensive in the types of object data replica problems it can automatically detect and repair.

More information:

- **"Repair-On-Read"** (page 150)

#### *LDAP authentication for system admins (7.2.2)*

LDAP authentication is now supported for users in the System Admin group. Previously it was supported only for regular users.

More information:

- **"LDAP Integration"** (page 131)

#### *Configurable option to suppress alerts for specific log messages (7.2.3)*

HyperStore now has a configuration setting that lets you suppress alerting for specific log messages, based on the message code.

More information:

- In *common.csv*: **"alert\_suppression\_list"** (page 543)

### *HyperStore Shell support for smartctl (7.2.3)*

The Linux command line utility *smartctl* can now be run from within the HyperStore Shell. This utility controls the Self-Monitoring, Analysis and Reporting Technology (SMART) system built into many ATA-3 and later ATA, IDE and SCSI-3 hard drives.

More information:

- **"Using the HSH"** (page 94)

### *Options for displaying custom security information on CMC login page (7.2.3)*

HyperStore now supports two options for displaying custom security information or other company information on the CMC login page:

- You can display custom banner text at the top of the login page.
- You can implement an acknowledgment gate that requires users to acknowledge having read the gate text before being allowed to log into the CMC.

More information:

- **"Configuring a Login Page Banner"** (page 407)
- **"Configuring a Login Page Acknowledgment Gate"** (page 408)

### *hsstool whereis enhancement to detect object corruption (7.2.3)*

The *hsstool whereis* command now supports an option to detect corruption of any of a specified object's replicas or erasure coded fragments.

More information:

- **"hsstool whereis"** (page 721)

### *Additional detail in hsstool metadata response (7.2.3)*

The *hsstool metadata* command now returns additional detailed metadata for the specified object. This detail may be useful if working with Cloudian Support to troubleshoot an issue in regard to the object.

More information:

- **"hsstool whereis"** (page 721)

### *Cross-region replication to external systems no longer an option in the CMC by default (7.2.3)*

The **Cross Region Replication** tab of the CMC's **Bucket Properties** dialog no longer displays settings for replicating to an external system (an S3 system other than the HyperStore system in which the bucket resides). Cloudian, Inc. now discourages use of cross region replication (CRR) to external systems. Instead CRR should be used only for replicating data from one bucket to another bucket within the same HyperStore system. The destination bucket can be in a different **service region** as the source bucket, but both service regions should be part of the same HyperStore system.

If your organization is a legacy user of cross region replication to an external system, and if you want to continue to give that option to your users as they configure their buckets in the CMC, you can re-enable the CMC's display of the relevant CRR settings if you wish. This entails changing a setting in *common.csv*.

More information:

- *common.csv*: **"cmc\_crr\_external\_enabled"** (page 541)

### *IAM request logging (7.2.3)*

There is now a log that records information about requests processed by the HyperStore IAM Service. The log

also records information about requests to the HyperStore STS Service.

Also, new fields have been added to the existing S3 request log, to distinguish S3 requests made by IAM users and by users who have assumed a role and are using temporary credentials issued by the STS Service.

More information:

- **"IAM Service Logs"** (page 613)
- **"S3 Service Logs (including Auto-Tiering, CRR, and WORM)"** (page 619)

#### *Automatic case creation for failed disks (7.2.3)*

As an enhancement to the Smart Support feature, if a data disk on a HyperStore node fails, information about the failed disk is now automatically sent to Cloudian Support within minutes. This results in the automatic creation of a Support case for the failed disk.

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

More information:

- **"Smart Support and Diagnostics Feature Overview"** (page 190)
- **"Automated Disk Management Feature Overview"** (page 157)

#### *New cleanup option to more efficiently target remnant data from deleted objects (7.2.3)*

The *hsstool cleanup* and *hsstool cleanupec* operations now include a *"-no"* option that has the cleanups focus on removing remnant data from objects that have been deleted from the system (through the S3 interface or the Admin API), while ignoring data that doesn't belong to the target node's token ranges (which can be time consuming to evaluate and process). This helps the cleanup operation more efficiently and quickly free up disk space after a large number of objects have been deleted.

This option is supported only on the command line -- not in the CMC's interface for these cleanup operations.

More information:

- **"hsstool cleanup"** (page 644)
- **"hsstool cleanupec"** (page 651)

#### *LDAP authentication for HyperStore Shell users (7.2.3)*

HyperStore now supports LDAP authentication for HyperStore Shell (HSH) users. This is supported only for system admin users created while you are running HyperStore version 7.2.3 or later, and only if you have enabled LDAP authentication for the System Admin group.

More information:

- **"LDAP Integration"** (page 131)
- **"Enabling the HSH and Managing HSH Users"** (page 90)

#### *Improved performance for auto-tiering and auto-expiration execution (7.2.3)*

The daily cron job that executes auto-tiering and auto-expiration of objects (in accordance with bucket lifecycle policies) now distributes the required processing work across all nodes in the same service region as the cron job primary node. Previously all the work was done by the cron job primary node itself.

More information:

- **"System cron Jobs"** (page 473)
- **"Auto-Tiering Feature Overview"** (page 176)

- **"Configure a Bucket Lifecycle Policy for Object Auto-Tiering or Expiration"** (page 227)

## Documentation -- New Features and Enhancements

*S3 API support documentation now matches new AWS S3 operation naming scheme (7.2.3)*

The documentation for HyperStore support of the AWS S3 REST API has been updated to match the operation naming scheme changes that were recently made by AWS. In most cases the operation name change is trivial. For example, what the AWS documentation -- and HyperStore documentation -- used to call "DELETE Object" is now called "DeleteObject". In some cases though the name change is more substantial -- for example what used to be named "POST Object Restore" is now named "RestoreObject", and what was formerly "PUT Bucket" is now "CreateBucket". In the updated HyperStore documentation that now uses the new naming scheme, for each operation there is a note indicating what the former name of the operation was.

More information:

- Section 13.2 "Supported S3 API Operations"

# Chapter 1. Introduction

## 1.1. HyperStore Documentation

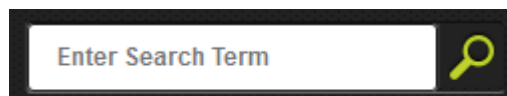
The HyperStore user documentation consists of:

- HyperStore Help (HTML5)
- HyperStore Administrator's Guide (PDF)
- HyperStore Installation Guide (PDF)
- HyperStore Quick-Start for Software-Only Users (PDF)

The Help is available through the CMC (by clicking the Help button) and is also available in the directory `<installation-staging>/doc/HyperStoreHelp` on each of your HyperStore nodes (in that directory you can open the `HyperStoreHelp.html` file). The PDF guides are available in the directory `<installation-staging>/doc/HyperStorePDFManuals` on each HyperStore node.

**The Help has the exact same content as the Installation Guide and Administrator's guide**, just in HTML rather than PDF. Further, starting with section "1. Introduction to HyperStore", the Help uses the exact same section numbering as is used in the Administrator's Guide -- so for example, section 4.1.2 in the Help is the same content as section 4.1.2 in the Administrator's Guide.

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 it.

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:

### 12.5 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.

- ▶ 12.5.1 DELETE /group Delete a group
- ▶ 12.5.2 GET /group Get a group's profile
- ▶ 12.5.3 GET /group/list Get a list of group profiles
- ▶ 12.5.4 GET /group/ratingPlanId Get a group's rating plan ID
- ▶ 12.5.5 POST /group Change a group's profile
- ▶ 12.5.6 POST /group/ratingPlanId Assign a rating plan to a group
- ▶ 12.5.7 PUT /group Create a new group

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

## 12.5 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.

▼ 12.5.1 DELETE /group Delete a group

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

The request line syntax for this method is as follows.

```
DELETE /group?groupId=xxx
```

There is no request payload.

**Example Using cURL**

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

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

**Response Format**

There is no response payload. For response status code this method will return either one of the [12.1.3 Common Response Status Codes](#) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameters : {groupId}

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



If you have a comment or request regarding the HyperStore documentation, please send it to this email address:

*cloudian-pubs@cloudian.com*

You will not receive a reply, but the Cloudian Technical Publications team will review your comment and, if appropriate, redress the issue in an upcoming HyperStore release. Thank you for your feedback.

## 1.2. HyperStore Overview

Cloudian 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 HTTP 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 users 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.
- **Group support.** An enterprise or work group can share a single HyperStore account. Each group member can have dedicated storage space, and the group can be managed by a designated group administrator.
- **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).
- **Horizontal scalability.** Running on commodity off-the-shelf hardware, a HyperStore system can scale up to thousands 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 16)
- **"Licensed Maximum On-Premise Storage Usage"** (page 17)
- **"Licensed Maximum Tiered Storage Usage"** (page 18)
- **"WORM (Object Lock) License"** (page 20)
- **"HyperIQ License"** (page 20)
- **"License Updating"** (page 20)
- **"Auditing"** (page 21)

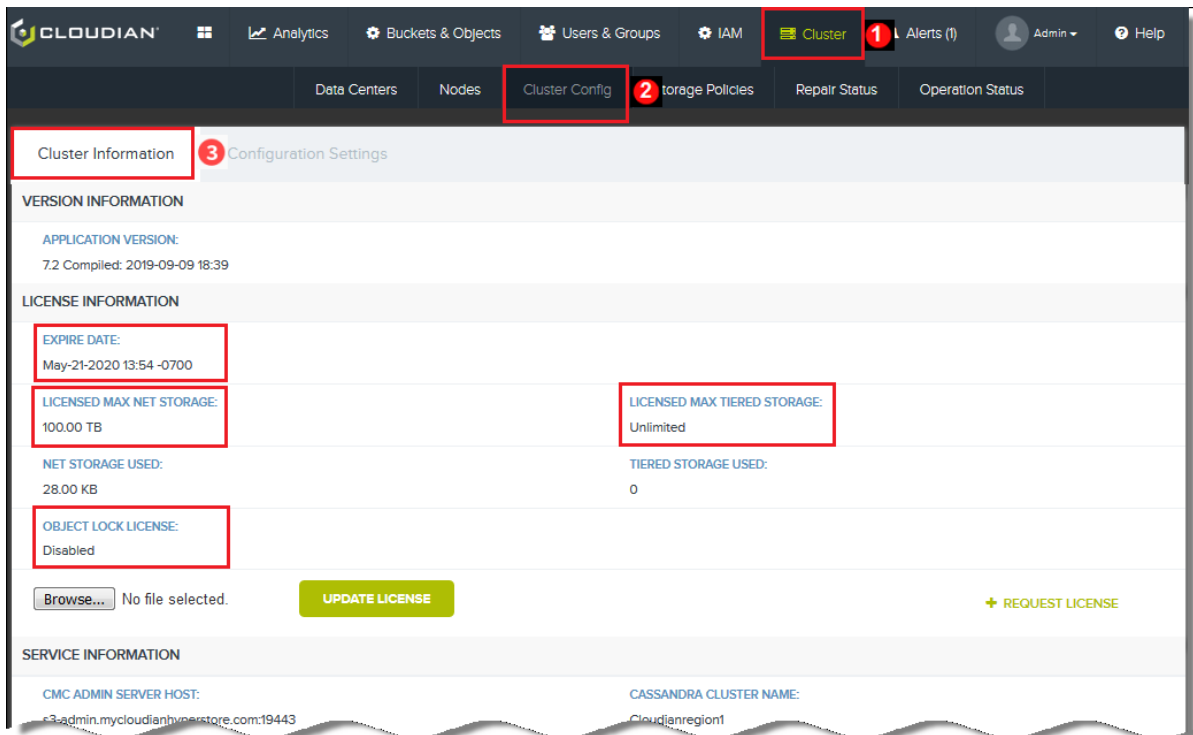
A valid Cloudian software license is required to run HyperStore software. Evaluation licenses are available as well as production licenses. Before using HyperStore software, you must obtain a license from Cloudian.

A Cloudian HyperStore license has four key attributes:

- Expiration date
- Maximum allowed on-premise storage volume

- Maximum allowed tiered storage volume
- Object lock functionality enabled or disabled

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 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. All incoming S3 requests will be rejected with a "503 Service Unavailable" 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 you will not be able to access users' stored S3 objects.
- The top of the CMC screen will display an error message indicating that your license has expired and that your HyperStore system has been disabled. Also, in the CMC's [Dashboard](#) page the Cluster Health panel will indicate that the system is disabled.
- If you stop the S3 Service on a node you will not be able to restart it. This applies also to the Admin Service and IAM Service, since those services stop and start together with the S3 Service.

It's best to update your license well in advance of your license expiration date. See **"License Updating"** (page 20) 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, the limit is on total object storage bytes minus overhead from [storage policies](#) (object replication or erasure coding). For example if a 1GB object is replicated three times in your system it counts as only 1GB 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 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 1GB object is replicated three times in your system it counts as 3GB 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 Cloudian 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 Cloudian 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 [hourly system cron job for processing the object deletion queue](#) runs. 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 20)). 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 Management Feature Overview"** (page 157).

### 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. For more information on versioning see **"Set Versioning for a Bucket"** (page 239).

### 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 Feature Overview"** (page 186).

### 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 176).

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 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 the 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 184).

- You acquire and install a new license with a larger tiered storage maximum (see "**License Updating**" (page 20)). 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 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. WORM (Object Lock) License

Your license may or may not include support for the HyperStore WORM (Object Lock) feature. To check whether your license supports this feature, see the CMC's [Cluster Information](#) page: it will indicate either "Object Lock License: Enabled" or "Object Lock License: Disabled". 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.

If your current license does not support Object Lock and you want to use this feature, contact Cloudian Support for information about a license that supports Object Lock.

For more information about this feature, see **"WORM (Object Lock)"** (page 121).

### 1.3.5. HyperIQ License

Cloudian HyperIQ is a solution for dynamic visualization and analysis of HyperStore monitoring 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](mailto:cloudian-license@cloudian.com).

Once you've obtained a new license file you can use the CMC's Cluster Information page to dynamically apply the new license file to your HyperStore system. For instructions see **"Install a New License File"** (page 337).

### 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. HyperStore Services

The Cloudian HyperStore™ 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 the HyperStore installation script distributes these services across a multi-node cluster. There are common services that are installed to and run on every node, and specialized support services that are installed and run on only one or a sub-set of nodes.

For services distribution diagrams, see **"Services Distribution -- 3 Nodes, Single DC"** (page 31).

**Note** Within your installation 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. If you want to modify these assignments after install, see **"Change Node Role Assignments"** (page 457).

If you installed HyperStore software on only one node, then all these services will run on that node.

Service Category	Service	Where It Is Installed
Common Services	<a href="#">S3 Service</a>	Every node.
	<a href="#">HyperStore Service</a>	Every node.
	<a href="#">Cassandra Service</a>	Every node.
	<a href="#">Admin Service</a>	Every node.
	<a href="#">IAM Service</a>	Every node in the default service region.
	<a href="#">STS Service</a>	Every node in the default service region.
	<a href="#">SQS Service</a>	Every node.
	<a href="#">Cloudian Management Console (CMC)</a>	Every node.
Specialized Support Services	<a href="#">Redis Credentials DB Master</a>	One node per entire HyperStore system.
	<a href="#">Redis Credentials DB Slaves</a>	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 Redis Credentials slaves. For instructions on adding

Service Category	Service	Where It Is Installed
		slaves, see <b>"Move or Add a Redis Credentials Slave or Redis QoS Slave"</b> (page 461). (Note: If you upgraded from a HyperStore version older than 6.0, you will have only one Redis Credentials slave per data center.)  Slaves will not be on same node as Credentials master
	<a href="#">Redis QoS DB Master(s)</a>	One node per service region.
	<a href="#">Redis QoS DB Slave(s)</a>	One node per data center. Slave will not be on same node as QoS Master.
	<a href="#">Redis Monitor</a>	One primary node and one backup node per entire HyperStore system.
	<a href="#">Crontab configuration and Monitoring Data Collector</a>	One primary node and one backup node per service region.
	<a href="#">Local NTP server</a>	One local NTP server per service region.
	<a href="#">Puppet Master</a>	One primary node and one backup node per entire HyperStore system.

### 1.4.1. S3 Service

The HyperStore system provides a high-performance S3 proxy service. The S3 Service processes S3 REST requests incoming from client applications (including the Cloudbian Management Console). On the back side, the S3 Service interfaces with:

- The HyperStore Service, Cassandra "UserData\_<policyid>" keyspaces, and Cassandra "ECKeyspace" keyspace in order to store, retrieve, and delete users' S3 data objects.
- The Cassandra "AccountInfo" keyspace to update and retrieve user account information.
- The Cassandra "Reports" keyspace to update users' transaction history.
- The Redis "Credentials" DB to implement user authentication, S3 bucket validation, and other functions in support of S3 request processing.
- The Redis "QoS" DB to enforce group and user level quality of service restrictions.

The S3 Service is built on [Jetty](#) server technology.

## 1.4.2. HyperStore Service and the HSFS

As an object store, Cassandra provides a wealth of valuable built-in 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 on 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 76).

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\_data\_directory**" (page 517) setting in *common.csv*.
- 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 (as specified by the configuration setting *common.csv: hyperstore\_data\_directory*)
- "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.

#### 1.4.2.1. 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 643) operations such as repair and cleanup. The digests are stored in high-performance [RocksDB](#) databases (persistent key-value stores) on the same mount point as the corresponding file. On each mount point, there is one RocksDB database for storing digests for replica data files, and one RocksDB database for storing digests for erasure coded data files.

For replica data files, the digest database is stored under:

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

For erasure coded data files, the digest database is stored under:

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

Within each database, 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).

**1.4.2.1.1. 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/1mDFFH13tL1DIsYhNKDX3d/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 Rocks DB database 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.

### 1.4.3. Cassandra Service

The HyperStore system uses the Apache open source storage platform [Cassandra](#) to store several types of data. The HyperStore system creates and uses several distinct "keyspaces" (approximately equivalent to databases) 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 Feature Overview"** (page 164).

**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 76).

- 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 **"HyperStore Support for the AWS SQS API"** (page 1041).

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

### 1.4.4. Redis Credentials and Redis QoS Services

The HyperStore system uses the lightweight, open source [Redis](#) key-value data store to store a variety of data that supports HyperStore S3 service features. There are two types of Redis DBs in a HyperStore deployment:

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

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

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

#### *Note for multi-region systems*

In a multi-region HyperStore deployment there will be:

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

### 1.4.4.1. Redis Node Roles

Each Redis DB is implemented across two or more nodes, with the nodes playing different roles. These roles are:

- **master** — All write requests from Redis clients are implemented on the master node. There is only one master node for each Redis DB.

#### *Note for multi-region systems*

In a multi-region HyperStore deployment, the universal Redis Credentials DB has one master node and each regional Redis QoS DB has its own master node.

- **slave** — In each Redis DB, data from the Redis 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 Redis 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"** (page 27).

Redis roles are assigned to your HyperStore nodes automatically during installation.

### 1.4.5. Redis Monitor Service

The Redis Monitor monitors Redis Credentials DB and Redis QoS DB cluster health and implements automatic failover of the Redis master node role within each of the two Redis DBs. For redundancy, the Redis Monitor runs on two HyperStore nodes, configured as primary on one node and as backup on the other node.

If the Redis Monitor detects that a Redis master node has gone down, it promotes an available slave node to the master node role; and informs the Redis cluster's clients (the S3 Service, IAM Service, Admin Service, HyperStore Service) of the identity of the new master.

**Note** In a multi-DC system the HyperStore installer puts the Redis Monitor backup in the same DC as the Redis Monitor primary. Keep them in the same DC -- do **not** migrate them such that the primary and backup are in different DCs.

In a **multi-region HyperStore deployment**, a single Redis Monitor instance will monitor multiple regional Redis QoS DBs as well as the one universal Redis Credentials DB. Each regional Redis QoS DB will have its own configured cluster membership list that the Redis Monitor will refer to if a slave needs to be promoted to master.

### 1.4.6. 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.

For information on the Admin API see **"HyperStore Admin API Introduction"** (page 741).

The **"Cloudian Management Console (CMC) Service"** (page 28) is a client to the Admin Service. You also have the option of building your own Admin Service client.

The Admin Service is closely integrated with the **"S3 Service"** (page 22). Both leverage [Jetty](#) technology; both are installed together; and both are started and stopped together by the same commands.

### 1.4.7. IAM, STS, and SQS Services

The HyperStore IAM, STS, and SQS services provide limited 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 IAM Service you can use the CMC as an IAM client, or use a third party or custom IAM 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 implement of these services see

- **"HyperStore Support for the AWS IAM API"** (page 991)
- **"HyperStore Support for the AWS STS API"** (page 1037)
- **"HyperStore Support for the AWS SQS API"** (page 1041)

### 1.4.8. Cloudian Management Console (CMC) Service

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).

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

- 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.
- Setting access control rights on users' buckets and stored objects.

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

The CMC acts as a client to the **"Admin Service"** (page 28) and the **"S3 Service"** (page 22).

### 1.4.9. Supporting Services

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

- **Cloudian Monitoring Agent** — The Cloudian Monitoring Agent runs on each HyperStore node and monitors node health and performance statistics. The Agent also plays a role in the triggering of event notification emails to system administrators. System and node statistics are viewable through the CMC; and you can configure event notification rules through the CMC as well.
- **Cloudian Monitoring Data Collector** — The Cloudian Monitoring Data Collector runs (together with the system maintenance cron jobs) on one node in each of your service regions, and regularly collects data from the Monitoring Agents. The Monitoring Collector writes its collected node health statistics to Cassandra's "Monitoring" keyspace. The Monitoring Collector is also configured (together with the cron jobs) 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.
- **Puppet** — As part of the HyperStore software installation, the HyperStore installer installs the open source version of [Puppet](#) and uses it to implement initial HyperStore system configuration. HyperStore also uses Puppet for support of ongoing configuration management. For more information see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506). Puppet agents run on every node. The Puppet Master runs on one node and is also configured on a backup node. Manual failover to the backup is supported if the primary Puppet Master instance goes down.
- **Pre-Configured *ntpd*** — 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 *ntpd*. In each HyperStore data center four of your HyperStore nodes are automatically configured to act as internal NTP servers, which synchronize with external NTP servers (by default the servers from the *pool.ntp.org* project). Other HyperStore hosts in each data center are configured as clients of the internal NTP servers. For more information see **"NTP Automatic Set-Up"** (page 598).

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.

**Note** 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.

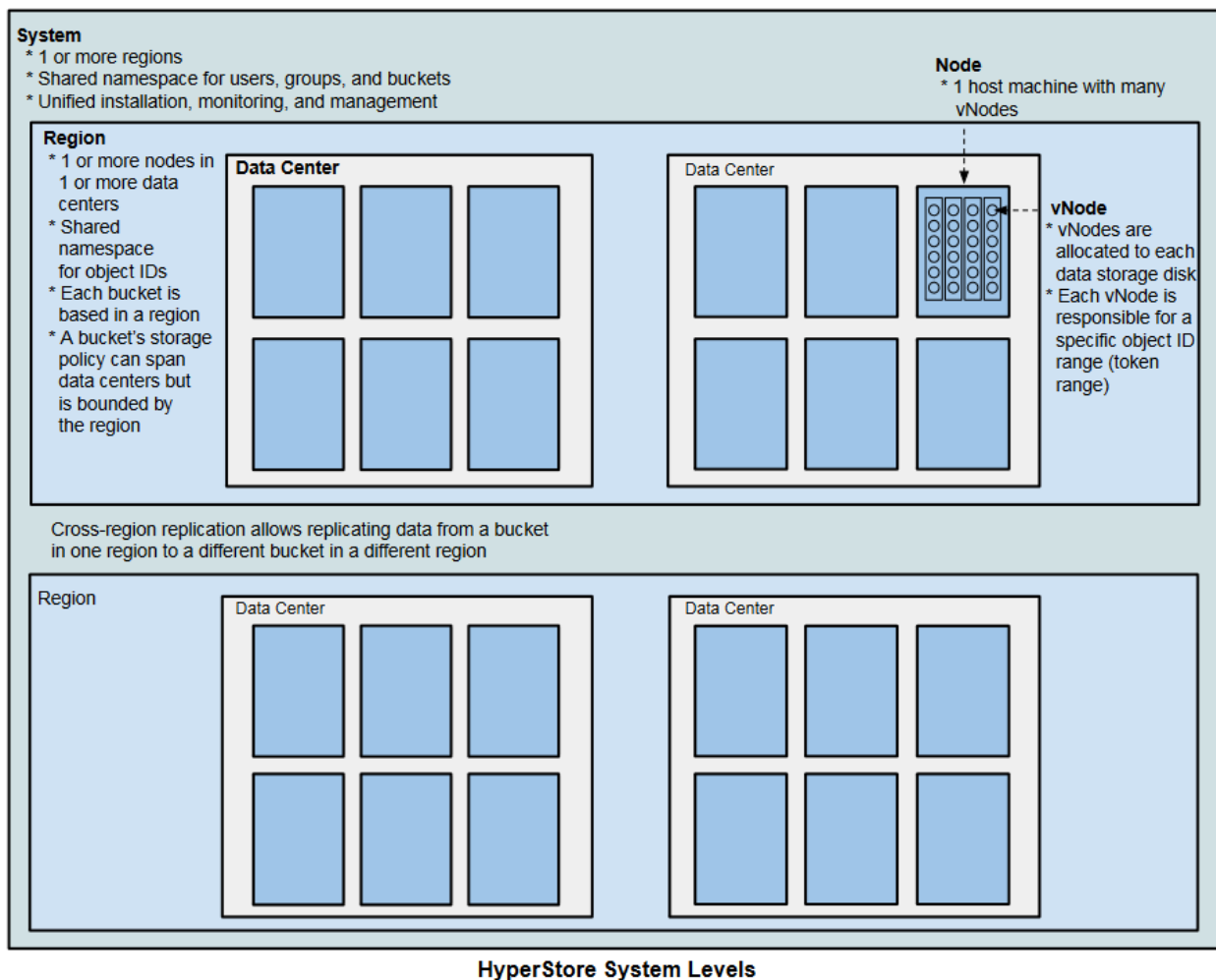
- **Dnsmasq** — [Dnsmasq](#) is a lightweight domain resolution utility. This utility is bundled with Cloudian HyperStore software. The HyperStore interactive installation wizard gives you the option to have *dnsmasq* installed and configured to resolve HyperStore service domains (specifically the S3 service domain, the S3 website endpoint domain, and the CMC domain). The *dnsmasq* utility may be helpful if you are evaluating a small HyperStore system but it is not appropriate for production use.

## 1.5. System Diagrams

### 1.5.1. 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:

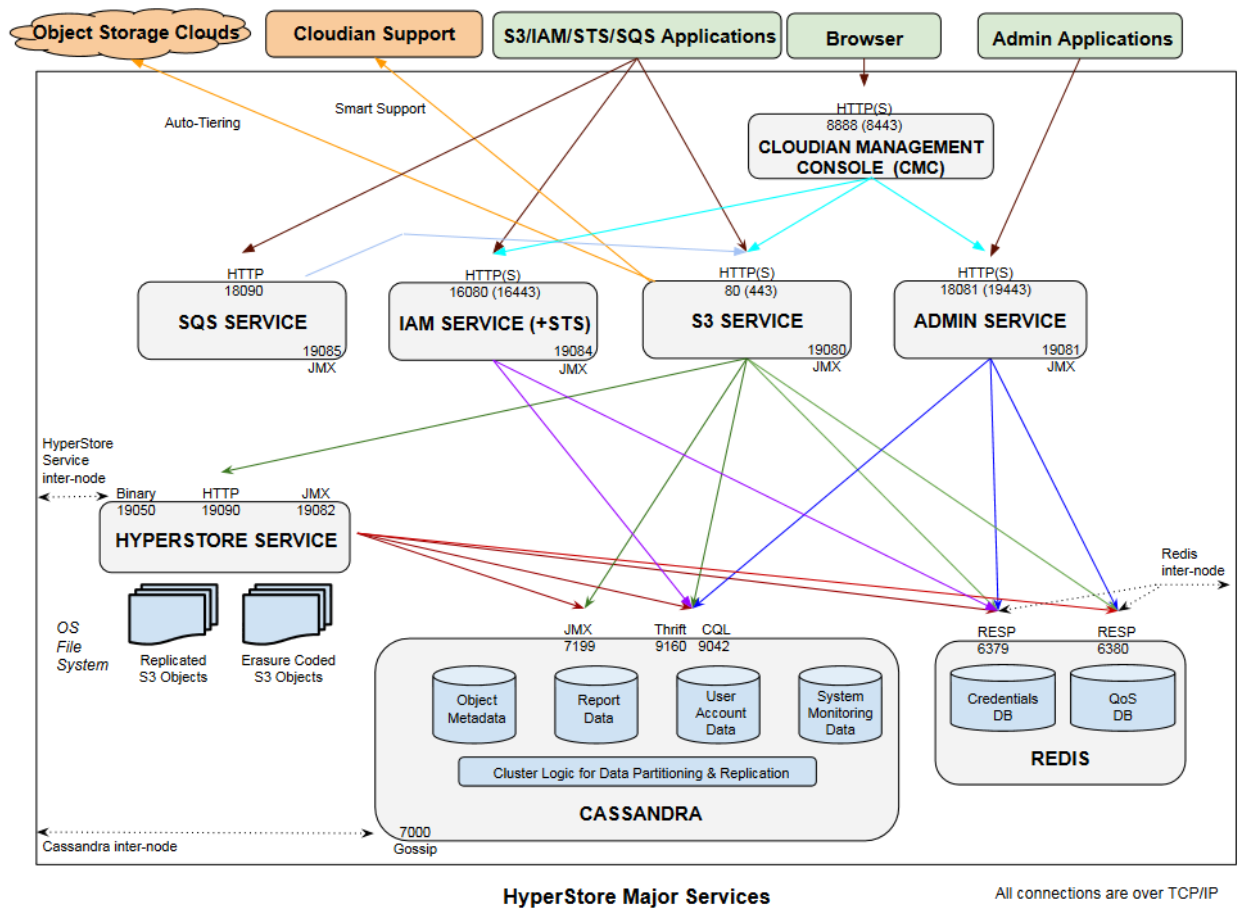
- **System**
- **Region** (also known as a "**Cluster**")
- **Data Center**
- **Node**
- **vNode**



HyperStore System Levels

### 1.5.2. Service Interconnections

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



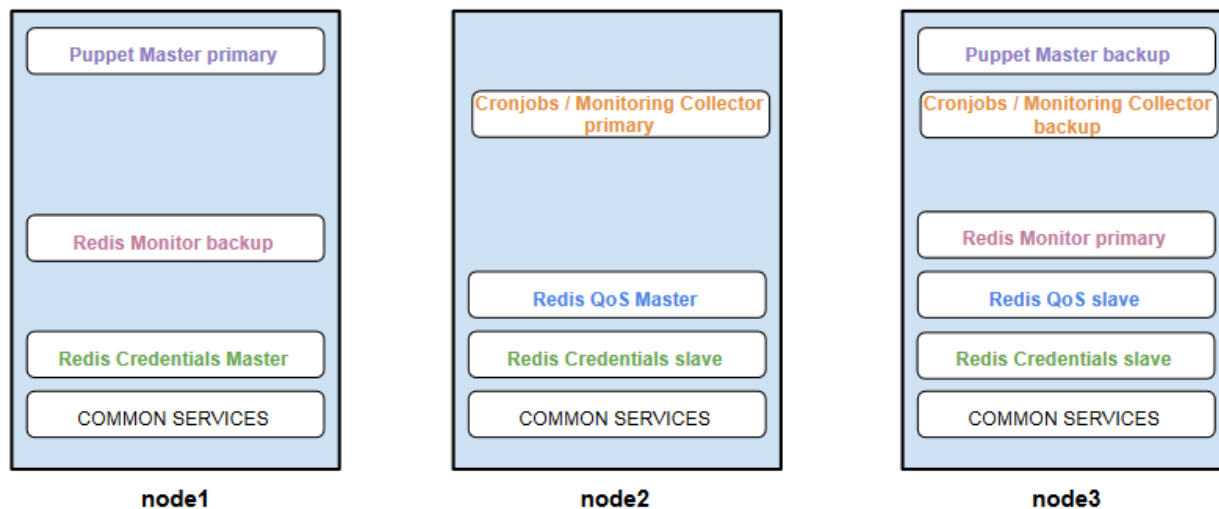
**Note** With the exception of the Redis DBs and the IAM/STS Services, each of the services shown in the diagram run on every HyperStore node. The diagram excludes certain supporting services such as the Redis Monitor, the Monitoring Data Collector, and Puppet. For a complete list of HyperStore services and the listening ports they use, see "HyperStore Listening Ports" in the Reference section of the *HyperStore Installation Guide*.

### 1.5.3. 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, Admin Service, HyperStore Service, Cassandra, CMC, Puppet Agent, 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 Puppet Master or Redis Monitor), the backup resides on a different node than the primary. Likewise the Redis QoS slave will reside on a different node than the Redis QoS master, and the two Redis Credentials slaves will reside on different nodes than the Redis Credentials 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 Cronjob instance and one backup

instance) — the only difference is that this fixed set of specialized service instances will be spread across your cluster rather than concentrated among three nodes as shown in the diagram.



**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. 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 457).

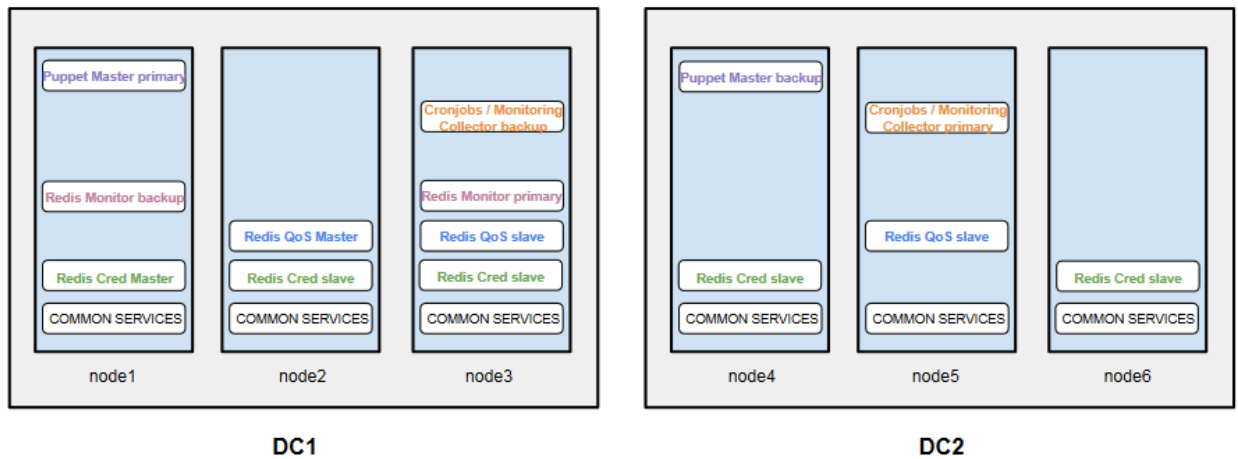
**Note** If you have a very large cluster (25 nodes or more in a data center), consult with Cloudbian Support about whether you should add more Redis Credentials slaves. For instructions on adding Credentials slaves, see **"Move or Add a Redis Credentials Slave or Redis QoS Slave"** (page 461).

**Note** Starting with HyperStore 7.2, the Salt master is located with the Puppet master.

### 1.5.4. 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, Admin Service, HyperStore Service, Cassandra, CMC, Puppet Agent, and Monitoring Agent) run on every node in your multi-DC system.
- Each data center has its own Redis QoS slave and its own two Redis Credentials slaves, for Redis read performance optimization.
- The Puppet Master backup is placed in a different DC than the Puppet Master 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 Redis Credentials master is located.

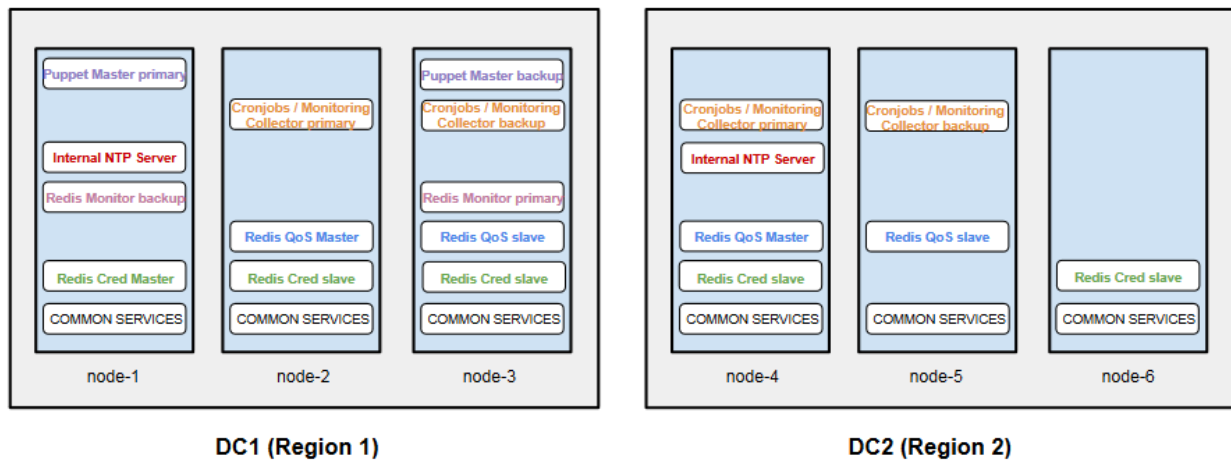
**Note** To check the current location of specialized services within your multi-DC HyperStore system, go to the CMC's [Cluster Information](#) page.

**Note** Starting with HyperStore 7.2, the Salt master is located with the Puppet master.

### 1.5.5. 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, Cassandra, CMC, Puppet Agent, and Monitoring Agent) run on every node in your multi-region system.
- The whole multi-region system is served by a single active Puppet master and a single Redis Credentials master.
- Each region has its own Redis QoS master and its own active Cronjob host.



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

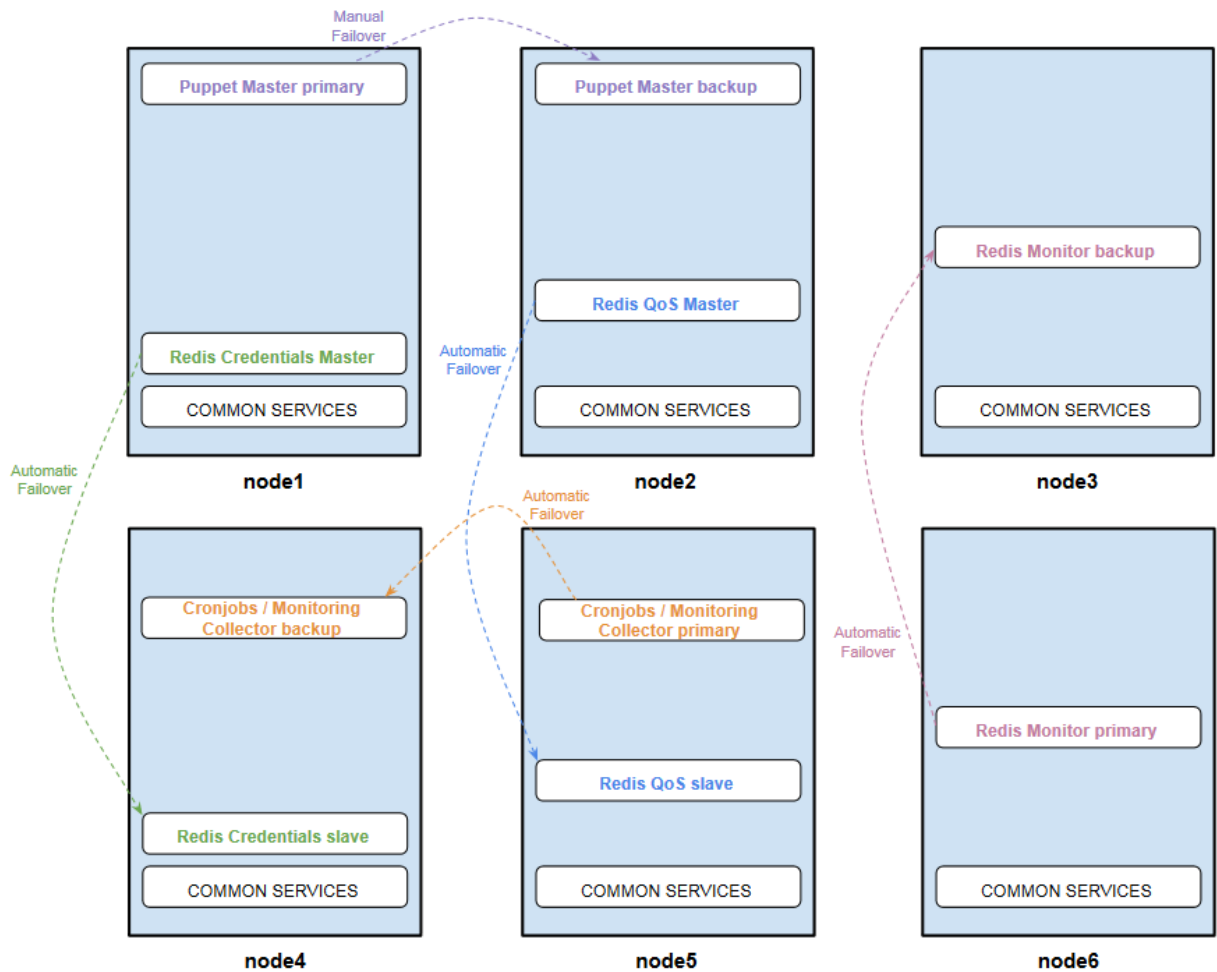
**Note** Starting with HyperStore 7.2, the Salt master is located with the Puppet master.

### 1.5.6. Specialized Services Availability

Along with the services that are common to every HyperStore node (such as the S3 Service, HyperStore Service, Cassandra, 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 Puppet Master role (for managing system configuration) — in the case of the Puppet Master you can [manually implement failover](#) if there's a problem with the primary instance.

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** 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.

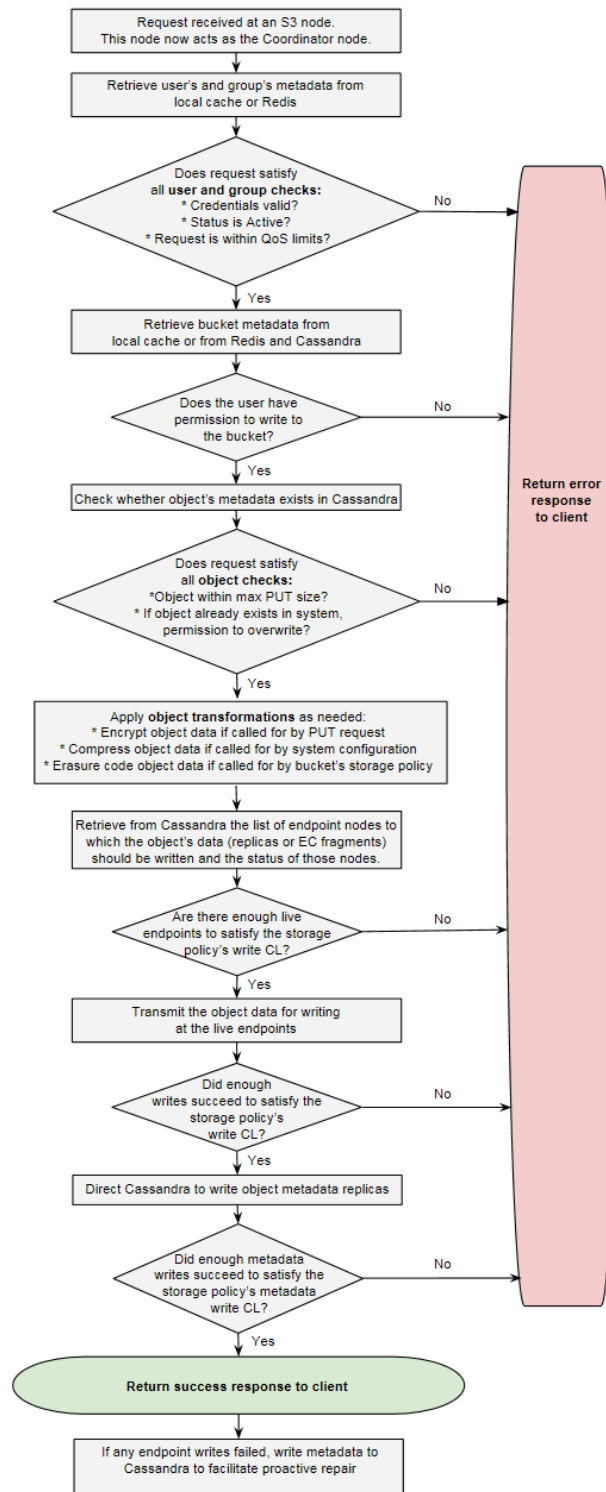
Also, the automatic failover of the Cronjobs and Monitoring Data Collector roles from the primary to the backup instance will not occur until the primary instance has been down for 10 minutes.

**Note** Starting with HyperStore 7.2, the Salt master is located together with the Puppet master, and if you manually fail over the Puppet master role to the backup node, the Salt master role moves to that backup node as well.

### 1.5.7. S3 PUT Processing Flow

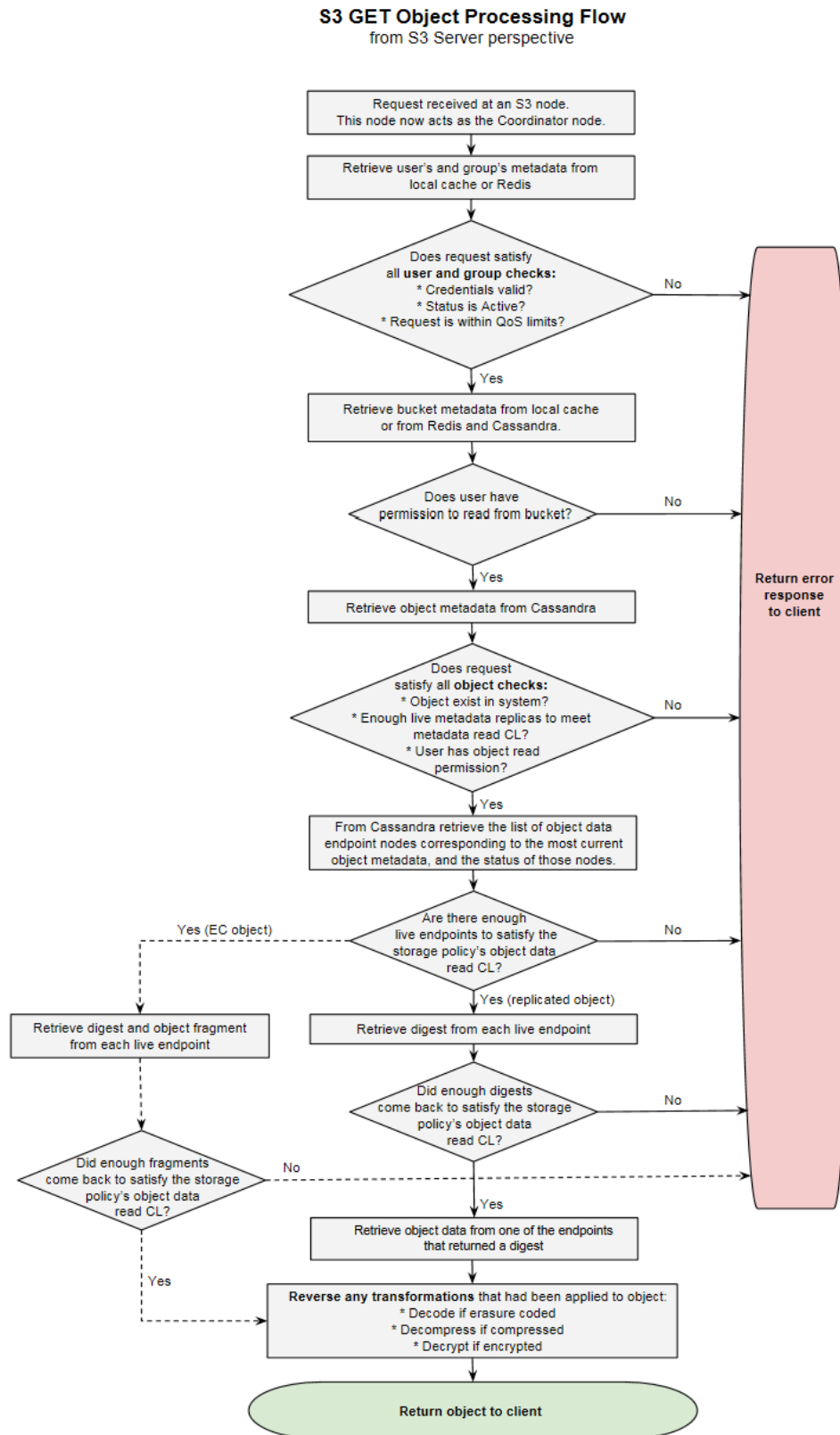
The diagram below shows the main aspects of how the HyperStore system processes an S3 PUT Object request. The flow is presented from the perspective of the S3 Service, which handles incoming S3 requests. The S3 Service runs on all of the nodes in your cluster.

### S3 PUT Object Processing Flow from S3 Server perspective



### 1.5.8. S3 GET Processing Flow

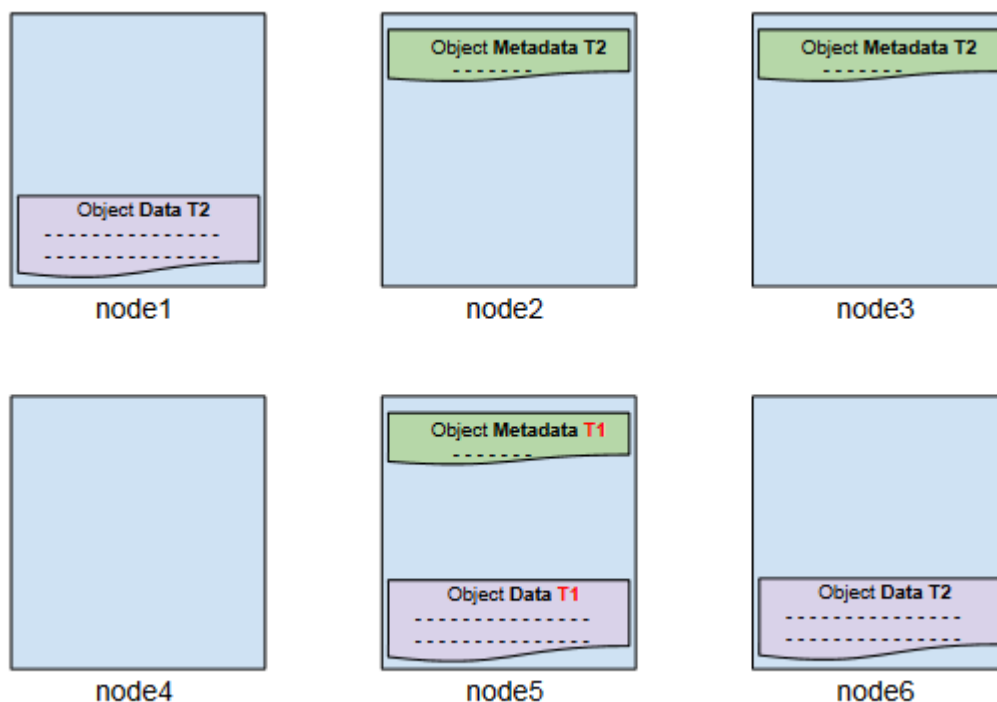
This diagram shows the main aspects of how the HyperStore system processes an S3 GET Object request, from the perspective of the S3 Service.



### 1.5.9. Data Freshness for Replicated Object Reads

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.

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 outdated 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 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.

**Note** HyperStore allows you to configure 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 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 84).

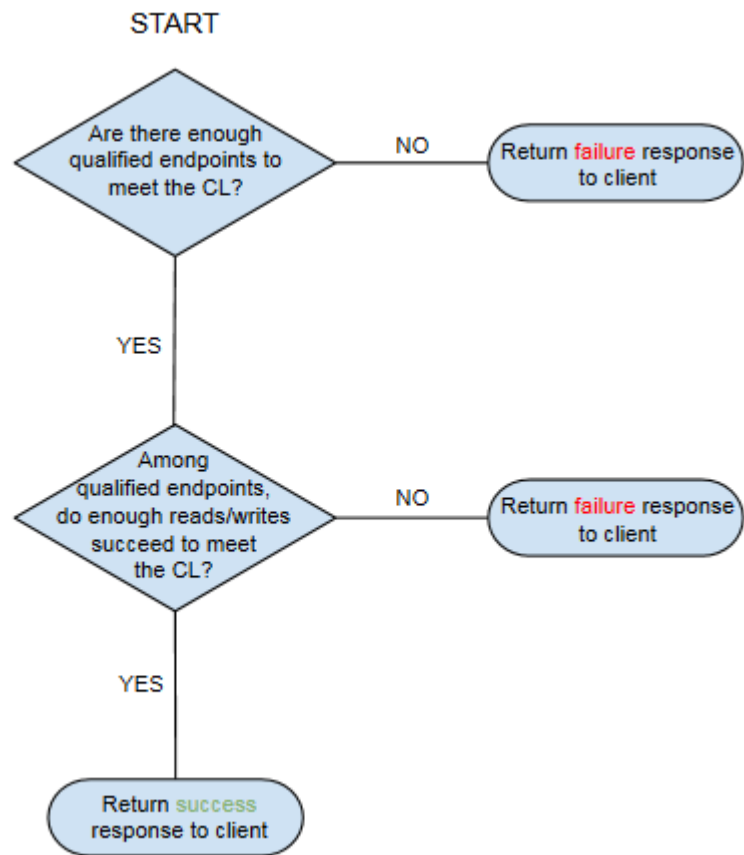
### 1.5.10. Dynamic Consistency Levels

When you **"Add a Storage Policy"** (page 353) 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 because there are not enough qualified endpoints, then the system tries to meet the less strict level (the "fallback" consistency level).

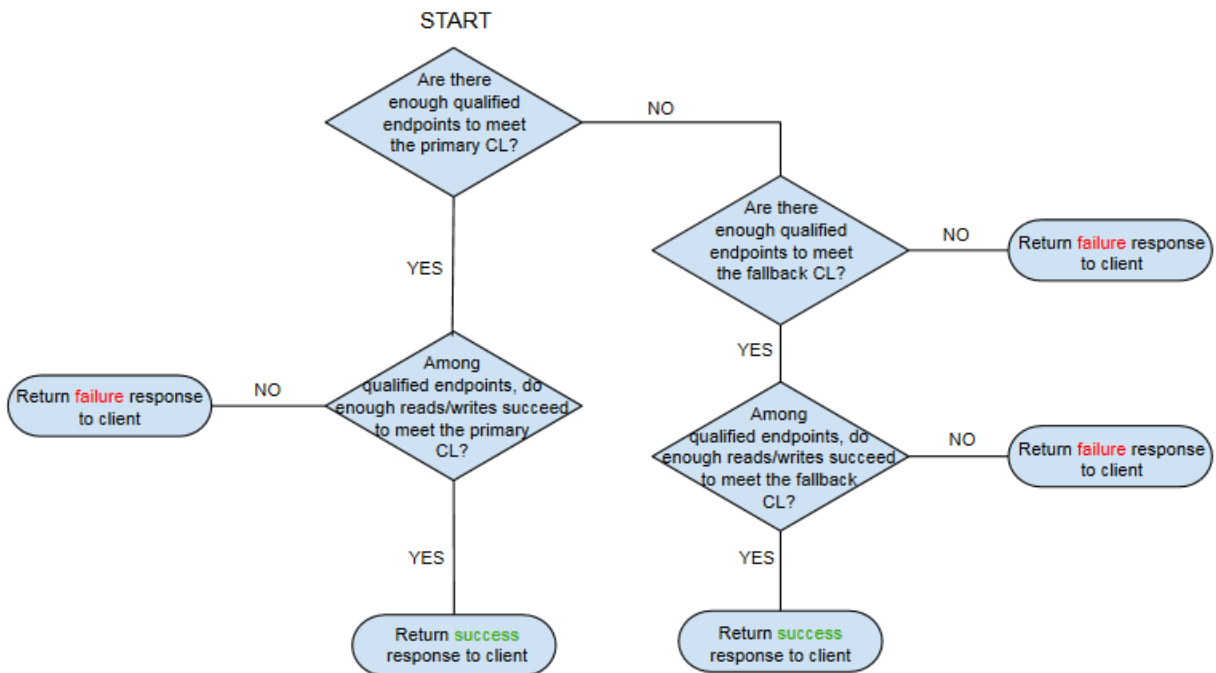
**Note** Although HyperStore applies dynamic consistency level logic only to the writing and reading of **object metadata** (and not object data), this still has the effect of controlling overall S3 request success or failure logic -- since for an S3 write or read operation to succeed it must succeed for the object metadata as well as the object data.

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".

### 1.5.10.1. Standard Consistency Level Logic



## 1.5.10.2. Dynamic Consistency Levels Logic



## 1.5.10.3. Dynamic Consistency Levels Logic Described

When the S3 Service processes an S3 request such as a PUT or a GET of an object, it checks the system for a list of "endpoints" for that particular object — the nodes that the object data and object metadata should be written to or read from. The system determines the endpoint node list based on the object token and the applicable storage policy (such as 3X replication or 4+2 erasure coding). The system then checks each of the endpoint nodes for any of these disqualifying conditions:

- Cassandra Service is down or unreachable
- HyperStore Service has been marked as down by the S3 Service (see **"Node Status Configuration"** (page 578))
- Node has been put into Maintenance Mode by operator (see **"Start Maintenance Mode"** (page 329))
- Node is in a StopWrite condition due to all data disks being 90% full or more (disqualifying only for write requests, not read requests; see **Automatic Stop of Writes to a Node**)
- Disk on which requested data resides is [disabled](#) (relevant only for read requests)

**If the number of qualified endpoint nodes -- endpoint nodes free of any of these disqualifying conditions -- is enough to achieve the primary CL**, the system proceeds with trying to achieve the primary CL. If enough endpoint writes or reads succeed to achieve the primary CL, a success response is returned to the S3 client. If not -- such as if an error is encountered at one of the endpoints during the attempted write or read of object metadata -- then the S3 request fails and an error is returned to the S3 client. The system will **not** try to achieve the fallback CL in this scenario.

**If the number of qualified endpoint nodes is too few to achieve the primary CL but enough to achieve the fallback CL**, the system proceeds with trying to achieve the fallback CL. If enough endpoint writes or reads succeed to achieve the fallback CL, a success response is returned to the S3 client. If not then the S3 request fails and an error is returned to the S3 client.

If the number of qualified endpoint nodes is too few to achieve the fallback CL, then the S3 request fails and an error is returned to the S3 client.

As an example, with a Replication Within Single Data Center storage policy, for the object Write CL configuration you could select the "ALL" CL and also the (less stringent) "QUORUM" CL. With this configuration, for a given request to write an S3 object, if the number of qualified endpoints is enough to achieve the ALL level, the system will try to achieve the ALL level for the operation. If the number of qualified endpoints is not enough to achieve ALL but is enough to achieve QUORUM, the system will try to achieve QUORUM for the operation. If the number of qualified endpoints is not enough to achieve even the QUORUM level, the operation will fail.

It's important to note that once the system determines that enough qualified endpoints are available to try to meet the stricter of the configured CLs, the system is then committed to that path in terms of the object metadata writes or reads. If there is a subsequent failure on that path — such as a metadata write failure on one or more of the endpoints — then the request fails and an error is returned to the client. The system does not go back and try to achieve the less-strict CL in this scenario.

### 1.5.11. 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 down 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

Number of Data Disks on Host	Size of Data Disks	Total TBs of Data Disk on Host	Number of Tokens Assigned To Host
			(1TB X .7 = .7, but minimum is 1 token per data disk)
4	8TB	32TB	22 (32TB X .7 = 22.4, rounded down = 22)
8	4 X 8TB 4 X 10TB	72TB	50 (72TB X .7 = 50.4, rounded down = 50)
12	10TB	120TB	84 (120TB X .7 = 84)
24	16TB	384TB	268 (384TB X 0.7 = 268.8, rounded down = 268)

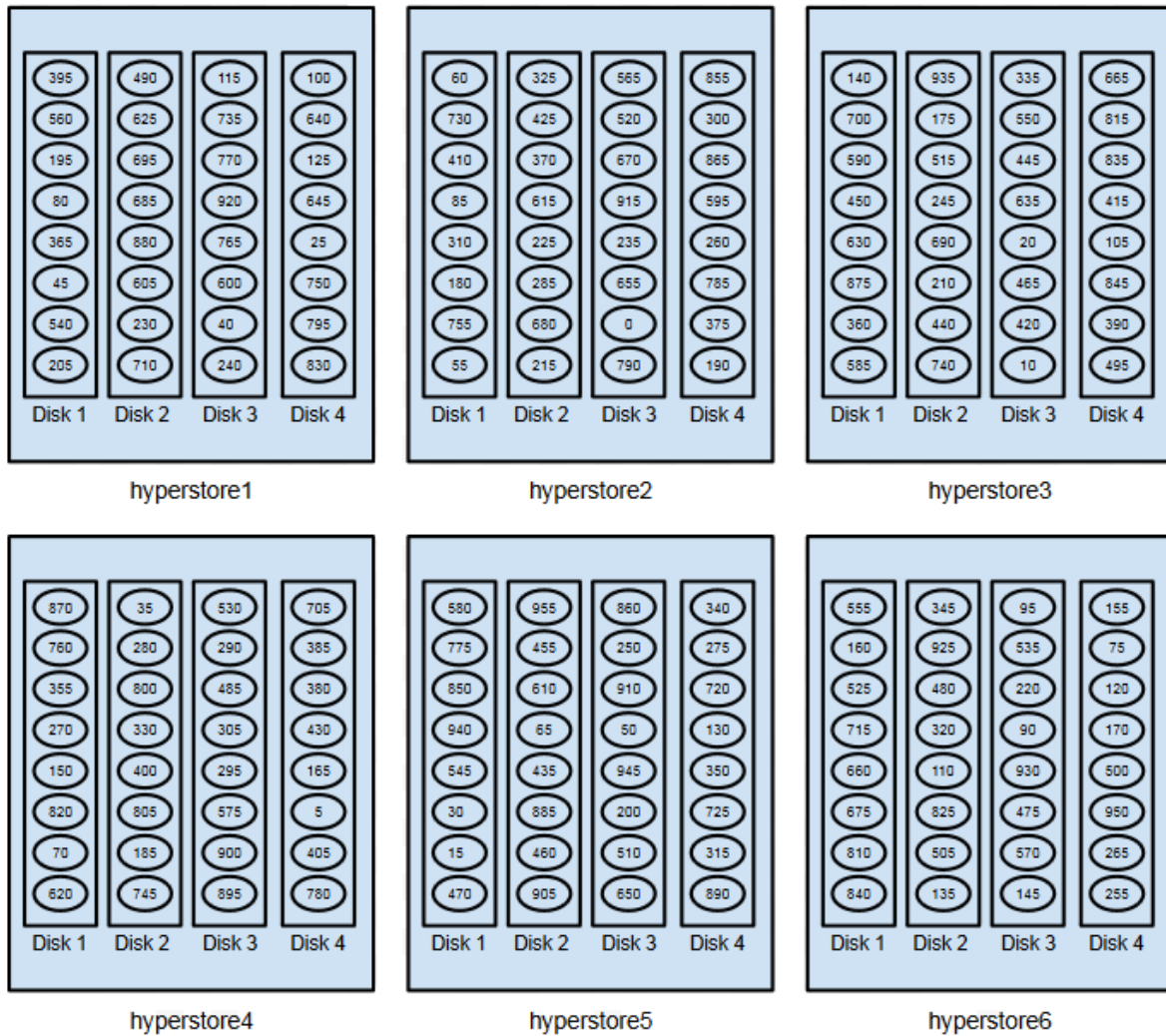
**Note** The data disk sizes in the table above are only for simple illustration of how the algorithm works. In calculations for actual hosts, the system uses not the disk raw size (the decimal-based size stated by the disk manufacturer) but rather the disk usable size after the disks are formatted and 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.

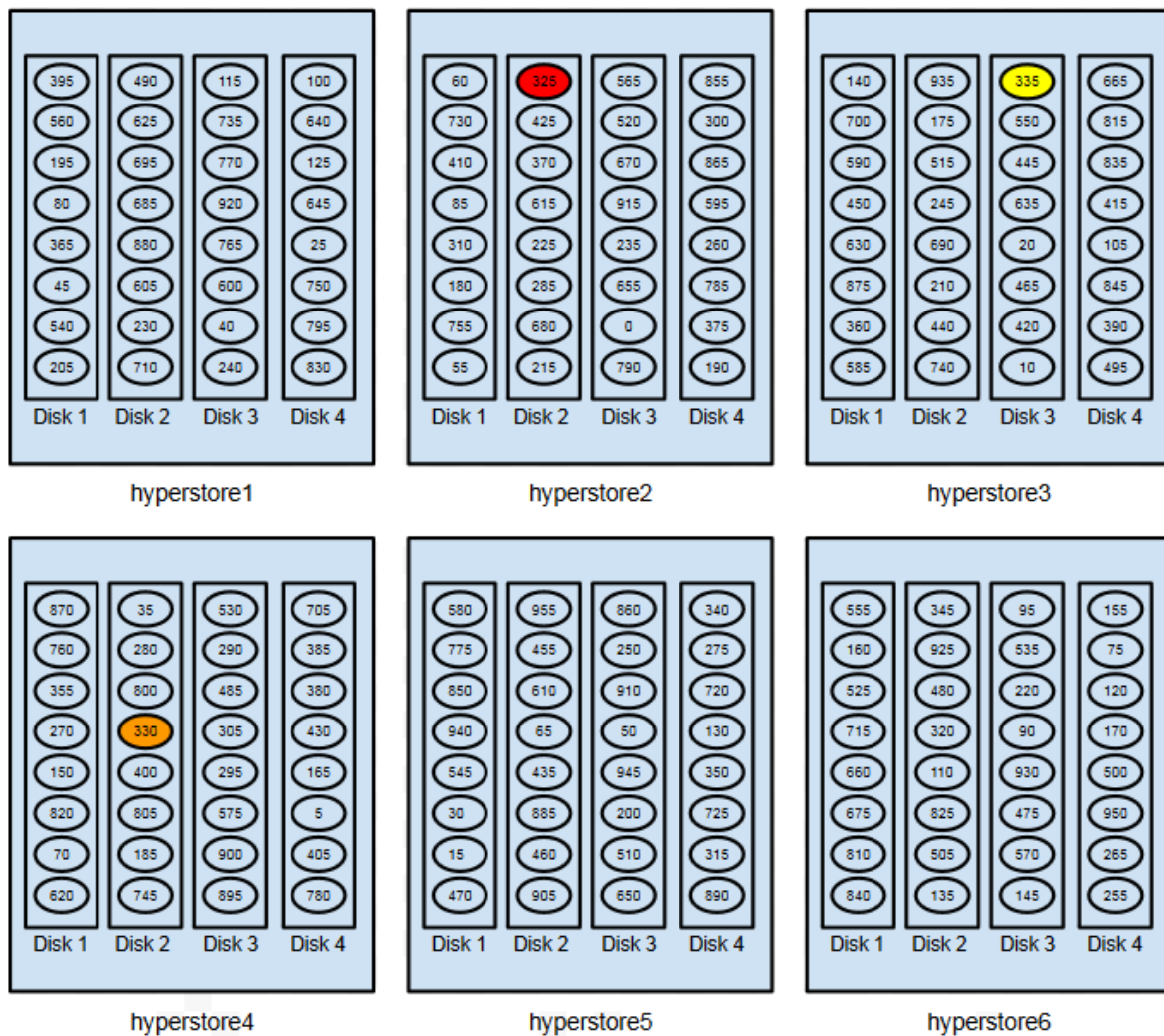
For illustration of how vNodes work to guide the distribution of data across a cluster, consider a cluster of six HyperStore hosts each of which has four disks designated for S3 object storage. Suppose that each physical host is assigned 32 tokens. And suppose for illustration that there is a simplified 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.



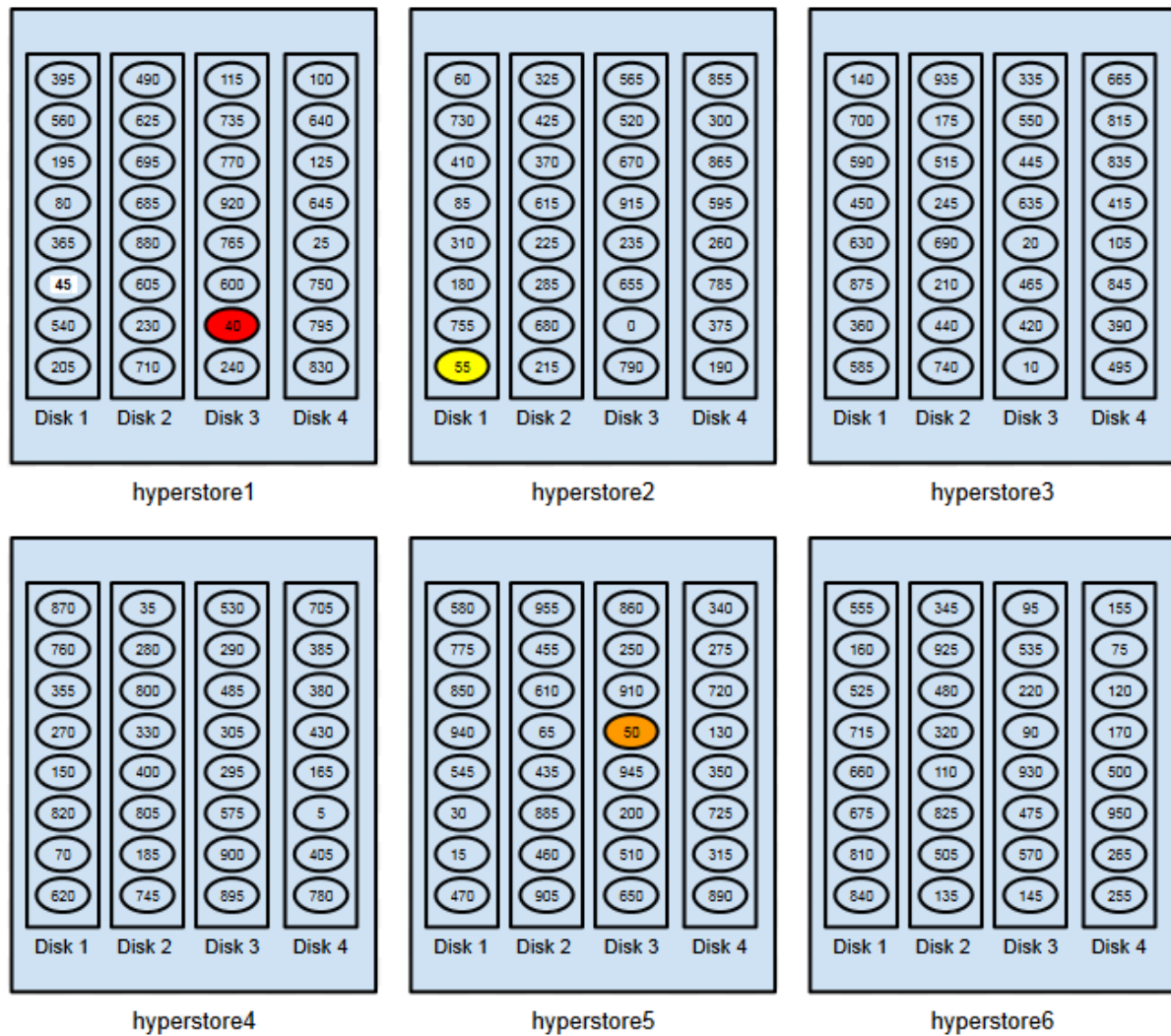
Now further suppose that you've configured your HyperStore system for 3X replication of S3 objects. And say that an S3 object is uploaded to the system and the hashing algorithm applied to the unique *<bucketname>/<objectname>* combination gives us a hash value of 322 (for this simplified example; in 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

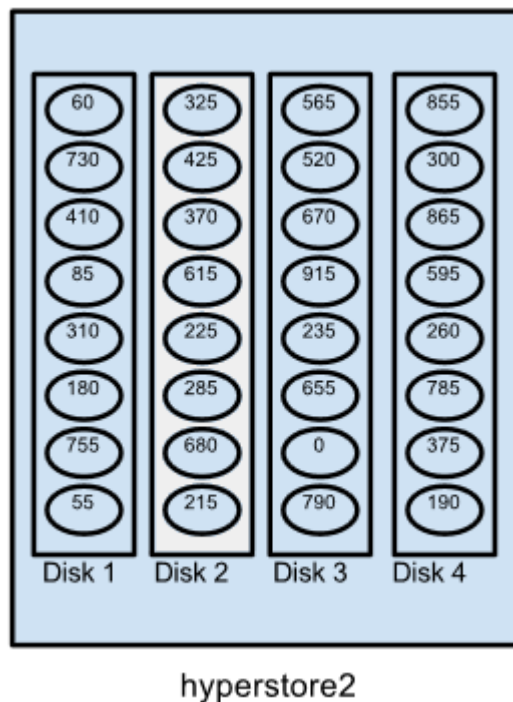
Now let's change perspective and see how things look for a particular disk from within the cluster. Recall that we've supposed a simplified token space with a total of 192 tokens (0, 5, 10, 15, and so on up to 955), randomly distributed across the cluster so that each host has 32 tokens and each host's 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

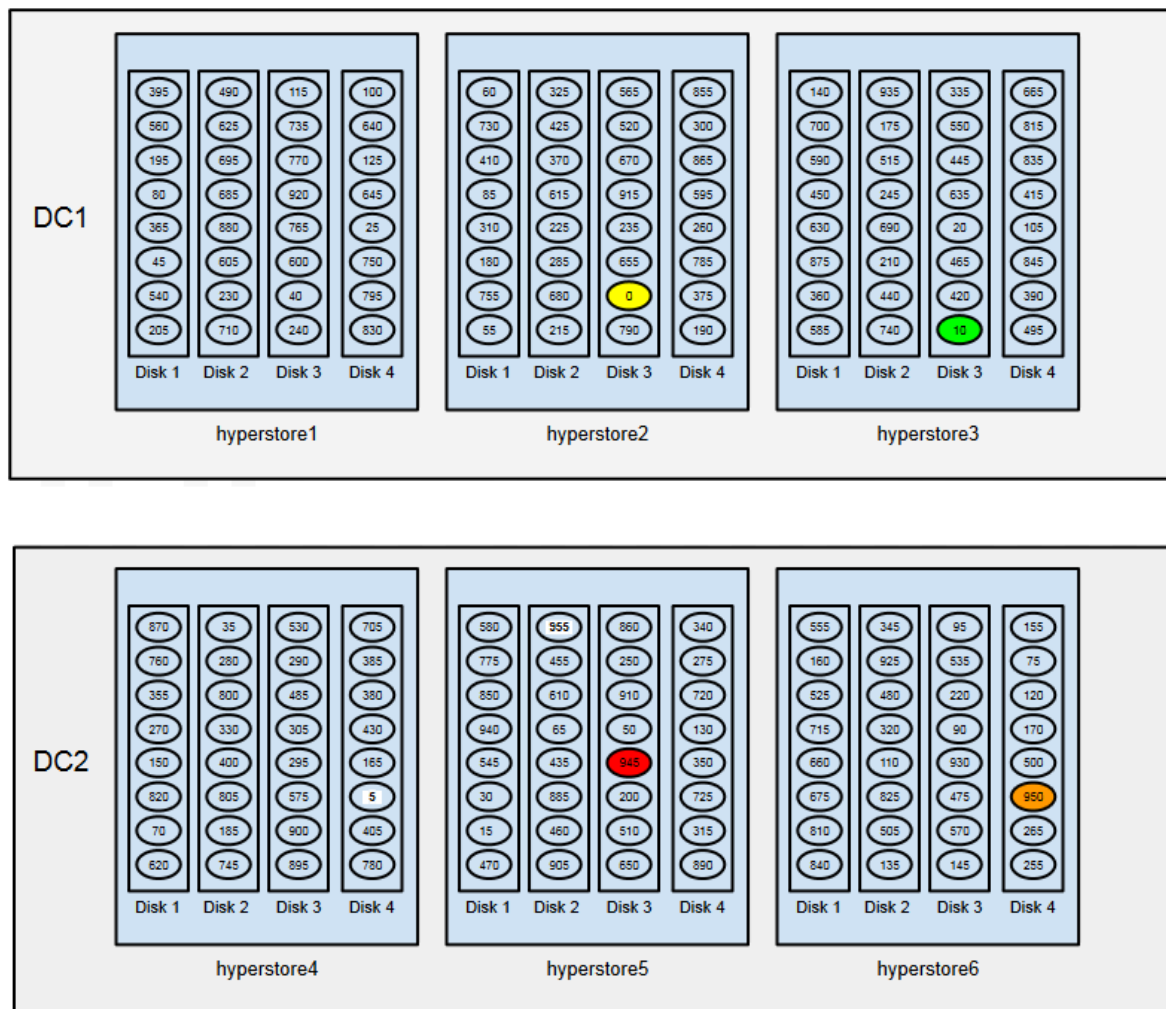
If you deploy your HyperStore cluster across multiple data centers within the same service region, your multiple data centers will be integrated in one unified token space.

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 `common.csv: hyperstore_data_directory`).

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.

This page left intentionally blank

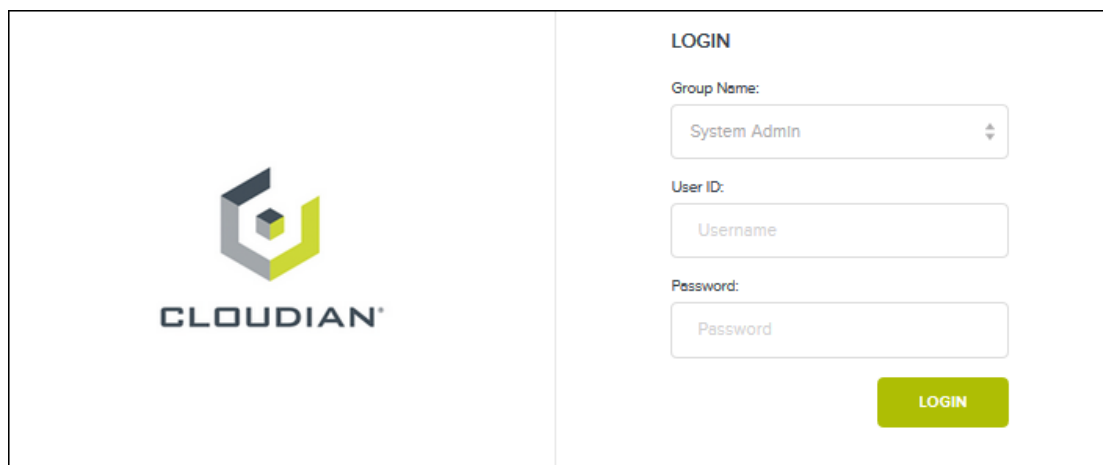
## Chapter 2. Getting Started with a New HyperStore System

Once a newly installed HyperStore system is up and running, you can use the Clouidian Management Console (CMC) to take the system for a test drive. Follow these steps:

1. Point a browser to `https://<CMC_host>:8443/Clouidian`

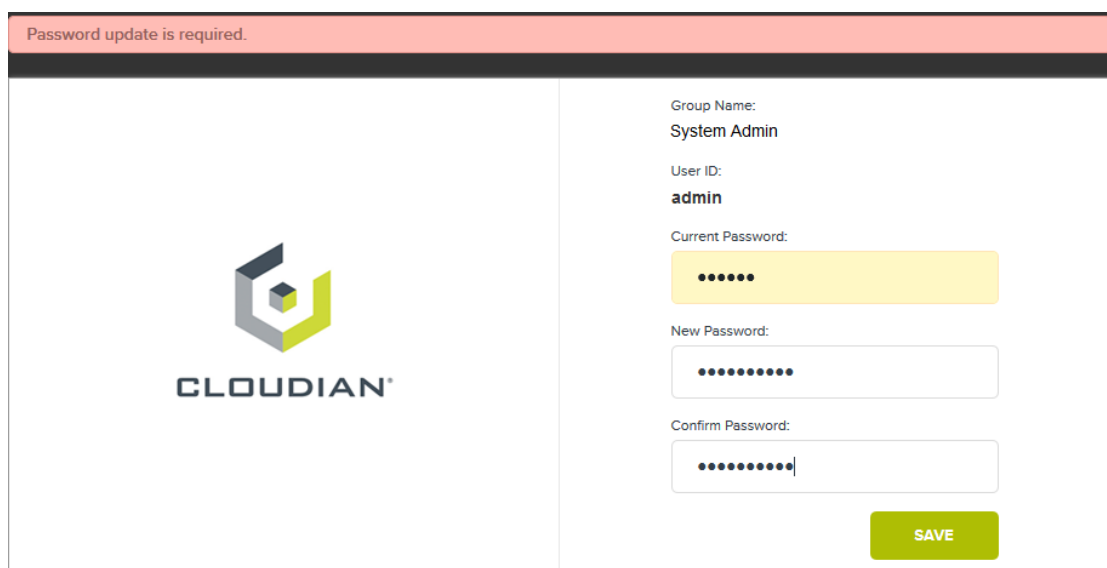
Since the CMC runs on all of your HyperStore nodes, for `<CMC_host>` you can use the fully qualified domain name (FQDN) or IP address of any node.

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.



The login screen features the Clouidian logo on the left. On the right, under the heading "LOGIN", there are three input fields: "Group Name" with a dropdown menu showing "System Admin", "User ID" with a text field containing "Username", and "Password" with a text field containing "Password". A green "LOGIN" button is positioned at the bottom right.

3. 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.



A red banner at the top states "Password update is required." The form on the right shows the "Group Name" as "System Admin" and "User ID" as "admin". It includes three password fields: "Current Password" (yellow background with dots), "New Password" (text field with dots), and "Confirm Password" (text field with dots). A green "SAVE" button is at the bottom right.

**Note** The first time you try to log into the CMC the system requires you to create a new password for the *admin* user. On subsequent logins to the CMC as the *admin* user, use the password that you created.

- Once you've logged into the CMC, select **Cluster** → **Storage Policies**. Then click **Create Storage Policy** to open the **Create New Policy** interface. Create a default storage policy for your system. A storage policy is a method of storing and protecting S3 object data and object metadata. Leave the "Group Visibility" unspecified so that this policy is visible to all groups. (At a later time you can edit this policy; create additional policies; and assign a different policy to be the system default policy if you wish).

STORAGE POLICIES

+ CREATE STORAGE POLICY

CREATE NEW POLICY

Policy Name

Policy Name

Policy Description

Policy Description

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
sfo-del-region1	DC1	1 of 3	
		2 of 3	disable
		3 of 3	

CONSISTENCY SETTING

CONSISTENCY LEVEL	READ	WRITE
ALL	<input type="checkbox"/>	<input type="checkbox"/>
QUORUM	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
ONE	<input type="checkbox"/>	

GROUP VISIBILITY

Please select a Group

ADD

Compression Type

NONE

Server-side Encryption

NONE

SAVE

CANCEL

For detail about the available options when you configure a new storage policy see **"Add a Storage Policy"** (page 353).

5. Create a regular S3 service user account that will enable you to test the system's S3 object storage services.

**Note** The system administrator role cannot have its own S3 storage service account.

- a. Select **Users & Groups** → **Manage Groups** → **New Group** to open the **Add New Group** interface. Create a new user group.

The screenshot shows the 'ADD NEW GROUP' form within the 'MANAGE GROUPS' section. At the top right, there are links for '+ GROUP QOS DEFAULT' and '+ NEW GROUP'. The form includes a checkbox for 'Active Group' which is checked. Fields include 'Group Name: \*' (text input), 'Group Description:' (text input), 'Rating Plan:' (dropdown menu with 'Please select a Rating Plan'), and two checkboxes: 'Enable S3 endpoints display filter' and 'Enable LDAP Authentication'. At the bottom right are 'CANCEL' and 'SAVE' buttons.

- b. Select **Users & Groups** → **Manage Users** → **New User** to open the **Add New User** interface. Create a new regular user, assigned to the user group that you created in the previous step. Make a note of the group name, user ID, and the password that you assign to the user so that you will be able to log in as that user.

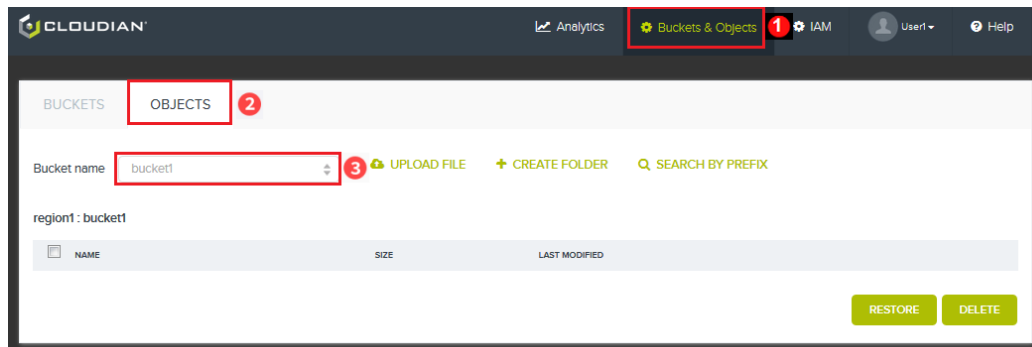
The screenshot shows the 'ADD NEW USER' form. At the top right is a link for '+ Active User'. The form includes fields for 'User ID: \*' (text input), 'User Type:' (dropdown menu with 'User' selected), 'Group Name: \*' (dropdown menu with 'Please select a Group'), 'Password: \*' (text input), and 'Confirm Password: \*' (text input). There is a 'More' link with a dropdown arrow. At the bottom right are 'CANCEL' and 'SAVE' buttons.

6. Log out of the CMC, and then log back in as the new user that you created. The CMC now displays only the functions that are available to regular service users, including the **Buckets & Objects** interface.

The screenshot shows the 'BUCKETS' section of the interface. At the top right is a link for '+ ADD NEW BUCKET'. The 'ADD NEW BUCKET' form includes fields for 'Bucket Name' (text input), 'Region' (dropdown menu with 'region1' selected), and 'Storage Policy' (dropdown menu with '\*HSFSPolicy\_region1' selected). Below these is a 'Storage Policy Description' field with the value 'HSFSPolicy\_region1'. At the bottom right are 'CANCEL' and 'CREATE' buttons.

7. Experiment with the CMC **Buckets & Objects** interface. For example:

- a. Add a new storage bucket by entering a bucket name and clicking **Create**. (You must create a bucket before you can upload any objects.)
- b. Use the **Objects** tab to create a folder in the bucket by entering a folder name and clicking **Create Folder**.



- c. Open the folder then upload a file into it by using the **Upload File** function.
- d. Verify that the uploaded file appears in the folder contents list, then verify that you can download the file by clicking on the file name.

# Chapter 3. Upgrading Your HyperStore Software Version

Subjects covered in this section:

- *Introduction (immediately below)*
- **"Preparing to Upgrade Your System"** (page 55)
- **"Upgrading Your System"** (page 58)
- **"Verifying Your System Upgrade"** (page 61)
- **"Installing a Patch"** (page 62)

The instructions that follow are for upgrading to HyperStore version 7.2.3 **from HyperStore version 7.1 or newer**. This upgrade procedure does not require S3 service interruption.

If you are already running HyperStore version 7.2.3 and are installing a patch release (such as 7.2.3.1 or 7.2.3.2), you can jump directly to **"Installing a Patch"** (page 62).

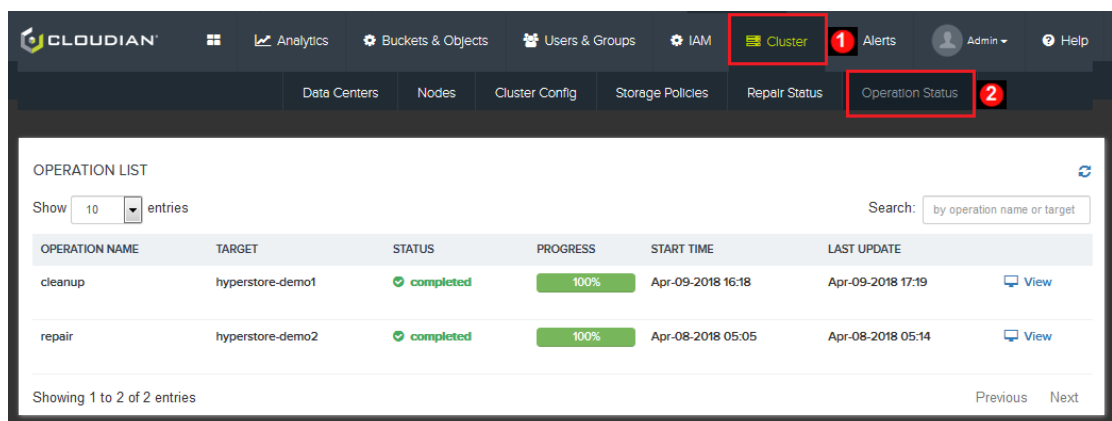
## IMPORTANT !

- \* To upgrade to HyperStore version 7.2.3, **all of your HyperStore host machines must have at least 128GB RAM**. If any of your existing HyperStore hosts have less than 128GB RAM, contact Clouidian Support.
- \* If you are currently on a HyperStore version older than 7.1, do not use the procedure described here. Instead contact Clouidian Support to request instructions for your particular upgrade path.
- \* If you are using Xen, Amazon EC2, or Logical Volume Manager (LVM), contact Clouidian Support before upgrading your system.

## 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.

- In the **Operation Status** page, check whether there are any operations currently in progress in any of your service regions.

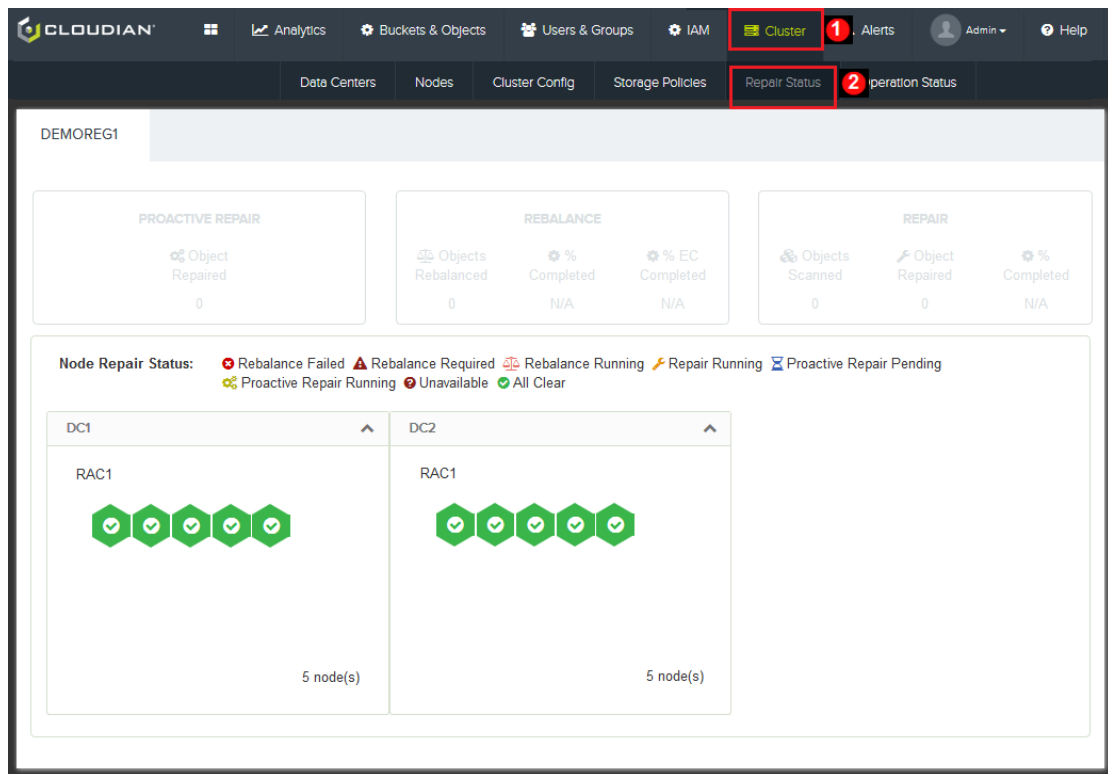


The screenshot shows the Clouidian CMC interface. The top navigation bar includes 'Cluster' (highlighted with a red box and a red '1') and 'Alerts'. Below this, the 'Operation Status' sub-tab is highlighted with a red box and a red '2'. The main content area displays the 'OPERATION LIST' with a table of operations.

OPERATION NAME	TARGET	STATUS	PROGRESS	START TIME	LAST UPDATE	
cleanup	hyperstore-demo1	completed	100%	Apr-09-2018 16:18	Apr-09-2018 17:19	<a href="#">View</a>
repair	hyperstore-demo2	completed	100%	Apr-08-2018 05:05	Apr-08-2018 05:14	<a href="#">View</a>

Showing 1 to 2 of 2 entries

- If any **repair** 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"** (page 686) and **"hsstool repairec"** (page 697) commands).
  - If any **cleanup**, **decommission**, or **rebalance** operation is in progress, Clouddian recommends that you wait until the operation completes before you perform the upgrade (or in the case of cleanup you can stop the cleanup by using the "stop" option that is supported by the **"hsstool cleanup"** (page 644) and **"hsstool cleanupec"** (page 651) commands).
- In the **Repair Status** page, **make sure there are no proactive repairs currently in progress** in any of your service regions. 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 by using the Maintenance command [proactiverepair](#) with the "stop" option selected. 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.

- In the **Data Centers** page, **make sure that 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..

**Node Status:** ● Unreachable ● Has Disk Error ● Disk Above 80% Full ● Has Alerts ● Under Maintenance ● All Clear

**DC1** **DC2**

**RAC1** **RAC1**

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	✓	✓	✓				✓

- If any **services are stopped**, start them. For instructions see **"Starting and Stopping Services"** (page 415).
- If any node is in **Maintenance Mode**, take it out of Maintenance Mode. For instructions see **"Stopping Maintenance Mode"** (page 330).
- If any node is in **"stop-write" condition**, contact Cloudian Support.

**Note** If the **Data Centers** page indicates that **any node has alerts**, go to the **"Node Status"** (page 313) 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 doing the upgrade:

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

```
CloudianInstallConfiguration.txt
```

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

### 3.1.2. Additional Upgrade Preparation If You Are Using Elasticsearch

If you have been using an Elasticsearch cluster together with HyperStore (for search or analysis of object metadata or logging data), then **before upgrading to HyperStore 7.2.3 you must upgrade your Elasticsearch cluster to version 6.6 or newer**. The Elasticsearch client included in HyperStore 7.2.3 is compatible only with Elasticsearch server version 6.6 or newer. To avoid integration errors between HyperStore and your Elasticsearch cluster, perform the Elasticsearch upgrade **before** the HyperStore upgrade.

## 3.2. Upgrading Your System

To perform the upgrade to HyperStore version 7.2.3:

1. Download the HyperStore product package (*CloudianHyperStore-7.2.3.bin* file) from the Cloudian Support portal into a working directory on your Puppet master node (such as */tmp* or your home directory). 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.
2. Copy your **current Cloudian license file** into the same working directory as the product package and signature file. Your current license file is located in the */opt/cloudian/conf* directory ends with suffix *.lic*. If there are multiple *.lic* files in this directory, use the most recent one. Copy this file to the working directory in which you've placed the new HyperStore product package.
3. In the working directory run the commands below to unpack the HyperStore package:

```
# chmod +x CloudianHyperStore-7.2.3.bin
# ./CloudianHyperStore-7.2.3.bin <license-file-name>
```

This creates a **new** installation staging directory named */opt/cloudian-staging/7.2.3*, 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 and your current Cloudian license file. Then from the shell, run these commands to unpack the HyperStore package (rather than the commands stated above):

```
$ chmod +x CloudianHyperStore-7.2.3.bin
$ hsrun --root CloudianHyperStore-7.2.3.bin <license-file-name>
```

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

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

```
# cd /opt/cloudian-staging/7.2.3
# ./cloudianInstall.sh
```

```
Cloudian HyperStore(R) 7.2 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: █
```

*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 Puppet 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.

5. From the installer main menu enter "3" for "Upgrade from <your version number> to 7.2.3". Then at the prompt, confirm that you wish to continue with the automated upgrade.

The upgrade script will first check your Puppet configuration template files (\*.erb files) from your existing HyperStore system to determine whether you made any customizations to those settings (changes from the default values):

- If you have **not** made any such changes, the upgrade proceeds.
- If you **have** made such changes, then in the new installation staging directory the installer creates a text file that lists those changes -- a "diff" file -- and prompts you to review the file, before the upgrade proceeds. Open a second terminal instance and in that terminal go to the new installation staging directory and view the "diff" file that the installer created (while the upgrade process remains paused in the first terminal instance). Then to carry forward your existing \*.erb file customizations that are identified in the diff file, in the second terminal instance manually make those same customizations to the new HyperStore version's \*.erb files (under /etc/cloudian-<new-version-#>-puppet/modules). For example, if in

your existing HyperStore system you had set a custom value for a *hyperstore-server.properties.erb* setting, edit that same setting in */etc/cloudian-7.2.3-puppet/modules/cloudians3/templates/hyperstore-server.properties.erb*. After saving your change -- and doing likewise for any of your other customizations to \*.erb settings -- return to the original terminal instance in which you are running the upgrade, and at the installer prompt continue with the upgrade. Note that you do not need to do a Puppet push, since the upgrade will apply your configuration edits.

*If you are using the HyperStore Shell to perform the upgrade*

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), you can view the diff file using *cat* and then use the following command to edit HyperStore \*.erb configuration files as needed:

```
$ hspkg config -e <filename>
```

In the background this invokes the Linux text editor *vi* to display and modify the configuration file.

**Note** Customizations that you may have made to the configuration file *common.csv* are handled differently. The installer detects such customizations and automatically applies the same customizations to the new version's *common.csv* file, without you having to do anything.

**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 progress. If you have a multi-region HyperStore system, the region that hosts the Redis Credentials master node will be upgraded last.

After the upgrade successfully completes, proceed to **"Verifying Your System Upgrade"** (page 61).

**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 Puppet 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/7.2.3*). HyperStore will continue to require certain files in this directory throughout the time that you are using this HyperStore version.

### 3.2.1. Upgrade Failure and Roll-Back

If the upgrade fails for certain nodes, those nodes are automatically rolled back to your previously existing HyperStore software version. The terminal will display basic information about the failure, and you can get more details from the *cloudian-installation.log* and the *cloudian-upgrade.log*, both of which are generated in the installation staging directory. Try to resolve the problems 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 also generates an *upgrade-logNconfig\*.tgz* "S.O.S" tar file (which packages together multiple upgrade-related files) that you can provide to Clouidian Support in the event that you need assistance in resolving any upgrade problems.

## 3.3. Verifying Your System Upgrade

After all HyperStore nodes have been upgraded, verify that all services are running and that the HyperStore version is now 7.2.3:

1. After the automated upgrade completes, you should be taken back to the main menu of the HyperStore installer. The first post-upgrade step is to confirm that all your HyperStore services are up and running:
  - a. From the installer's main menu select "Cluster Management".
  - b. From the Service Management sub-menu that displays select "Manage Services".
  - c. At the "Select a service to manage:" prompt, select All Services.
  - d. At the "Enter command" prompt, type *status*.

All services on all nodes should then indicate that they are running.

2. Next, confirm that the HyperStore software version is correct:
  - a. Still on the Service Management menu, at the "Select a service to manage:" prompt select the S3 Service.
  - b. At the "Enter command" prompt, type *version*.

On all nodes the S3 version should indicate version 7.2.3.

After confirming the version you can exit the installer.

3. Use the CMC to check on your upgraded cluster:
  - 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. (Although you disabled auto-repair prior to doing the upgrade, the system automatically 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.
4. If prior to the upgrade 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 and/or the color scheme, by again following the instructions for "**Rebranding the CMC UI**" (page 404).
5. If you have been using ElasticSearch for search of HyperStore object metadata, you should have upgraded your ES cluster to version 6.6 or new before upgrading HyperStore to version 7.2.3 (as noted in "**Preparing to Upgrade Your System**" (page 55)). Now, after having upgraded HyperStore, run this command from any HyperStore node to verify that a sync-up of the object metadata in your ES cluster against the object metadata in HyperStore can still be performed without error:

```
# /opt/cloudian/bin/elasticsearchSync all
```

*If you are using the HyperStore Shell to perform the upgrade*

If you are using the HyperStore Shell, you can run the ES sync tool as follows (with no path):

```
$ elasticsearchSync all
```

You are now done with upgrading to HyperStore 7.2.3.

**Note** 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 58)), 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*.

## 3.4. Installing a Patch

On occasion Cloudian may release a "patch" that enables customers to benefit from a recent HyperStore bug fix or fixes without having to wait for the next full release. A HyperStore patch release has a 4-digit release number, and can only be installed on systems running the preceding 3-digit release. For example:

Hypothetical Patch Release Number	Can Only Be Installed On Systems Running
7.1.7.1	7.1.7
7.1.7.2	7.1.7 (or a 7.1.7 system that's been patched to 7.1.7.1)
7.2.0.1	7.2.0 (7.2)
7.2.0.2	7.2.0 (or a 7.2.0 system that's been patched to 7.2.0.1)
7.2.1.1	7.2.1

It may happen that there are multiple patches in between full releases -- for example, if you are running 7.2.3 it may be that there is a 7.2.3.1 patch release and then later there is a 7.2.3.2 patch release. In this case you can install each patch when it comes out, or alternatively if you miss the first patch for some reason, you can install just the second patch and the second patch will include the fixes from the first patch.

Cloudian Support will announce patch releases when they come out, and you can download the patch from the Support portal. A patch is released as a self-extracting binary file, named as *S3Patch-<version>.bin* (for example *S3Patch-7.2.3.1.bin*). You can also download from the Support portal the signature file (*.sig* file) corresponding to the patch -- you will need the signature file if you are using the HyperStore Shell to apply the patch.

**Before installing a patch**, check to confirm that:

- All services are up in your system
- No repair, cleanup, or rebalance operations are currently running in your system.

For more information on performing these checks, see **"Preparing to Upgrade Your System"** (page 55).

**To install a patch:**

1. Place the patch binary file in a working directory on your Puppet master node (such as */tmp* or your home directory).
2. Change into that directory, and then run the following commands to run the patch file:

```
# chmod +x S3Patch-<version>.bin
# ./S3Patch-<version>.bin
```

*If you are using the HyperStore Shell to apply the patch*

If you are using the [HyperStore Shell \(HSH\)](#) to apply the patch, make sure you have in your working directory the signature file (*.sig* file) corresponding to the patch file. Then from the shell, run these

commands (rather than the commands stated above):

```
$ chmod +x S3Patch-<version>.bin
$ hsrun --root S3Patch-<version>.bin
```

When you run the patch file, you will be prompted to confirm that you want to install the patch -- enter **y** to do so. Then it automatically takes all the actions necessary to apply the patch to each of your HyperStore nodes. Specifically, the following actions are automatically executed:

- The *S3Patch-<version>.bin* file content -- including a patch installation script -- is extracted into a */s3patch/<patch-version>/* sub-directory under your HyperStore system's current installation staging directory
- The patch installation script is automatically launched. The script performs a non-disruptive, rolling install of the patch to each of your HyperStore nodes one at a time -- including automatically restarting the affected services on one node at a time.
- The status of the patch installation process is written to the console, and log messages pertaining to the patch installation are written to *<current-staging-directory>/installs3patch.log*. Also, a backup copy of the original, unpatched version of the main *.jar* file (Java archive file) from your existing HyperStore version is written to *<current-staging-directory>/s3patch/backup/*.

After a successful patch upgrade of all nodes, you can launch the main HyperStore installer (*./cloud-ianInstall.sh* in your current staging directory), go to the Manage Services menu, and for the S3 Service check the version. The results should show that on all nodes, the S3 Service now has the version number of the patch that you installed. You should also log into the CMC and check some of the main status reporting pages -- such as the [Data Centers](#) page and the [Alerts](#) page -- to confirm that your patched HyperStore system is healthy. You may also want to exercise the system by, for instance, uploading some objects into a bucket.

**Note** Do not delete the *S3Patch-<version>.bin* file from the working directory in which you placed it. You may need to use the file again, as described in the sections below.

### 3.4.1. Reapplying the Patch in the Case of Installation Errors

In some cases the patch installation script may write messages to the console indicating that the patch did not successfully install on a certain node or nodes. An example is if a node is down at the time that the patch installation script runs -- the script will not be able to install the patch to that node. In this scenario, first correct the underlying condition -- for example, bring a down node back up. Then on the Puppet master node change into the working directory where the *S3Patch-<version>.bin* file is located and run the file again:

```
# ./S3Patch-<version>.bin
```

You will again be prompted to confirm that you want to install the patch -- enter **y** to do so. The patch script will then check each HyperStore node and install the patch only on nodes on which it has not already been successfully installed. The status will be written to the console, and also logged to *<current-staging-directory>/installs3patch.log*.

### 3.4.2. Reverting a Patch

In the unlikely event that you need to revert a patch, the patch binary file supports doing so. You might need to revert a patch if, for example:

- You are unable to successfully install the patch to all nodes -- that is, you are in a condition where some nodes were successfully patched while errors prevented patching of the other nodes, and you are unable to correct the errors.
- You successfully patch all nodes, but subsequently you encounter negative behavior in your system that you had not encountered prior to the installation of the patch.

If you are reverting a patch, **contact Cloudian Support** (either before or after reverting the patch).

To revert a patch:

1. On the Puppet master node, change into the working directory in which the *S3Patch-<version>.bin* file is located.
2. Run the patch bin file using the **-r** option:

```
# ./S3Patch-<version>.bin -r
```

You will be prompted to confirm that you want to revert the patch -- enter **y** to do so. The patch script will then revert your HyperStore system to the preceding **3-digit release version**. For example, reverting a 7.2.3.1 patch will revert your system to 7.2.3; and reverting a 7.2.3.2 patch will also revert your system to 7.2.3 (not to 7.2.3.1).

### 3.4.3. Adding Nodes to a Patched System

If you patch your system and then you [add nodes](#) (or a [new data center](#) or [new region](#)) to your system before you have upgraded to a later full release version, you will need to run the patch installer again to patch the newly added nodes. For example, if you are running a 7.2.3 system, and you install a 7.2.3.1 patch, and then you add more nodes to your system, you will need to run the 7.2.3.1 patch installer to patch the new nodes. Subsequently, when you've upgraded your system to a later 3-digit or 2-digit release (a full release rather than a patch release), there is no longer any need to run the 7.2.3.1 patch installer if you add more nodes.

To patch nodes that you've added to your system:

1. On the Puppet master node, change into the working directory in which the *S3Patch-<version>.bin* file is located.
2. Run the patch bin file:

```
# ./S3Patch-<version>.bin
```

You will be prompted to confirm that you want to install the patch -- enter **y** to do so. The patch script will then check each HyperStore node and install the patch only on nodes on which it has not already been successfully installed. The status will be written to the console, and also logged to *<current-staging-directory>/installs3patch.log*.



# Chapter 4. Working with HyperStore Major Features

## 4.1. Management Interfaces and Tools

### 4.1.1. HyperStore Management Interfaces and Tools -- Feature Overview

Your HyperStore system includes several interfaces and tools through which you can monitor, manage, and use the system:

Interface or Tool	Purpose	More Information
Cloudian Management Console (CMC)	The CMC is the GUI for the HyperStore system. It supports monitoring, managing, and expanding the system, and also includes an S3 client interface.	See <b>"Cloudian Management Console (CMC)"</b> (page 196) and the rest of Section 5. Also, while using the CMC you can click the Help button in the upper right of the interface to view information about the particular CMC page that you're on.
Installer	You can use the HyperStore installer ( <i>cloudianInstall.sh</i> ) not only to install the system but also to perform certain types of system customizations, to push configuration file edits out to the system, and to stop or restart services.	<ul style="list-style-type: none"> <li>• <b>"Installer Advanced Configuration Options"</b> (page 501)</li> <li>• <b>"Pushing Configuration File Edits to the Cluster and Restarting Services"</b> (page 506)</li> <li>• <b>"Starting and Stopping Services"</b> (page 415)</li> </ul>
hsstool	With the <i>hsstool</i> utility you can retrieve information about, and perform management operations on, the object data and metadata that is stored in your HyperStore system.	<b>"hsstool "</b> (page 643)
HyperStore Shell (HSH)	The HyperStore Shell 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.	<b>"Security Features"</b> (page 89)
Configuration files	While you can perform most common system configuration tasks by using the CMC or the installer, many additional configuration options are available to you in configuration files that you can edit.	<b>"HyperStore Configuration Files"</b> (page 511)

Interface or Tool	Purpose	More Information
Admin API	The HyperStore Admin API is a RESTful HTTP API through which you can provision users and groups, manage rating plans and quality of service (QoS) controls, retrieve monitoring data, and perform other administrative tasks.	" <b>HyperStore Admin API Introduction</b> " (page 741) and the rest of Section 12.

In addition to the above-listed tools that are included as part of HyperStore, other helpful tools are:

- **HyperIQ** -- HyperIQ is a separate Cloudian product that provides advanced real-time system monitoring and analytics for a HyperStore system. Contact your Cloudian representative about obtaining HyperIQ.
- **Cloudian Support tools** -- The Cloudian Support team has a variety of special-purpose tools that they may use, or provide for you to use, if they are working with you to troubleshoot or rectify system issues.

## 4.2. Support for AWS APIs

### 4.2.1. Support for AWS APIs -- Feature Overview

HyperStore supports several Amazon Web Services (AWS) APIs -- most importantly the S3 API, but also some related APIs:

API	API Purpose and Degree of HyperStore Coverage	More Information
Simple Storage Service (S3)	Create and manage storage buckets. Write, read, and manage objects in those buckets.  HyperStore provides <b>comprehensive</b> support for the AWS S3 API. HyperStore supports most AWS S3 API actions, and typically the unsupported actions are specific to AWS services and not applicable outside of AWS.	" <b>HyperStore Support for the AWS S3 API</b> " (page 931) and the rest of Section 13
Identity and Access Management service (IAM)	Under HyperStore regular user accounts, create and manage IAM groups and IAM users who have S3 permissions restricted by configurable IAM policies.  HyperStore provides <b>limited</b> support for the AWS IAM API, supporting those IAM actions that are most relevant to HyperStore's S3-compatible object storage service.	" <b>HyperStore Support for the AWS IAM API</b> " (page 991) and the rest of Section 14
Security Token Service (STS)	Request temporary, limited-privilege credentials for IAM users or federated users authenticated by SAML.  HyperStore provides <b>limited</b> support for the AWS STS API, as a means of allowing SAML-based authentication of users.	" <b>HyperStore Support for the AWS STS API</b> " (page 1037) and the rest of Section 15
Simple Queue Service (SQS)	Create, populate, and manage message queues.	" <b>HyperStore Support for the AWS SQS API</b> " (page 1041)

API	API Purpose and Degree of HyperStore Coverage	More Information
	HyperStore provides <b>limited</b> support for the AWS SQS API, as a means of implementing the S3 bucket notification feature.	and the rest of Section 16

#### 4.2.1.1. HyperStore Service Endpoints for AWS APIs

API	Default Service Endpoint(s)
Simple Storage Service (S3)	<code>s3-&lt;regionname&gt;.&lt;your-domain&gt;:80</code> (HTTP) <code>s3-&lt;regionname&gt;.&lt;your-domain&gt;:443</code> (HTTPS, <a href="#">disabled by default</a> )
Identity and Access Management service (IAM)	<code>iam.&lt;your-domain&gt;:16080</code> (HTTP) <code>iam.&lt;your-domain&gt;:16443</code> (HTTPS)
Security Token Service (STS)	<code>sts.&lt;your-domain&gt;:16080</code> (HTTP) <code>sts.&lt;your-domain&gt;:16443</code> (HTTPS) <div><b>Note</b> STS requests are serviced through the IAM listening ports.</div>
Simple Queue Service (SQS)	<code>s3-sqs.&lt;your-domain&gt;:18090</code> (HTTP) <div><b>Note</b> HTTPS is not currently supported for the SQS Service.</div>

## 4.3. Nodes, Data Centers, and Regions

### 4.3.1. Nodes, Data Centers, and Regions Feature Overview

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Storage Policies in Multi-DC or Multi-Region Systems"** (page 69)
- **"Service Architecture in Multi-DC or Multi-Region Systems"** (page 69)
- **"Deploying HyperStore to Multiple DCs or Regions"** (page 70)

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** -- Also known as a **cluster**, a 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 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 71).
- **Multiple data centers in a single region.** With this topology, HyperStore nodes running in multiple data centers comprise one unified storage cluster. 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 endpoint (to which S3 clients submit requests), its own independent storage cluster, 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 affords a substantial degree of 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 30).

#### 4.3.1.1. 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. Users when they create a bucket 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 cluster 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 76).

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 186).

#### 4.3.1.2. Service Architecture in Multi-DC or Multi-Region Systems

##### 4.3.1.2.1. Multi-DC, Single Region System

Along with common services that run on every node in a cluster (such as the S3 Service, the HyperStore Service, and Cassandra), the HyperStore system 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 Redis Credentials slave nodes and each DC will have one node acting as a Redis QoS slave node.

For a summary of services and how they are allocated, see "**HyperStore Services**" (page 21).

For a diagram showing typical services distribution in a multi-DC region, see "**Services Distribution -- Multi-DC, Single Region**" (page 32).

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 598).

#### 4.3.1.2.2. Multi-Region System

A multi-region HyperStore system has the following characteristics:

- One of the regions serves 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 cluster 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 Redis Credentials database serves the entire multi-region system. There is just one Redis Credentials master node for the whole system, and that node is located in your default region. Within each region, there are two Redis Credentials 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 Redis QoS database. Each regional Redis QoS database has its own master node. In each region there is also one Redis QoS slave node per data center.
- The Redis Monitor application will monitor all the Redis databases in all the regions (and if necessary trigger failover of the Redis 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 Cassandra database in the default region. HyperStore services in non-default regions access Cassandra in the default region to retrieve this group and user information as needed.
- Just one Puppet master 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 33).

#### 4.3.1.3. Deploying HyperStore to Multiple DCs or Regions

A common 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 430)
- **"Adding a Region"** (page 437)

HyperStore also supports the option of installing HyperStore software to multiple DCs or regions from the outset, upon the initial system installation. From a HyperStore system configuration perspective the key here is the "survey file", which the HyperStore *system\_setup.sh* tool helps you to create. By responding to that tool's interactive prompts, you create a survey file that identifies (among other things) the name of the data center and region that each node resides in. Subsequently the installer tool (*cloudianInstall.sh*) installs HyperStore software to each of those nodes and configures the system to be a single-DC, multi-DC, or multi-DC / multi-region system, in accordance with your survey file.

Below is an example of an installation survey file for a 3-node HyperStore system that will be configured as just a single service region with just a single data center. Note that you must provide a data center name and a region name even if you will have just one DC constituting just one region. Here the region name is "tokyo" and the data center name is "DC1".

```
tokyo,cloudian-vm7,66.10.1.33,DC1,RAC1
tokyo,cloudian-vm8,66.10.1.34,DC1,RAC1
tokyo,cloudian-vm9,66.10.1.35,DC1,RAC1
```

Here is a second example, this time for a system that will be installed as a single-region system with two data centers:

```
tokyo,cloudian1,66.1.1.11,DC1,RAC1
tokyo,cloudian2,66.1.1.12,DC1,RAC1
tokyo,cloudian3,66.1.1.13,DC1,RAC1
tokyo,cloudian4,67.2.2.17,DC2,RAC1
tokyo,cloudian5,67.2.2.18,DC2,RAC1
tokyo,cloudian6,67.2.2.19,DC2,RAC1
```

Below is a third example, this time for a system that will be installed as a two-region system. Note that in this example, the "tokyo" region encompasses two data centers while the "osaka" region consists of just one data center.

```
tokyo,cloudian1,66.1.1.11,DC1,RAC1
tokyo,cloudian2,66.1.1.12,DC1,RAC1
tokyo,cloudian3,66.1.1.13,DC1,RAC1
tokyo,cloudian4,67.2.2.17,DC2,RAC1
tokyo,cloudian5,67.2.2.18,DC2,RAC1
tokyo,cloudian6,67.2.2.19,DC2,RAC1
osaka,cloudian7,68.10.3.24,DC3,RAC1
osaka,cloudian8,68.10.3.25,DC3,RAC1
osaka,cloudian9,68.10.3.26,DC3,RAC1
```

For more information about installation including node preparation, DNS, and load balancer requirements, see the HyperStore Installation Guide.

### 4.3.2. Capacity Monitoring and Expansion

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Monitoring Cluster Storage Capacity Utilization"** (page 72)
- **"Adding Nodes to Your System"** (page 74)

**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 up to 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 72), further below). As best practices for cluster expansion timing, Cloudbian 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:

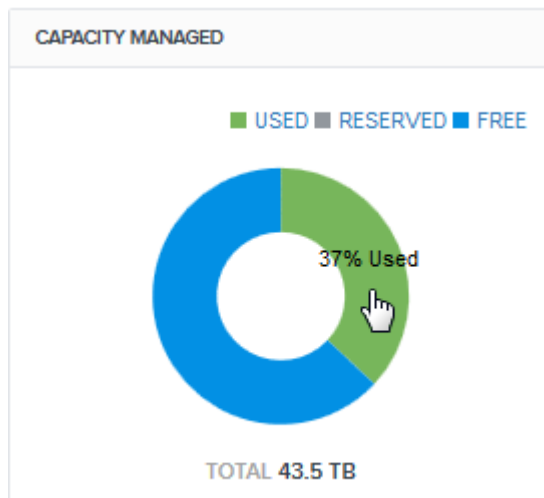
- The minimum unit of expansion is a node. HyperStore does not support adding disks to existing nodes.
  - You need to allow time to acquire host machines and prepare them for being added to your cluster.
  - Preferably, cluster expansions should be substantial enough that the expanded cluster will allow 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.
  - Cloudbian 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.

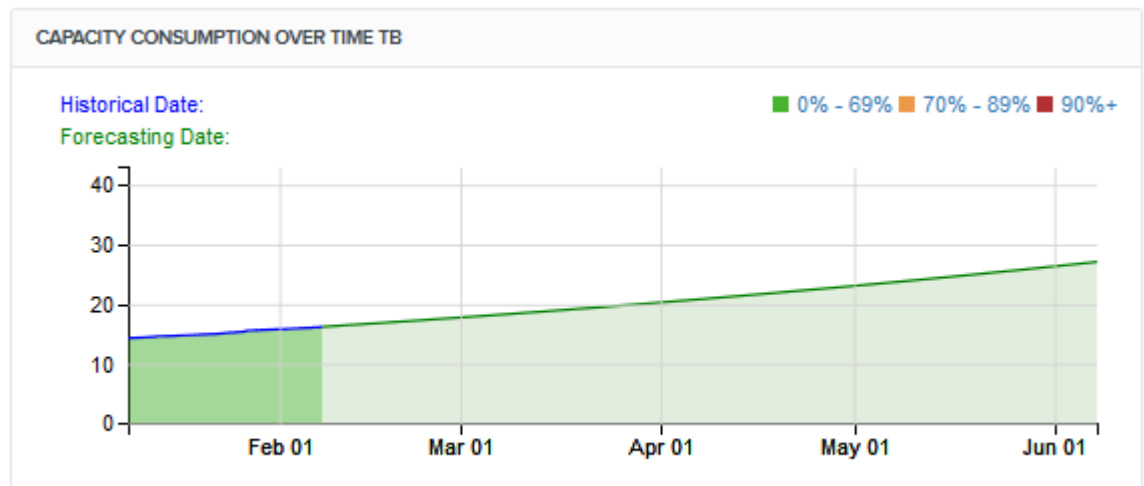
#### 4.3.2.1. Monitoring Cluster Storage Capacity Utilization

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](#) 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 view each node's storage capacity utilization as well as the utilization level for each disk on each node.

#### 4.3.2.2. Adding Nodes to Your System

To add nodes to an existing HyperStore data center, follow the documented procedure **"Adding Nodes"** (page 420).

To add a new data center (with new nodes) to an existing HyperStore service region, follow the documented procedure **"Adding a Data Center"** (page 430).

To add a new service region (with new nodes) to an existing HyperStore system, follow the documented procedure **"Adding a Region"** (page 437).

**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).

#### 4.3.3. Using the CMC with Multiple DCs or Regions

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

##### 4.3.3.1. Multi-DC, Single Region System

In the CMC, in most respects a multi-DC deployment within a service region is presented as one integrated cluster. For example, performance statistics such as S3 transactions per second and S3 bytes throughput per second are reported for the cluster (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 cluster on a per-DC basis.
- The [Object Locator](#) feature lets you see exactly where all of a specified object's replicas or erasure coded fragments are located (on which nodes, in which DC)

##### 4.3.3.2. 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 263), 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 271): 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) Controls"** (page 285), 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 218), they choose which region to create the bucket in.
- Usage Reporting and Billing
  - When you **"Create a Usage Report"** (page 211) 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 420) to the system, you specify the region and data center in which to add it.

#### 4.3.3.2.1. 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=xxx&groupId=xxx[&region=xxx] 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"** (page 741).

## 4.4. Storage Policies

### 4.4.1. Storage Policies Feature Overview

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Supported Erasure Coding Configurations for a Single DC"** (page 77)
- **"Multi- Data Center Storage Policies"** (page 77)

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. Users when they create a new storage bucket can then 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 and read 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, 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.**

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 77)).

Also as part of the configuration options for each storage policy, you can choose whether to compress and/or encrypt stored objects.

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.

#### 4.4.1.1. 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 Clouddian Support or your Clouddian 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 84).

#### 4.4.1.2. 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 D2 (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](#).

## 4.4.2. 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 endpoint 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.

- **"Consistency Level "ALL""** (page 369)
- **"Consistency Level "QUORUM""** (page 374)
- **"Consistency Level "EACH QUORUM""** (page 371)
- **"Consistency Level "LOCAL QUORUM""** (page 372)
- **"Consistency Level "ANY QUORUM""** (page 370)
- **"Consistency Level "ONE""** (page 373)

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 84).

**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 39).

### 4.4.2.1. 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 Cassandra) 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 Cassandra, and all the object's data replicas' file digests in RocksDB -- and then it retrieves one object data replica and returns it to the client.

#### 4.4.2.2. Note About Bucket Content List Reads

In the documentation of the supported consistency levels such as "ALL", "QUORUM", and so on (see the cross references above), when read consistency requirements are discussed the focus is on reads of individual objects -- that is, the consistency requirements for successfully implementing S3 *GET Object* requests. It's worth noting however that your configured read consistency requirements also apply to bucket content list reads -- that is, implementing S3 *GET Bucket (List Objects)* requests.

Metadata for objects is stored in two different types of record in Cassandra: 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 *GET Object* request requires reading the object's object-level metadata record and a *GET Bucket (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 buckets content lists. So for example if you use a QUORUM read consistency requirement, then in order to successfully execute a *GET Bucket (List Objects)* request the system must be able to read a QUORUM of the bucket-level object metadata records for the bucket.

For more on the meaning of QUORUM and the other supported consistency levels, see the cross references above.

#### 4.4.3. Object Metadata Replication

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Object Metadata in Replication Storage Policies"** (page 80)
- **"Object Metadata in Erasure Coding Storage Policies"** (page 81)
- **"Two Types of Object Metadata Record"** (page 81)

HyperStore [object metadata](#) is stored in Cassandra, and is protected by replication. The degree to which object metadata is replicated depends on the type of storage policy being used.

##### 4.4.3.1. 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 Cassandra). 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.

#### 4.4.3.2. 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 small in size, and therefore not appropriate for erasure coding.

Specifically, for erasure coding storage policies the system will store  **$2m-1$  replicas of object metadata**. For example, with a 4+2 erasure coding storage policy, the object metadata is protected by 3X replication.

Additional examples of  $2m-1$  object metadata replication:

- Single-DC,  $k+m = 6+2 \rightarrow 3$  replicas of object metadata
- Single-DC,  $k+m = 9+3 \rightarrow 5$  replicas of object metadata
- Replicated multi-DC,  $k+m = 4+2 \rightarrow 3$  replicas of object metadata in each DC
- Distributed multi-DC,  $k+m = 5+4 \rightarrow 7$  replicas of object metadata distributed across the DCs

#### 4.4.3.3. Two Types of Object Metadata Record

Metadata for an object is stored in two different types of record in Cassandra: 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 Structure in Cassandra**" (page 167).

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 Cassandra, both types of object metadata record are part of the *UserData\_<policyid>* keyspaces.

#### 4.4.4. System Metadata Replication

HyperStore stores system metadata in several Cassandra keyspaces: the *AccountInfo* keyspace (for user account information), the *Reports* keyspace (for usage reporting data), and the *Monitoring* keyspace (for system monitoring data). The initial replication level for this system metadata is set by the operator during HyperStore system installation (with a default of 3 replicas per service region). Subsequently, the system automatically adjusts the system metadata replication level in response to your creation of storage policies. The system uses whichever of these criteria yields the **highest** replication level:

- The system metadata replication level set during installation
- Matching the replication factor of any replication storage policies created in the system

- Having  $2m-1$  replicas in each DC for any regular erasure coding or replicated erasure coding policies created in the system
- Having  $2m+1$  replicas distributed across DCs for any distributed erasure coding policies created in the system

The table below shows examples of what the automatic system metadata replication level would be in different system configuration scenarios.

System Configuration	System Metadata Replication Level	Comment
<ul style="list-style-type: none"> <li>• Single data center</li> <li>• System metadata replication level configured during install = 2</li> <li>• Just one storage policy created in system: a 3X replication policy</li> </ul>	3	The highest replication level is yielded by matching the level of the 3X replication storage policy
<ul style="list-style-type: none"> <li>• Single data center</li> <li>• System metadata replication level configured during install = 5</li> <li>• Just one storage policy created in system: a 3X replication policy</li> </ul>	5	The highest replication level is yielded by using the level configured by the operator during system installation
<ul style="list-style-type: none"> <li>• Single data center</li> <li>• System metadata replication level configured during install = 3 (the default)</li> <li>• Two storage policies created in system: a 3X replication policy and a 9+3 erasure coding policy</li> </ul>	5	The highest replication level is yielded by using $2m-1$ from the 9+3 erasure coding policy
<ul style="list-style-type: none"> <li>• Two data centers in a single service region</li> <li>• System metadata replication level configured during install = 3 (the default; implemented as 1 replica in one DC and 2 in the other)</li> <li>• One storage policy created in system: a replicated 4+2 erasure coding policy</li> </ul>	3 in each DC	The highest replication level is yielded by using $2m-1$ per DC from the replicated 4+2 erasure coding policy
<ul style="list-style-type: none"> <li>• Three data centers in a single service region</li> <li>• System metadata replication level configured during install = 3 (the default; implemented as 1 in each DC)</li> <li>• Two storage policies created in system: a replication policy at 2X per DC; and a 5+4 distributed erasure coding policy</li> </ul>	3 in each DC	The highest replication level is yielded by using $2m+1$ (distributed across DCs) from the distributed 5+4 erasure coding policy

The general logic behind this automated adjustment of system metadata replication level is that the **greater the resilience of your configured storage policies** -- in terms of ability to read and write object data and object metadata when a node or nodes are unavailable -- **the greater will be the resilience built into the system metadata storage configuration**.

#### 4.4.4.1. Consistency Requirements for System Metadata Reads and Writes

Consistency requirements for object data and object metadata reads and writes are set per storage policy, when you create storage policies. By contrast consistency requirements for reads and writes of system metadata are set at the system level, by these configuration properties in [mts.properties.erb](#):

- **"cloudian.cassandra.default.ConsistencyLevel.Read"** (page 558) (default = LOCAL\_QUORUM,ONE)
- **"cloudian.cassandra.default.ConsistencyLevel.Write"** (page 559) (default = QUORUM,LOCAL\_QUORUM)

#### 4.4.5. 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 353)
- **"Edit a Storage Policy"** (page 377)
- **"Designate a Default Storage Policy"** (page 377)
- **"Disable a Storage Policy"** (page 378)
- **"Delete a Storage Policy"** (page 379)

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** The system supports a configurable maximum number of storage policies (*mts.properties*: **"cloudian.protection.policy.max"** (page 576), default = 25). After you have created this many storage policies, you cannot create additional new policies until you either delete unused policies or increase the configurable maximum.

##### 4.4.5.0.1. Retrieving Storage Policy Usage Through the Admin API

You cannot create or modify storage policies through the Admin API. However, you can use the API to retrieve a list of buckets that use each storage policy, with the [GET /bppolicy/bucketsperpolicy](#) method.

#### 4.4.6. 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:

- **"Add a Bucket"** (page 218)

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:

- [PUT Bucket](#)

### 4.4.7. 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.

#### Finding an Object's Replicas or EC Fragments (CMC)

To find an object's replicas or fragments locations using the CMC:

- [Object Locator](#)

#### 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](#)

### 4.4.8. 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](#).

#### 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	Writes succeed for some objects and fail for others.	Writes succeed for some objects and fail for others.
		2	All writes fail	All writes fail	Writes succeed for some objects and fail for others
	QUORUM (default)	1	All writes succeed	All writes succeed	All writes succeed
		2	All writes fail	Writes succeed for some objects and fail for others	Writes succeed for some objects and fail for others
	ONE	1 or 2	All writes succeed	All writes succeed	All writes succeed
		3 or more	All writes fail	All writes fail	All writes fail
Reads	ALL	1	All reads fail	Reads succeed for some objects and fail for others	Reads succeed for some objects and fail for others
		2	All reads fail	All reads fail	Reads succeed for some objects and fail for others
	QUORUM (default)	1	All reads succeed	All reads succeed	All reads succeed
		2	All reads fail	Reads succeed for some objects and fail for others	Reads succeed for some objects and fail for others
	ONE	1 or 2	All reads succeed	All reads succeed	All reads succeed
		3 or more	All reads fail	All reads fail	All reads fail

### 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	Writes succeed for some objects and	Writes succeed for some objects 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
				fail for others	fail for others
		2	All writes fail	All writes fail	Writes succeed for some objects and fail for others
	QUORUM (default)	1	All writes succeed	All writes succeed	All writes succeed
		2	All writes fail	Writes succeed for some objects and fail for others	Writes succeed for some objects and fail for others
Reads	ALL	1 or 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
	QUORUM (default)	1	All reads succeed	All reads succeed	All reads succeed
		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

### 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	Writes succeed for some objects and fail for others	Writes succeed for some objects and fail for others
		2	All writes fail	All writes fail	Writes succeed for some objects and fail for others
	QUORUM (default)	1 or 2	All writes succeed	All writes succeed	All writes succeed
Reads	ALL	1 or 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

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
	QUORUM (default)	1 or 2	All reads succeed	All reads succeed	All reads succeed

### 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	Writes succeed for some objects and fail for others	Writes succeed for some objects and fail for others
		2	All writes fail	All writes fail	Writes succeed for some objects and fail for others
	QUORUM (default)	1 or 2	All writes succeed	All writes succeed	All writes succeed
Reads	ALL	1 or 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
	QUORUM (default)	1 or 2	All reads succeed	All reads succeed	All reads succeed

### 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 (for more detail on this

configurable feature see **"System Settings"** (page 342)). 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.

## 4.5. Security Features

### 4.5.1. HyperStore Shell (HSH)

#### 4.5.1.1. 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 every 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 90). 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 94).

##### 4.5.1.1.1. 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 Log"** (page 613).

#### 4.5.1.1.2. The HSH and Object Lock

If you want to use the HyperStore Object Lock feature (for WORM protection of object data in designated buckets), you must first enable the HSH and disable the root password. 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 **"WORM (Object Lock)"** (page 121).

For information about enabling HSH and disabling the root password, see **"Enabling the HSH and Managing HSH Users"** (page 90).

#### 4.5.1.2. Enabling the HSH and Managing HSH Users

*Subjects covered in this section:*

- **"Enabling the HSH"** (page 90)
- **"Adding HSH Users"** (page 91)
- **"Elevating a Regular HSH User to the "Trusted" Role"** (page 92)
- **"Deleting an HSH User"** (page 93)
- **"Disabling the root Password"** (page 93)

##### 4.5.1.2.1. Enabling the HSH

By default, after you have completed a fresh installation of HyperStore 7.2 or an upgrade to HyperStore 7.2, the HyperStore shell (HSH) is installed but disabled.

**Note** The exception is that HyperStore 7.2 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 Cloudian representative.

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

1. Log into the Puppet Master node as the *root* user.
2. Check to confirm that the HSH is currently disabled. (Here and in the steps that follow you will use *hsctl*, a new node management tool that remains mostly behind the scenes in HyperStore 7.2 but will be more prominent in future HyperStore releases. You can run the *hsctl* commands in this procedure from any directory.)

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

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

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

4. Push the configuration change out to the cluster.

```
# hsctl 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 91) below.

#### 4.5.1.2.2. 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, log into the CMC as the "admin" user and <a href="#">change the "admin" user's password</a> . (Alternatively, the password can be changed through the <a href="#">Admin API</a> .) 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, a legacy CMC System Admin User can log into the CMC and <a href="#">change their password</a> . (Alternatively, the password can be changed through the <a href="#">Admin API</a> .) 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 <a href="#">in the CMC</a> or through the <a href="#">Admin API</a> ), 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 131) and especially the sub-section **"Special Considerations for LDAP Authentication of System Administrators"** (page 134).

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 94).

**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.

#### 4.5.1.2.3. 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 94).

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 Puppet Master 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. Push the configuration change out to the cluster. (Here you are again using the `hscctl` tool that was mentioned previously).

```
$ hscctl 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.

#### 4.5.1.2.4. 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 an active CMC System Admin user.

#### 4.5.1.2.5. 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 [Object Lock](#) feature.

Before you can disable the *root* account password:

- HSH must be enabled (see **"Enabling the HSH"** (page 90))
- An HSH user account must be created for the CMC default System Admin user (see **"Adding HSH Users"** (page 91))

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 HyperStore nodes as *root* using a password but it will not prevent a *root* user from accessing HyperStore nodes using an SSH key, if in your environment SSH key access has been set up for the *root* user.

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

**Note** You will need to log in as *root* to perform this task -- you cannot do it while logged in as an HSH user.

1. As *root*, log into the Puppet Master node and then change into the [installation staging directory](#).
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.

#### 4.5.1.3. Using the HSH

*Subjects covered in this section:*

- **"Starting and Ending an HSH Session"** (page 95)
- **"HyperStore Commands and Utilities Supported by the HSH"** (page 96)
- **"Linux Commands and Utilities Supported by the HSH"** (page 99)
- **"General Restrictions On Command Usage"** (page 100)

#### 4.5.1.3.1. 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 90),) 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, d6b3c8d46ecaaaa69b62af25c4e7d4270f8b4d7e (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*.

#### 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.

#### 4.5.1.3.2. 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 92)).

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

Command	Purpose	More Information
hsrun	Run a binary file or script that is cryptographically signed by Cloudbian (such as a HyperStore release package file).	In shell, <i>hsrun -help</i>
hspkg Sub-commands include those listed below.	Manage HyperStore installation and configuration	In shell, <i>hspkg -help</i>
<ul style="list-style-type: none"> <li>hspkg setup (requires Trusted role)</li> </ul>	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 <i>HyperStore Installation Guide</i>
<ul style="list-style-type: none"> <li>hspkg install (requires Trusted role)</li> </ul>	Launch the HyperStore installer ( <i>cloudbianInstall.sh</i> )	In shell, <i>hspkg install --help</i>  See also:  <b>"Upgrading Your HyperStore Software Version"</b> (page 55)  <b>"Installer Advanced Configuration Options"</b> (page 501) <div> <b>Note</b>              You cannot perform           </div>

Command	Purpose	More Information
		<p>the "Disable the root password" task as an HSH user. See <b>"Disabling the root Password"</b> (page 93).</p> <p><b>"Pushing Configuration File Edits to the Cluster and Restarting Services"</b> (page 506)</p>
<ul style="list-style-type: none"> <li>hspkg config (requires Trusted role)</li> </ul>	View or edit a HyperStore configuration file	<p>In shell, <i>hspkg config --help</i></p> <p>Or for more detailed information see:</p> <p><b>"Using the HSH to Manage Configuration Files"</b> (page 510)</p>
<ul style="list-style-type: none"> <li>hspkg role (requires Trusted role)</li> </ul>	Manage HSH "Trusted" role membership	<p>In shell, <i>hspkg role --help</i></p> <p>Or for more detailed information see:</p> <p><b>"Elevating a Regular HSH User to the</b></p>

Command	Purpose	More Information
		<b>"Trusted Role"</b> (page 92)
<ul style="list-style-type: none"> <li>hspkg version</li> </ul>	Check the HyperStore software version	--
hslog	View a HyperStore log file	<p>In shell, <i>hslog -help</i></p> <p>Or for more detailed information see:</p> <p><b>"Using the HSH to View Logs"</b> (page 640)</p>
hscctl (requires Trusted role)	<i>hscctl</i> is a new HyperStore node management tool that is of limited use in HyperStore 7.2 but will be more prominent in future releases	--
hsstool	The HSH supports running any <i>hsstool</i> command, such as <i>hsstool repair</i> , <i>hsstool cleanup</i> , and so on	<p><b>"hsstool "</b> (page 643)</p> <div> <p><b>Note</b></p> <p>The HSH also supports running Cassandra <i>node-tool</i> commands, although you would typically use <i>hsstool</i> and not <i>node-tool</i>.</p> </div>
elasticsearchSync	Synchronize object metadata in your Elastic-	<b>"Elasticsearch</b>

Command	Purpose	More Information
	search cluster to object metadata currently in your HyperStore cluster	<b>Integration for Object Metadata"</b> (page 171)
install_jks.sh (requires Trusted role)	Copy keystore related files to or from the Puppet configuration directory, in connection with modifying TLS/SSL for HyperStore services.	<b>"HTTPS Support (TLS/SSL)"</b> (page 114)
rebrand_cmc.sh (requires Trusted role)	Copy image and resource files to or from the Puppet configuration directory, in connection with rebranding the CMC interface.	<b>"Rebranding the CMC UI"</b> (page 404)
jetty_password.sh	Create a Jetty-obfuscated password, in connection with modifying the HTTP/S Basic Authentication password for the Admin Service.	<b>"HTTP/S Basic Authentication for Admin API Access"</b> (page 748)

### Commands for HyperStore Appliances

The following commands pertain only to HyperStore appliances and would typically be used only on instructions from Cloudian Support. **All** of these commands require the Trusted role.

- disk\_list\_helper.py
- gem\_utils.py
- isdct
- sas3ircu
- sbuapp
- slot\_disk\_map\_1500.sh
- slot\_disk\_map\_4000.sh
- slot\_disk\_map\_hsx.sh
- slot\_disk\_map\_x3650.sh
- sluapp
- standard\_utils.py
- storcli64
- suuapp

#### 4.5.1.3.3. 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 92)).

blkid	iostat	kill (requires Trusted role)
-------	--------	------------------------------

cat	ip	ps
cd	ipmitool	pwd
chmod	keytool	reset
cp	kill (requires Trusted role)	rm
curl	less	rmdir
date	ls	scp
df	lsblk	sfdisk (requires Trusted role)
diff	lspci	smartctl
dmesg	lsscsi	ssh
dmidecode	md5sum	systemctl
domainname	mdadm (requires Trusted role)	tail
du (requires Trusted role)	mkdir	tar
file	mlnx_install (requires Trusted role)	top
grep	mv	truncate
gzip	nc (requires Trusted role)	uname
grub2-install (requires Trusted role)	netstat	unzip
grub2-mkconfig (requires Trusted role)	ntpq	vi
head	openssl (requires Trusted role)	vmstat
host	pgrep	wget
hostname	ping	zgrep

#### 4.5.1.3.4. General Restrictions On Command Usage

For security reasons, the HSH enforces the following general 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 shell.
- Entering more than one command on a line is not allowed. Characters such as ";", "&", and "|" are processed as literal characters, not as special characters.
- Input and output redirection are not allowed.
- Job control commands such as *ctrl-z* are not allowed.
- When listing files, the use of the wildcard character "\*" is not allowed.
- Tab completion for command names is supported but tab completion for file names is not.

### 4.5.2. HyperStore Firewall

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Enabling or Disabling the HyperStore Firewall"** (page 101)
- **"Default Behavior of the HyperStore Firewall When Enabled"** (page 102)
- **"Customizing the HyperStore Firewall"** (page 103)
- **"HyperStore Firewall Logging"** (page 105)

Each HyperStore node includes a built-in HyperStore Firewall that is pre-configured with settings appropriate for a typical HyperStore deployment. The HyperStore Firewall is either enabled or disabled by default depending on whether your original HyperStore installation was older than version 7.2:

- In systems originally installed as version 7.2 or newer, the HyperStore Firewall is enabled by default.
- In systems originally installed as a version older than 7.2 and then later upgraded to 7.2 or newer, the HyperStore Firewall is available but is disabled by default.

You can enable or disable the HyperStore Firewall using the installer's Advanced Configuration Options, as described below in **"Enabling or Disabling the HyperStore Firewall"** (page 101). When the Firewall is enabled, you can optionally customize certain aspects of the Firewall's behavior, as described further below in **"Customizing the HyperStore Firewall"** (page 103).

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. Alternatively, if you have upgraded to HyperStore 7.2 from an older version and you already have a custom firewall in place that you have been successfully using with HyperStore, you may prefer to keep using that firewall -- since in HyperStore 7.2 the HyperStore Firewall is limited as to how much it can be customized.

If you have upgraded to HyperStore 7.2 from an older version and you do wish to enable the HyperStore Firewall rather than continuing to use your own custom firewall, then before enabling the HyperStore Firewall do the following:

- 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**.
- Disable your existing firewall service on each node. For example, to disable *firewalld* do the following on each node:

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

The HyperStore installer will not allow you to enable the HyperStore Firewall on your nodes if existing firewall rules are in effect on any of the HyperStore nodes.

**Note** The HyperStore Firewall service is implemented as a custom version of the *firewalld* service, named *cloudian-firewalld*.

#### 4.5.2.1. Enabling or Disabling the HyperStore Firewall

To enable or disable the HyperStore Firewall on all your HyperStore nodes:

1. On the Puppet master node, change into your current HyperStore version [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 Puppet 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.

2. At the installer main menu enter **4** for "Advanced Configuration Options".
3. At the Advanced Configuration Options menu enter **s** for "Configure Firewall".
4. At the Cloudian HyperStore Firewall Configuration menu enter **a** for "Enable/Disable Cloudian HyperStore Firewall".
5. In the Enable/Disable Cloudian HyperStore Firewall interface, the Firewall's current status is displayed. At the prompt enter **enable** to enable the Firewall or **disable** to disable the Firewall. After the interface indicates that the Firewall is set as you specified, press any key to return to the Firewall Configuration menu.
6. At the Firewall Configuration menu enter **x** for "Apply configuration changes and return to previous menu". Then at the next prompt that displays enter **yes** to confirm that you want to apply your configuration changes. This will apply your change to all nodes in your HyperStore system (all nodes in all of your data centers and service regions).

The installer interface should then display a status message "result: **OK**" for each node, to indicate the successful applying of your configuration change to each node.

**Note** If the installer displays a Warning about one or more nodes not responding you can try the Firewall Configuration menu's **x** option again, and this retry may work for the node(s) if the problem the first time was a temporary network issue. If one of your nodes is **down**, the node will automatically be updated with your configuration change when the node comes back online.

You are now done with enabling or disabling the HyperStore Firewall. **You do not need to do a Puppet push or restart any services.**

#### 4.5.2.2. 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**. 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**:
  - 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)

Traffic originating from sources other than HyperStore nodes will be blocked (DROP'd) on all TCP ports other than those listed above.

**Note** The 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 Firewall is enabled, the 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 Firewall accommodates this change automatically. In the case of expanding your cluster, the Firewall will be automatically enabled on new nodes, and the Firewall on the existing nodes will allow inbound traffic from the new nodes, on any port. In the case of removing nodes, the 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 Firewall is disabled when you add new nodes to your cluster, the Firewall will also be disabled on the new nodes.

- If you change the port number that a particular HyperStore public service uses, the Firewall's configuration is automatically adjusted accordingly. After you complete a port change you only need to apply the updated Firewall configuration to the cluster. For instructions see "**Changing S3, Admin, or CMC Listening Ports**" (page 599).

#### 4.5.2.3. Customizing the HyperStore Firewall

The default behavior of the HyperStore Firewall (when enabled) is as described above. If you wish you can customize the behavior by denying access to one of the services that the Firewall allows access to by default. For example, you could deny access to the S3 HTTP service so that the S3 HTTPS service is used exclusively. Subsequently, if there is a change in your preferences or circumstances, you could customize the Firewall to once again allow access to that service.

To customize the HyperStore Firewall on all your HyperStore nodes:

1. On the Puppet master node, change into your current HyperStore version installation staging directory. Then launch the installer.

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

2. At the installer main menu enter **4** for "Advanced Configuration Options".
3. At the Advanced Configuration Options menu enter **s** for "Configure Firewall".
4. At the Cloudian HyperStore Firewall Configuration menu enter the menu letter corresponding to the service for which you want to deny or allow access (for example **h** for S3 HTTP).

```
Cloudian HyperStore(R) Firewall Configuration
-----

When enabling the Cloudian HyperStore(R) Firewall, all internal services
like Cassandra and Hyperstore are automatically protected to outside access
and by default, connections to all public-facing services like S3 and IAM,
are all allowed. External access to each service can be further managed by
enabling (Allow) or disabling (Deny) traffic to each public Service.

a ) Enable/Disable Cloudian HyperStore(R) Firewall
b ) Configure access to Admin API HTTP
c ) Configure access to Admin API HTTPS
d ) Configure access to CMC HTTP
e ) Configure access to CMC HTTPS
f ) Configure access to IAM HTTP
g ) Configure access to IAM HTTPS
h ) Configure access to S3 HTTP
i ) Configure access to S3 HTTPS
j ) Configure access to SQS HTTP
k ) Configure access to SQS HTTPS

x ) Return to previous menu

Choice: █
```

5. In the Allow/Deny Access to Service <Service Type> interface, the service's current setting is displayed ("Allow" or "Deny"). At the prompt enter **deny** to deny access to the service or **allow** to allow access to the service. After the interface indicates that the service is set as you specified, press any key to return to the Firewall Configuration menu.
6. At the Firewall Configuration menu enter **x** to apply your configuration changes. Then at the next prompt that displays enter **yes** to confirm that you want to apply your configuration changes. This will apply your change to all nodes in your HyperStore system (all nodes in all of your data centers and service regions).

The installer interface should then display a status message "result: **OK**" for each node, to indicate the successful applying of your configuration change to each node.

**Note** If the installer displays a Warning about one or more nodes not responding you can try the Firewall Configuration menu's **x** option again, and this retry may work for the node(s) if the problem the first time was a temporary network issue. If one of your nodes is **down**, the node will automatically be updated with your configuration change when the node comes back online.

You are now done with customizing the HyperStore Firewall. **You do not need to do a Puppet push or restart any services.**

#### 4.5.2.4. HyperStore Firewall Logging

On each node, requests blocked by the HyperStore Firewall are logged to `/var/log/firewall.log`. For more information about this log including its rotation and retention behavior, see **"HyperStore Firewall Log"** (page 609).

### 4.5.3. Server-Side Encryption

#### 4.5.3.1. Server-Side Encryption Feature Overview

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Encryption Configuration Changes Do Not Apply Retroactively"** (page 106)
- **"Encryption and Auto-Tiering"** (page 106)
- **"Encryption and Cross-Region Replication"** (page 106)

The HyperStore system supports server-side encryption (SSE) to protect the confidentiality of data at rest. Several different methods of server-side encryption are supported:

- Encryption using a HyperStore **system-generated encryption key** (regular SSE)
- Encryption using a **customer-provided encryption key** (SSE-C)
- Encryption using encryption keys managed by the **Amazon Web Services Key Management Service** (AWS KMS)

The selection of whether to use server-side encryption, and which method to use, can be made at the object level (as specified by headers in the object upload request), or the bucket level (so that a default encryption method is applied to all objects uploaded to the bucket), or the storage policy level (so that a default encryption method is applied to all objects uploaded to any bucket that uses the storage policy). When the system is processing a given object upload, the precedence ordering among these different configuration levels is as follows:

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.

Put differently, any encryption configuration specified by object upload request headers takes precedence over the bucket default configuration; and the bucket default configuration takes precedence over the storage policy configuration. If no encryption method is specified in either the object upload request, the configuration of the bucket into which the object is being uploaded, or the configuration of the storage policy used by the bucket, then no server-side encryption is applied to that object.

For more information about using the supported server-side encryption methods, see:

- **"Using Regular SSE"** (page 107)
- **"Using SSE-C"** (page 108)
- **"Using AWS KMS"** (page 110)

#### 4.5.3.1.1. Encryption Configuration Changes Do Not Apply Retroactively

You cannot apply server-side encryption 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.

#### 4.5.3.1.2. 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.

Encryption in HyperStore	Tiering Destination	Object Handling
Regular SSE	Amazon, Google, or HyperStore	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.
	Azure or Spectra BlackPearl	HyperStore decrypts the object, then tiers it to the destination system in decrypted form. These destinations do not support a server-side encryption option in object upload requests.
SSE-C	Any	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.
AWS KMS	Amazon, Google, or HyperStore	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.
	Azure or Spectra BlackPearl	HyperStore decrypts the object, then tiers it to the destination system in decrypted form. These destinations do not support a server-side encryption option in object upload requests.

#### 4.5.3.1.3. 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 target replication bucket.
- Objects encrypted by the SSE-C method or the AWS KMS method **are not** replicated to the target replication bucket.

### 4.5.3.2. Using Regular SSE

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Recommended System Set-Up Before Using SSE"** (page 107)
- **"Requesting SSE for Specific Objects (CMC or S3 API)"** (page 107)
- **"Configuring SSE as the Default for a Bucket (S3 API Only)"** (page 108)
- **"Configuring SSE as the Default for a Storage Policy (CMC Only)"** (page 108)

**Regular server-side encryption (SSE)** can be configured at the object level, the bucket level, or the storage policy level. For information about the precedence ordering among these levels see **"Server-Side Encryption"** (page 105).

When regular SSE is used the HyperStore system generates a unique encryption key for each object that the system encrypts, and the encryption key is stored as part of the object's metadata.

#### 4.5.3.2.1. Recommended System Set-Up Before Using SSE

With regular SSE, the HyperStore system generates the 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, Inc. recommends using AES-256. You can enable AES-256 in your HyperStore system as described in **"Enabling AES-256"** (page 112).

**Note** Amazon uses AES-256 for its regular SSE implementation, and AES-256 is called for in Amazon's SSE specification.

#### 4.5.3.2.2. Requesting SSE for Specific Objects (CMC or S3 API)

Regular SSE can be set for specific objects as those objects are uploaded to the HyperStore system. This can be done either through the CMC or through a third party S3 client application that invokes the HyperStore implementation of the S3 API.

##### Object-Level SSE Through the CMC

In the CMC's **Buckets & 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). For more information see **"Upload an Object"** (page 246).

##### Object-Level SSE Through the S3 API

In compliance with the Amazon S3 REST API, the HyperStore system's S3 API implementation 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:

- [PUT Object](#)
- [Initiate Multipart Upload](#)
- [Upload Part](#)
- [POST Object](#)
- [PUT Object - copy](#)

**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*.

#### 4.5.3.2.3. Configuring SSE as the Default for a Bucket (S3 API Only)

In compliance with the Amazon S3 REST API, the HyperStore system's S3 API supports setting and managing a bucket default server-side encryption method by the use of these operations:

- [PUT Bucket encryption](#)
- [GET Bucket encryption](#)
- [DELETE Bucket encryption](#)

To configure a bucket for regular SSE, in the *PUT Bucket encryption* request body set the *SSEAlgorithm* element to *AES256*.

**Note** If you have not [enabled AES-256](#) in your HyperStore system, the system will use AES-128 encryption even though the *SSEAlgorithm* element specifies *AES256*.

**Note** The CMC does not support setting a bucket default server-side encryption method.

#### 4.5.3.2.4. Configuring SSE as the Default for a Storage Policy (CMC Only)

When you use the CMC to create storage policies, one of the configurable policy attributes is server-side encryption. To configure a storage policy to use regular SSE, in the "Server-Side Encryption" field of the storage policy configuration interface, select "SSE".

For more information on storage policy configuration see:

- **"Add a Storage Policy"** (page 353)
- **"Edit a Storage Policy"** (page 377)

#### 4.5.3.3. Using SSE-C

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Required System Set-Up Before Using SSE-C"** (page 109)
- **"Requesting SSE-C for Specific Objects (S3 API Only)"** (page 109)

**Server-side encryption with customer-provided encryption keys (SSE-C)** can only be set at the per-object level, and only if you use a third party S3 client that supports requesting this type of encryption. Setting SSE-C as the default encryption method for a bucket or a storage policy is not supported.

When 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.

**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 \(SSE-C\)](#).

#### 4.5.3.3.1. Required System Set-Up Before Using SSE-C

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. For instructions see **"Enabling AES-256"** (page 112).
- 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 **"HTTPS Support (TLS/SSL)"** (page 114).

**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 parameter `mts.properties.erb:"cloudian.s3.ssec.usessl"` (page 577).

#### 4.5.3.3.2. Requesting SSE-C for Specific Objects (S3 API Only)

In compliance with the Amazon S3 REST API, the HyperStore system'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:

- [PUT Object](#)
- [Initiate Multipart Upload](#)
- [Upload Part](#)
- [POST Object](#)
- [PUT Object - copy](#) (supporting also the `x-amz-copy-source-server-side-encryption-customer-*` request headers)
- [GET Object](#)
- [HEAD Object](#)

For the full list of supported `x-amz-server-side-encryption-customer-*` request headers for each of these operations, follow the links above.

**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.

#### 4.5.3.4. Using AWS KMS

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Required System Set-Up Before Using AWS KMS Encryption"** (page 110)
- **"Requesting AWS KMS Encryption for Specific Objects (S3 API Only)"** (page 111)
- **"Configuring AWS KMS Encryption as the Default for a Bucket (S3 API Only)"** (page 112)
- **"Deleting a Bucket That Uses AWS KMS Encryption"** (page 112)

Server-side encryption using encryption keys managed by the Amazon Web Services Key Management Service (AWS KMS) can be configured at the object level or the bucket default level, only if you use a third party S3 client that supports requesting this type of encryption. Setting the AWS KMS method as the default for a storage policy is not supported. For information about the precedence ordering between these levels see **"Server-Side Encryption"** (page 105).

When AWS KMS based encryption is used the HyperStore system triggers in the remote 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 Redis) a CMK ID that allows HyperStore to tell AWS KMS which CMK to use for creating an 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 deletes the plain text data key, while the encrypted version of the data key is stored within the encrypted object data.

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 decrypted data key to HyperStore, which uses it to decrypt the object and then deletes the decrypted data key from memory.

**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 \(KMS\)](#).

##### 4.5.3.4.1. Required System Set-Up Before Using AWS KMS Encryption

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 your Puppet master node, open the following file in a text editor:

```
/etc/cloudian-<version>-  
puppet/modules/cloudians3/files/awscredentials.properties
```

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 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 your Puppet master node, open the following file in a text editor:

```
/etc/cloudian-<version>-puppet/modules/cloudians3/templates/mts.properties.erb
```

4. Find the property `util.awskmsutil.region` and set it to the AWS service region of the AWS KMS that you want HyperStore to use. The default is "us-east-1". Save your change and close the file.
5. Push your changes to the cluster and restart the S3 Service. If you need instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

#### 4.5.3.4.2. Requesting AWS KMS Encryption for Specific Objects (S3 API Only)

In compliance with the Amazon S3 REST API, the HyperStore system'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:

- [PUT Object](#)
- [Initiate Multipart Upload](#)
- [Upload Part](#)
- [POST Object](#)
- [PUT Object - copy](#)

**Note** 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.

**Note** 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.

#### 4.5.3.4.3. Configuring AWS KMS Encryption as the Default for a Bucket (S3 API Only)

In compliance with the Amazon S3 REST API, the HyperStore system's S3 API supports setting and managing a bucket default server-side encryption method by the use of these operations:

- [PUT Bucket encryption](#)
- [GET Bucket encryption](#)
- [DELETE Bucket encryption](#)

To configure a bucket for server-side encryption with AWS KMS, in the *PUT Bucket encryption* request body set the *SSEAlgorithm* element to *aws:kms*.

**Note** The CMC does not support setting a bucket default server-side encryption method.

#### 4.5.3.4.4. 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.

#### 4.5.3.5. Enabling AES-256

By default the HyperStore system does not use AES-256, the most secure form of the Advanced Encryption Standard. Instead it uses AES-128.

You must enable AES-256 in your HyperStore system if you want to do either of the following:

- Use regular SSE in a manner compliant with the Amazon SSE specification
- Use SSE-C

**Note** Enabling AES-256 is not necessary for -- and not relevant to -- server-side encryption using AWS KMS.

To enable AES-256 in your HyperStore system, do the following:

1. On the Puppet master node, in the [common.csv](#) file, set *cloudian\_s3\_aes256encryption\_enabled* to *true*. (By default it is set to *false*.)
2. Push your changes out to the cluster and restart the S3 Service. For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

AES-256 is now enabled in your HyperStore system.

### 4.5.4. FIPS Support

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Enabling FIPS Compliance for SSH"** (page 113)
- **"Limitations to FIPS Compliance of Server-Side Encryption"** (page 113)
- **"HTTPS and HTTP"** (page 114)

Federal Information Processing Standard (FIPS) Publication 140-2 defines security requirements for cryptographic modules. Starting with HyperStore version 7.2, HyperStore cryptographic functions are designed and implemented to meet the requirements of this standard. As of the time of the 7.2 Early Availability release, third party certification of HyperStore's FIPS 140-2 compliance is in progress.

Starting with HyperStore version 7.2, HyperStore in most respects meets the FIPS 140-2 standard by default. In particular, HyperStore by default meets FIPS 140-2 requirements for:

- **AES encryption** (used for HyperStore's [server-side encryption](#) feature)
- **SHA-256** (used to hash request payloads for some types of S3 requests)
- **HMAC** (used for Signature version 4 validation of S3 requests)

HyperStore's cryptographic module meets FIPS 140-2 requirements for AES, SHA, and HMAC by utilizing the OpenSSL FIPS Object Module 2.0 -- which is FIPS 140-2 certified -- enveloped by an OpenSSL4J JNI (Java Native Interface) wrapper. Through this wrapper, the HyperStore S3 Service makes calls for AES, SHA, or HMAC functions whenever they are needed.

**Note** For the full FIPS 140-2 standard see <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf>.

#### 4.5.4.1. Enabling FIPS Compliance for SSH

By default, the **SSH** service on HyperStore nodes -- OpenSSH -- is **not** FIPS 140-2 compliant because it supports ciphers that are not FIPS 140-2 approved as well as ciphers that are FIPS 140-2 approved. If you wish you can configure HyperStore so that only FIPS 140-2 approved ciphers are used for SSH connections to HyperStore nodes.

To make HyperStore's SSH implementation FIPS compliant, follow these steps:

1. On the Puppet master node, in the [common.csv](#) file, set `fips_enabled` to `true`. (By default it is set to `false`.)
2. Push your configuration change out to the cluster and then restart the S3 Service and the CMC. If you need instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

#### 4.5.4.2. Limitations to FIPS Compliance of Server-Side Encryption

As described in **"Server-Side Encryption"** (page 105), HyperStore supports several types of server-side encryption. For Regular SSE, HyperStore generates the encryption keys and implements the encryption and decryption of object data. This type of server-side encryption is fully FIPS 140-2 compliant, starting with HyperStore version 7.2.

However, HyperStore also supports methods of server-side encryption for which the encryption keys come from outside of HyperStore. With SSE-C, the keys are provided by the user. With server-side encryption using AWS KMS, the keys are generated by an external key management system.

For these types of server-side encryption, although HyperStore executes the encryption and decryption of object data using FIPS-compliant AES, the generation of the encryption keys is outside HyperStore control. These types of server-side encryption are fully FIPS-compliant only if the keys are generated in a FIPS-compliant manner.

#### 4.5.4.3. HTTPS and HTTP

Starting with HyperStore version 7.2, HyperStore's HTTPS listeners support only TLS 1.2 and no older versions of TLS. This is in keeping with the recommendations of most contemporary security standards.

By default several of HyperStore's services also support regular HTTP connections. To enhance system security, you can configure HyperStore services to only support HTTPS connections and not regular HTTP connections. For more information see **"HTTPS Support (TLS/SSL)"** (page 114).

#### 4.5.5. HTTPS Support (TLS/SSL)

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Managing HTTPS and Certificate Keystores for HyperStore Services"** (page 115)
- **"Disabling Regular HTTP for HyperStore Services"** (page 119)
- **"Configuring HTTP/S Basic Authentication for the Admin Service"** (page 120)

The table below describes the default status of HyperStore services in respect to using HTTPS and HTTP for incoming connections. These services differ as to whether HTTPS support is enabled by default; whether a default TLS/SSL certificate keystore for the service is included in HyperStore; and whether the service accepts regular (non-secure) HTTP connections by default.

Service	Is HTTPS Enabled By Default?	Is There a Default Key-store?	Is Regular HTTP Allowed By Default?
S3	No	No	Yes
IAM	Yes	Yes	Yes
CMC	Yes	Yes	No
Admin	Yes	Yes	No for systems originally installed as HyperStore version 6.0.2 or newer Yes for systems originally installed as a version older than 6.0.2

**Note** All HyperStore HTTPS listeners use TLS v1.2 and will not accept client connections that use TLS versions older than 1.2.

**Note** In the current HyperStore release, the [Simple Queue Service](#) (SQS) does not support HTTPS. Only regular HTTP access is supported for SQS.

#### 4.5.5.1. Managing HTTPS and Certificate Keystores for HyperStore Services

The HyperStore installer's "Advanced Configuration Options" menu provides a simple and highly automated way to manage HTTPS and the associated keystores (in which reside TLS/SSL certificates) for HyperStore services. To access the "Advanced Configuration Options" menu:

1. Log into the Puppet master node and change into the [installation staging directory](#).
2. 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 Puppet 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.

3. At the installer's main menu enter **4** for "Advanced Configuration Options". This takes you to the Advanced Configuration Options menu:

```
Advanced Configuration Options
-----

a ) Change server role assignments
b ) Change S3, Admin and CMC ports
c ) Change S3, S3-Website, Admin, or CMC endpoints
d ) Configure diagnostic data collection options
e ) Configure SSL for S3
f ) Configure SSL for CMC
g ) Configure SSL for IAM
h ) Remove existing Puppet SSL certificates
i ) Start or stop Puppet daemon
j ) Remove Puppet access lock
k ) Enable or disable DNSMASQ
l ) Configure Performance Parameters on Nodes
m ) Disable the root password
r ) Exclude host(s) from configuration push and service restarts
s ) Configure Firewall
x ) Return to Main Menu

Choice: █
```

From the Advanced Configuration Options menu you can choose to configure SSL for either the S3 service, the CMC service, or the IAM service.

**Note** Reconfiguration of the Admin Service's keystore and HTTPS listener is not supported.

*If you are using the HyperStore Shell*

In some of the procedures that follow, some steps call for you to copy SSL keystore related files into the Puppet configuration directory, or copies files out from there. If you are using the [HyperStore Shell](#) (HSH), you will not be allowed direct access to that directory (you will not be able to directly copy into it or from it). Instead, when logged into the Puppet master node using the HSH, create or choose a working directory under your home directory. Then change into the working directory. Then you can use the HSH's `install_jks.sh` tool to copy files between the working directory and the Puppet configuration directory. You must be an HSH [Trusted user](#) to run this tool. The syntax is as follows:

- `install_jks.sh put <filename> [--cmc]`

Copy the specified file from the current directory (the working directory) to the Puppet configuration directory. Use the `--cmc` option only if you are configuring SSL for the CMC Service. Do not use the `--cmc` option if you are configuring SSL for the S3 Service or the IAM Service.

For example, if you are following the procedure for having the S3 Service use a keystore that you provide (as described further below), in the step where the procedure calls for copying the keystore file into the Puppet configuration directory, instead of copying it there directly you can put the keystore file into your working directory and use the `install_jks.sh put <filename>` command to copy the file to the Puppet configuration directory.

- `install_jks.sh get <filename> [--cmc]`

Copy the specified file from the Puppet configuration directory to the current directory (the working directory). Use the `--cmc` option only if you are configuring SSL for the CMC Service. Do not use the `--cmc` option if you are configuring SSL for the S3 Service or the IAM Service.

For example, if you are following the procedure for having the S3 Service use a CA-signed certificate (as described further below), in the step where the procedure calls for getting the CSR file from the Puppet configuration directory and sending it to your CA, instead of getting it directly from the Puppet configuration directory you can use the `install_jks.sh get <filename>` command to copy it from the Puppet configuration directory into the working directory, and then from there you can send it to your CA.

#### 4.5.5.1.1. S3 Service

For the S3 Service, HTTPS is disabled by default and there is no default keystore. To set up HTTPS for the S3 Service, from the installer's Advanced Configuration Options menu enter **e** for "Configure SSL for S3". Then from the S3 SSL Configuration menu, the tasks to complete depend on what sort of certificate and keystore you want to use for the S3 HTTPS listener.

**To use a self-signed certificate in a keystore that HyperStore creates:**

1. Enter **a** for "Generate keystore for S3" and then follow the prompts to complete that task. When generating the keystore you can accept the default configuration values (which are displayed at the prompts) or customize those values.
2. Back at the S3 SSL Configuration menu, enter **b** for "Enable/Disable HTTPS for S3" and follow the prompts to complete that task.
3. Go to the installer's main menu and use the **b** "Cluster Management" menu to first push the configuration changes out to the cluster and then restart the S3 Service. Then exit the installer.

**To use a CA-signed certificate in a keystore that HyperStore creates:**

1. Enter **a** for "Generate keystore for S3" and then follow the prompts to complete that task. When generating the keystore you can accept the default configuration values (which are displayed at the prompts) or customize those values. Make a note of the keystore name and password -- you will need those later in this procedure. When you've completed this task exit the installer, but remain logged into the Puppet master node.
2. Step 1 creates under the Puppet master node directory `/etc/cloudian-7.2.3-puppet/modules/baselayout/files` both the keystore and a Certificate Signing Request (CSR) that by default is named `<regionname>_s3.csr`. Submit the CSR file to your preferred Certificate Authority (CA), using the submission instructions from the CA. When you get the certificate files back from the CA, continue to Step 3 below.

**Note** HyperStore requires that the certificate files be in PEM format, which is the most common format.

3. Copy the certificate files you received from the CA into the installation staging directory on the Puppet master node. This typically includes root certificate file(s), intermediate certificate file(s), and the CA-signed certificate file itself. You will need to provide the file names in the next step below.
4. Launch the installer again and navigate back to the S3 SSL Configuration menu. Then enter **c** for "Import CA-signed certificate into keystore" and follow the prompts to complete that task. As part of this you will need to provide the certificate file names.
5. Back at the S3 SSL Configuration menu, enter **b** for "Enable/Disable HTTPS for S3" and follow the prompts to complete that task.

**Note** If you had already enabled HTTPS for S3 previously -- with a self-signed version of the certificate, or with a different certificate -- then menu item **b** will be labeled "Enable new keystore for S3" (rather than "Enable/Disable HTTPS for S3"). Still you must complete task **b** here. Completing task **b** here is necessary to correctly apply your changes.

6. Go to the installer's main menu and use the **b** "Cluster Management" menu to first push the configuration changes out to the cluster and then restart the S3 Service. Then exit the installer.

**To have the S3 Service use an existing keystore that you provide:**

1. Copy the keystore file into the directory `/etc/cloudian-7.2.3-puppet/modules/baselayout/files` on the Puppet master node.
2. At the installer's S3 SSL Configuration menu, enter **d** for "Import already existing keystore for S3" and then follow the prompts to complete that task.
3. Back at the S3 SSL Configuration menu, enter **b** for "Enable/Disable HTTPS for S3" and follow the prompts to complete that task.

**Note** If you had already enabled HTTPS for S3 previously -- with a different keystore -- then menu item **b** will be labeled "Enable new keystore for S3" (rather than "Enable/Disable HTTPS for S3"). Still you must complete task **b** here. Completing task **b** here is necessary to correctly apply your changes.

4. Go to the installer's main menu and use the **b** "Cluster Management" menu to first push the configuration changes out to the cluster and then restart the S3 Service. Then exit the installer.

#### 4.5.5.1.2. CMC or IAM Service

For the CMC and for the IAM Service, HTTPS is enabled by default and there are default keystores in place. You do not need to take any action to use HTTPS for the CMC or the IAM Service. However, you can take any of the optional actions described below, after navigating from the Advanced Configuration Options menu to either the CMC SSL Configuration menu or the IAM SSL Configuration menu.

**Note** In the instructions that follow, "(CMC or IAM)" indicates that you will see either "CMC" or "IAM" in the installer interface text depending on which service you're working with.

**To create and use a new keystore and self-signed certificate, rather than the default keystore and certificate:**

1. Enter **a** for "Generate keystore for (CMC or IAM)" and then follow the prompts to complete that task. When generating the keystore you can accept the default configuration values (which are displayed at the prompts) or customize those values.
2. Back at the (CMC or IAM) SSL Configuration menu, enter **b** for "Enable new keystore for (CMC or IAM)" and follow the prompts to complete that task.
3. Go to the installer's main menu and use the **b** "Cluster Management" menu to first push the configuration changes out to the cluster and then restart the (CMC or IAM) Service. Then exit the installer.

**To create and use a new keystore and CA-signed certificate, rather than the default keystore and certificate:**

1. Enter **a** for "Generate keystore for (CMC or IAM)" and then follow the prompts to complete that task. When generating the keystore you can accept the default configuration values (which are displayed at the prompts) or customize those values. Make a note of the keystore name and password -- you will need those later in this procedure. When you've completed this task exit the installer, but remain logged into the Puppet master node.
2. Step 1 creates on the Puppet master node both the keystore and a Certificate Signing Request (CSR):
  - For CMC the keystore and CSR are in directory `/etc/cloudian-7.2.3-puppet/modules/cmc/files` and the CSR by default is named `cmc.csr`.
  - For IAM the keystore and CSR are in directory `/etc/cloudian-7.2.3-puppet/modules/baselayout/files` and the CSR by default is named `iam-1.csr`.

Submit the CSR file to your preferred Certificate Authority (CA), using the submission instructions from the CA. When you get the certificate files back from the CA, continue to Step 3 below.

**Note** HyperStore requires that the certificate files be in PEM format, which is the most common format.

3. Copy the certificate files you received from the CA into the installation staging directory on the Puppet master node. This typically includes root certificate file(s), intermediate certificate file(s), and the CA-signed certificate file itself. You will need to provide the file names in the next step below.

4. Launch the installer again and navigate back to the (CMC or IAM) SSL Configuration menu. Then enter **c** for "Import CA-signed certificate into keystore" and follow the prompts to complete that task. As part of this you will need to provide the certificate file names.
5. Back at the (CMC or IAM) SSL Configuration menu, enter **b** for "Enable new keystore for (CMC or IAM)" and follow the prompts to complete that task.
6. Go to the installer's main menu and use the **b** "Cluster Management" menu to first push the configuration changes out to the cluster and then restart the (CMC or IAM) Service. Then exit the installer.

**(IAM Only) To have the IAM Service use an existing keystore that you provide:**

1. Copy the keystore file into the directory `/etc/cloudian-7.2.3-puppet/modules/baselayout/files` on the Puppet master node.
2. At the installer's IAM SSL Configuration menu, enter **d** for "Import already existing keystore for IAM" and then follow the prompts to complete that task.
3. Back at the IAM SSL Configuration menu, enter **b** for "Enable new keystore for IAM" and then follow the prompts to complete that task.
4. Go to the installer's main menu and use the **b** "Cluster Management" menu to first push the configuration changes out to the cluster and then restart the IAM Service. Then exit the installer.

#### 4.5.5.2. Disabling Regular HTTP for 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 are using the HyperStore firewall, see "**Customizing the HyperStore Firewall**" (page 103) for instructions. (If you have not yet enabled the HyperStore firewall, and want to do so, see "**HyperStore Firewall**" (page 100)).
- Set the **HyperStore system configuration** so that the regular HTTP listeners are disabled for the Admin Service, for the IAM Service, and for the CMC.

**Note** The S3 Service does not have a system configuration setting for disabling the regular HTTP listener. Use a firewall to block regular HTTP access to the S3 Service, as stated above.

On the Puppet master node, in the configuration file [common.csv](#), there are settings that enable or disable the regular HTTP listener for the Admin Service, for the IAM Service, and for the CMC:

- For the Admin Service, set `admin_secure` 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 IAM Service, set `iam_secure` to `true` (it defaults to `false`)
- For the CMC, set `cmc_web_secure` to `true` if it is not already set to `true` (it defaults to `true`)

After making any edits to `common.csv`, launch the installer and use the **b** "Cluster Management" menu to first push the configuration changes out to the cluster and then restart each of the services for which you changed the configuration. (To restart the Admin Service, restart the S3 Service -- this has the effect of restarting the Admin Service also.) Then exit the installer.

#### 4.5.5.3. Configuring HTTP/S Basic Authentication for the Admin Service

The Admin Service is unique among HyperStore's HTTP/S-based services in that it requires that clients use HTTP/S Basic Authentication. With HTTP/S Basic Authentication, the connecting client must provide the correct user name and password in order for the Admin Service to accept the HTTP/S connection request. The user name and password are part of the Admin Service configuration. By default the required user name for this purpose is "sysadmin" and the default password is either a randomly generated password unique to your system (if your original HyperStore install was version 7.2.2 or newer) or "public" (if your original HyperStore install was older than version 7.2.2). **If the Admin API HTTP/S Basic Authentication password in your system is "public", you should change it to something more secure.** For more information and instructions see **"HTTP/S Basic Authentication for Admin API Access"** (page 748).

#### 4.5.6. Secure Delete

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Enabling Secure Delete"** (page 121)
- **"Secure Delete Logging"** (page 121)

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 23)). 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.

**Using secure delete impacts system performance for delete operations.** Consult with your Clouidian representative if you are considering using secure delete.

**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.

**Note** Secure delete does not apply to object metadata stored 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 application(s) to information that does not require secure delete.

#### 4.5.6.1. Enabling Secure Delete

To enable the secure delete feature:

1. On the Puppet Master node, in the [hyperstore-server.properties.erb](#) file, set `secure.delete` to `true`. (By default it is set to `false`.)
2. Use the installer to push out your configuration change to the cluster and to restart the HyperStore Service. If you need instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

#### 4.5.6.2. 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 609).

### 4.5.7. WORM (Object Lock)

#### 4.5.7.1. WORM (Object Lock) Feature Overview

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Object Lock Audit Logging"** (page 122)
- **"Object Lock and Bucket Lifecycle Policies"** (page 122)
- **"Object Lock and Cross-Region Replication"** (page 123)
- **"Object Lock and S3 Notifications"** (page 123)
- **"Object Lock and User Deletion"** (page 123)

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 enables this feature. Also, to use the Object Lock feature you must enable the HyperStore Shell (HSH) and disable the `root` account password on your HyperStore hosts. For more information see the "Prerequisites" section in **"Setting Up Object Lock"** (page 123).

If your license supports Object Lock and you have enabled the HSH and disabled the `root` account password, then the HyperStore S3 Service supports all the standard AWS S3 API methods and headers associated with the S3 Object Lock feature. Using these standard S3 API methods and headers, S3 client applications can:

1. Enable Object Lock on **new buckets** as those bucket are created. Note that:
  - Object Lock can only be enabled on newly created buckets, as part of the operation that creates the bucket. Object Lock cannot be enabled on buckets that already exist.
  - Enabling Object Lock on a bucket as the bucket is created automatically enables Versioning on the bucket. Object Lock can only be used in combination with Versioning.
  - Enabling Object Lock on a bucket as the bucket is created does not by itself have the effect of locking objects that are subsequently stored in that bucket. It only makes it possible to lock such objects, using the methods described in points number 2 and 3 below.
2. For an Object Lock enabled bucket, optionally set a **bucket default Object Lock configuration** that will apply to all objects that are subsequently created in the bucket. The default Object Lock configuration specifies a **Retention** time period that will be applied to objects that are subsequently created in the bucket. Each object's retention period starts when the object is created in the bucket (and for objects with multiple versions, each object version's retention period starts when that object version is created). The default Object Lock configuration also specifies which of two modes the Object Lock is implemented in:
  - **Governance** mode, which allows privileged users to change the retention period or delete objects before their retention period completes.
  - **Compliance** mode, which does not allow any user to change the retention period or delete objects before their retention period completes.
3. For an Object Lock enabled bucket, optionally set **Object Lock attributes on individual objects**, either as the objects are created in the bucket or after the objects have been created in the bucket. The Object Lock set on an object can be either or both of:
  - **Retention**, in Governance mode or Compliance mode
  - **Legal Hold**, which applies for an indefinite period until explicitly released. While objects are in Legal Hold, no user can delete them.

These per-object Object Lock attributes override the bucket's default Object Lock configuration, if a default configuration has been set.

For more information on applying Object Lock to buckets and objects in HyperStore, see "**Setting Up Object Lock**" (page 123).

For general information on the AWS S3 Object Lock feature, in the AWS documentation see:

- [Amazon S3 Object Lock Overview](#)
- [Locking Objects Using Amazon S3 Object Lock](#)
- [Managing Amazon S3 Object Locks](#)

#### 4.5.7.1.1. 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* that resides on each HyperStore node. For more information see "**S3 Service Logs (including Auto-Tiering, CRR, and WORM)**" (page 619).

#### 4.5.7.1.2. 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.

Also, the system 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.

#### 4.5.7.1.3. Object Lock and Cross-Region Replication

In the current release, 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.

#### 4.5.7.1.4. Object Lock and S3 Notifications

In the current release, HyperStore does not support S3 Notifications in regard to Object Lock related events.

#### 4.5.7.1.5. 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, see **"Object Protection Under Governance Retention, Compliance Retention, and Legal Hold"** (page 128).

### 4.5.7.2. Setting Up Object Lock

*Subjects covered in this section:*

- **"Prerequisites"** (page 123)
- **"Setting Up Object Lock on a Bucket (S3 API or CMC)"** (page 124)
- **"Setting Object Lock Attributes on Individual Objects (S3 API or CMC)"** (page 126)
- **"Object Protection Under Governance Retention, Compliance Retention, and Legal Hold"** (page 128)

#### 4.5.7.2.1. Prerequisites

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 "Enabled" or "Disabled". If you want to use the Object Lock feature but it is disabled by your current license, contact your Clouddian representative to inquire about obtaining a different license.
- **You must enable the HyperStore Shell and disable the *root* account password** on your HyperStore hosts. For instructions see **"Enabling the HSH and Managing HSH Users"** (page 90). This must be completed before Object Lock enabled buckets can be created.

#### 4.5.7.2.2. 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.

**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 ***PUT Bucket*** request that includes the request header ***x-amz-object-lock-enabled: true***. For HyperStore support of the S3 ***PUT Bucket*** operation, see [PUT Bucket](#).
- **With the CMC**, as a user creates a new bucket she can choose to enable Object Lock for the bucket. For more information see **"Add a Bucket"** (page 218).

Enabling Object Lock on a 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.

**Note** Enabling Object Lock on a new bucket automatically enables Versioning on that bucket as well. Object Lock can only be used in combination with Versioning (which protects objects from being over-written).

**Note** If your HyperStore system does not meet the [prerequisites for Object Lock](#), then if an S3 application user submits a ***PUT Bucket*** request that includes the request header ***x-amz-object-lock-enabled: true***, the request will fail and a 403 error response will be returned to the client application. If the CMC is the client application, the error that the CMC will display in this event is "Unable to perform operation. Access is denied."

#### 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 (S3 API or CMC)**" (page 126). 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 **PUT Bucket object lock configuration** request to the HyperStore S3 Service. For HyperStore support of this S3 API operation see [PUT Bucket object lock configuration](#). 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 privileged users to change the object retention period or delete objects before their retention period completes) or **Compliance** mode (which does not allow any user to change the object retention period or delete objects before their retention period completes).
- **With the CMC**, the bucket owner can use the Object Lock tab of the bucket Properties page to set a default Object Lock configuration on the bucket. For more information see "**Configure Object Lock Properties for a Bucket**" (page 242).

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 is locked, HyperStore rejects an S3 *DELETE Object Version* request for that object with an HTTP 403 Forbidden response. The exception is if Governance mode is being used and the requesting user is a privileged user; for more information see "**Object Protection Under Governance Retention, Compliance Retention, and Legal Hold**" (page 128).

The *PUT Bucket object lock configuration* request can be used again to change or remove the default Object Lock configuration on a bucket. 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 *DELETE Object* request -- as opposed to an S3 *DELETE Object Version* request -- 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 **GET Bucket object lock**

**configuration** request to the HyperStore S3 Service. For HyperStore support of this S3 operation see [GET Bucket object lock configuration](#). 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 Properties page for the bucket and check the Object Lock tab to see what the default Object Lock configuration is for the bucket (if any).

#### 4.5.7.2.3. Setting Object Lock Attributes on Individual Objects (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** that supports the S3 APIs for object locking, 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.

#### 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:

- *PUT Object*
- *PUT Object Copy*
- *POST Object*
- *Initiate Multipart Upload*

For HyperStore support of these S3 operations see [PUT Object](#), [PUT Object Copy](#), [POST Object](#), and [Initiate Multipart Upload](#).

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, for an indefinite period of time until the Legal Hold is explicitly removed from the object (by a privileged user). 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.

**Note** Object upload requests that include Object Lock headers must use Signature Version 4 request authentication. This is consistent with Amazon's Object Lock requirements.

### 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 **PUT Object retention** and/or **PUT Object legal hold**. For HyperStore support of these S3 operations see [PUT Object retention](#) and [PUT Object legal hold](#).

**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 **PUT Object retention** 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 being used and the requesting user is a privileged user, as described in "Object Protection Under Governance Retention, Compliance Retention, and Legal Hold" (page 128)).

- **With the CMC**, the bucket owner can assign Object Lock attributes to an existing object in the bucket by accessing the object's Properties page and using the Object Lock tab. For more information see "Set Object Lock Attributes on an Object" (page 255).

### 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 **GET Object**, **HEAD Object**, **GET Object retention**, and/or **GET Object legal hold**. For HyperStore support of these S3 operations see [GET Object](#), [HEAD Object](#), [GET Object retention](#), and [GET Object legal hold](#).

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 Properties page and check the Object Lock tab to see what the lock attributes are for that object version.

#### 4.5.7.2.4. Object Protection Under Governance Retention, Compliance Retention, and Legal Hold

The table below shows the key differences between Governance mode retention, Compliance mode retention, and Legal Hold in terms of the protection that they provide to locked objects. 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 both <code>s3:DeleteObjectVersion</code> and <code>s3:BypassGovernanceRetention</code> permission can use the <i>DELETE Object Version</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 both <code>s3:PutObjectRetention</code> and <code>s3:BypassGovernanceRetention</code> permission can use the <i>PUT Object retention</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 with <code>s3:PutObjectLegalHold</code> permission can use the <i>PUT Object legal hold</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 both <code>s3:PutObjectRetention</code> and <code>s3:BypassGovernanceRetention</code> permission can use the <i>PUT Object retention</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	Not applicable

**Note** In the CMC, only the bucket owner can access and manage the objects in his or her bucket. (Depending on your system configuration, the CMC may also allow system administrators to access a bucket and its objects on behalf of the bucket owner. By default configuration, system administrator access to users' buckets is not allowed.)

## 4.6. User Provisioning and LDAP Integration

### 4.6.1. User Provisioning and LDAP Integration 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 once one or more groups exist 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 131).
- 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.)
- 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. 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 (in the [mts-ui.properties.erb](#) file, see the `admin.manage_users.enabled` property and those that follow it). Individual CMC UI functions and sub-functions can be displayed to or hidden from group administrators depending on your configuration settings.

#### 4.6.1.1. IAM Support

HyperStore provides limited support for the Amazon Identity and Access Management (IAM) API. For an overview of this HyperStore feature -- including support for creating IAM groups and users -- see **"HyperStore Support for the AWS IAM API"** (page 991).

#### 4.6.1.2. SAML Support

HyperStore supports Security Assertion Markup Language (SAML) based access to S3 storage resources. For more information see **"SAML Support"** (page 1032).

### 4.6.2. 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 131).

#### Provisioning Groups (CMC)

In the CMC's **Manage Groups** page you can perform group operations including:

- **"Add a Group"** (page 271)
- **"Set Quality of Service (QoS) Controls"** (page 285)
- **"Retrieve a Group or a List of Groups"** (page 275)
- **"Edit a Group"** (page 276) (including suspending a group)
- **"Delete a Group"** (page 277)

#### 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](#)

### 4.6.3. Provisioning Users

You can provision individual users through the CMC, or the Admin API, or by enabling LDAP integration.

**Note** The HyperStore system does not currently support bulk provisioning of users. Users must be added one at a time.

## Provisioning Users (CMC)

In the CMC's **Manage Users** page you can perform user operations including:

- **"Add a User"** (page 263)
- **"Set Quality of Service (QoS) Controls"** (page 285)
- **"Retrieve a User or List of Users"** (page 265)
- **"Edit or Suspend a User"** (page 266) (including suspending a user)
- **"Delete a User"** (page 270)

**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](#).

## 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](#)

## Provisioning Users (LDAP / Active Directory)

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 131).

### 4.6.4. LDAP Integration

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Enabling LDAP Authentication for a Group"** (page 132)
- **"Provisioning of Users within LDAP-Enabled Groups"** (page 132)
- **"Deleting Users from the CMC and/or LDAP"** (page 133)
- **"Disabling LDAP Authentication After Having Used It"** (page 133)
- **"Special Considerations for LDAP Authentication of System Administrators"** (page 134)

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 supports 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.

**Note** 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.

#### 4.6.4.1. 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 *IdapEnabled* attribute to true and also set the other LDAP-related attributes.

**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 "**Special Considerations for LDAP Authentication of System Administrators**" (page 134).

#### 4.6.4.2. 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.

#### 4.6.4.3. 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).

#### 4.6.4.4. 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.

#### 4.6.4.5. Special Considerations for LDAP Authentication of System Administrators

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 132)).

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 configuration step:
    1. Log in to the Puppet Master 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
```

For more information on the HyperStore Shell see:

- **"Security Features"** (page 89)
- **"Enabling the HSH and Managing HSH Users"** (page 90)
- **"Using the HSH"** (page 94)

**Note** If any system admin users were set up for LDAP authentication while you were running **HyperStore 7.2.2** (the first version to support LDAP for system admin users), for those users

their HyperStore Shell password is a local, hard-coded version of the LDAP password that they used when they were first provisioned into HyperStore, and LDAP authentication is **not** performed when they log into the HyperStore Shell. LDAP authentication for HyperStore Shell users works only for system admin users created in HyperStore version 7.2.3 and later.

## 4.7. Quality of Service Controls

### 4.7.1. Quality of Service (QoS) Feature Overview

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Service Usage Types Subject to QoS Controls"** (page 135)
- **"QoS Assignment Granularity"** (page 136)

The Clouddian 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.

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.

#### 4.7.1.1. Service Usage Types Subject to QoS Controls

Several types of service usage metrics can be configured for QoS controls:

- Storage quota, by number of KBs.
- 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 KBs per minute.
- Peak data download rate, in KBs per minute.

When configuring QoS controls, you have the option of limiting some of the usage types above while leaving others unrestricted. For example, you could limit per-user and/or per-group storage volume (by KBs), while placing no restrictions on number of stored objects. Similarly, 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.

For HTTP request rate and for upload and download rates, 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 log an INFO 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.)

**Note** The **storage overhead associated with replication or erasure coding does not count toward a user's storage quota**. For example, a 1MB object that is protected by 3X replication or by 4+2 erasure coding counts as only 1MB 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 179).

#### 4.7.1.2. 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 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, the system also provides you the ability to configure different QoS settings for different regions. For example, you might configure default user QoS settings that allow users 20GB of storage in your "North" service region and 30GB in your "South" service region.

### 4.7.2. 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 that 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 enabling in 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 **"Setting QoS Limits for Users"** (page 137) and **"Setting QoS Limits for Groups"** (page 137).

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 objects controls to unlimited and the rate controls to your desired levels.

### 4.7.3. Setting QoS Limits for Groups

As with user QoS settings, you can set group QoS settings through either the CMC or the Admin API. Group QoS settings limit the **aggregate activity** of all users in a group.

#### Setting QoS Limits for Groups (CMC)

First you should decide whether you want to set default QoS limits for all groups in your HyperStore system. If so, you can do this by going to the CMC's [Manage Groups](#) page and clicking **Group QoS Default** to open the **Group QoS Limits: Defaults** panel. Here you can configure default group QoS limits for your whole system.

Next, you can set group QoS limits for a specific group. If you do so, these limits will override any system defaults for group QoS. To do this, 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.

**Note** For details about working with the CMC's QoS configuration panels see "**Set Quality of Service (QoS) Controls**" (page 285).

#### Setting QoS Limits for Groups (Admin API)

To establish group QoS settings through the Admin API, you use the same method that you do for user QoS settings: [POST /qos/limits](#). URI parameters enable you to specify that you're creating system defaults for group QoS settings, or settings for a particular group.

The [GET /qos/limits](#) and [DELETE /qos/limits](#) methods can be used to retrieve or delete group QoS settings.

### 4.7.4. Setting QoS Limits for Users

By system 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](#), service users still are not subject to QoS controls until you establish specific QoS limits for the various service usage types.

You can set QoS limits for users through either the CMC or the Admin API.

## Setting QoS Limits for Users (CMC)

First you should decide whether you want to set default QoS limits for all users of your HyperStore system. If so, you can do this by going to the CMC's [Manage Users](#) page and clicking **User QoS Default** to open the **User QoS Limits: Defaults** panel, where you can configure default user QoS limits for your system.

Next, decide whether you want to set default QoS limits for all users who belong to a particular user group. If you do so, this will override the system-wide default, for users within that group. To do this, 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.

Finally, you also have the option of setting QoS limits for a specific individual user. If you do so, these limits will override any system or group default limits, for that user. To do this, 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.

**Note** For details about working with the CMC's QoS configuration panels see **"Set Quality of Service (QoS) Controls"** (page 285).

## Setting QoS Limits for Users (Admin API)

With the Admin API method [POST /qos/limits](#) you can set QoS limits for HyperStore service users. With the method's URI parameters you can specify whether you're setting default limits for all users in the system; or default limits for all users within a specified group; or limits for a specific user. URI parameters also enable you to set the numerical limits for each service usage type — for example, a Storage Bytes limit of 10GB.

The Admin API also supports:

- A [GET /qos/limits](#) method, for retrieving system-default, group-default, or user-specific QoS limits
- A [DELETE /qos/limits](#) method, for deleting system-default, group-default, or user-specific QoS limits

## 4.8. Usage Reporting and Billing

### 4.8.1. Usage Reporting and Billing Feature Overview

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Bucket Usage Statistics"** (page 139)
- **"How Usage Data is Calculated, Tracked, and Processed"** (page 139)
- **"Protection of Usage Data Through Replication"** (page 141)
- **"S3 Request Traffic Analysis and Visualization"** (page 141)
- **"Billing"** (page 141)

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:

- 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.

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.

#### 4.8.1.1. 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.

In the current release, bucket statistics are not available through the CMC. Bucket statistics are supported only by Admin API calls.

The bucket statistics feature is **disabled by default**. For information on enabling this feature see **"Enabling Advanced Usage Reporting Features"** (page 142).

**Note** The Admin API call that returns the current total number of bytes and total number of objects in a bucket -- `POST /usage/repair/bucket` -- works even if bucket statistics are disabled in the system. The other bucket statistics related API calls only work if bucket statistics are enabled. For more information on the API calls see **"usage"** (page 868).

#### 4.8.1.2. 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
<ul style="list-style-type: none"> <li>• Storage Bytes</li> <li>• Storage Objects</li> </ul>	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

Usage Data Types	System Handling
	<p>system includes the object name size and object metadata size as well as the size of the object itself.</p> <p>Note that storage <b>overhead associated with replication or erasure coding does not count</b> 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.</p> <p>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 <a href="#">Quality of Service</a> configurations.</p> <p>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.</p> <p>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:</p> <pre> 10MB X 20/60 = 3.33MB + 15MB X 40/60 = 10 MB = 13.33MB weighted average for hour </pre> <p>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.</p> <div> <p><b>Note</b> 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 <a href="#">system cron jobs</a>.</p> </div>
<ul style="list-style-type: none"> <li>• HTTP Requests</li> <li>• Bytes IN</li> <li>• Bytes OUT</li> </ul>	<p>By default system configuration, HTTP Requests, Bytes IN, and Bytes OUT are not tracked.</p> <p>If you <a href="#">enable usage tracking for HTTP Requests and Bytes IN/OUT</a>, 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.</p> <p>In recording the Bytes IN and Bytes OUT impacts of a given S3 trans-</p>

Usage Data Types	System Handling
	<p>action, 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).</p> <p>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.</p> <p>The hourly roll-up data is rolled up daily; and the daily roll-up data is rolled up monthly.</p> <p>If you fully enable HyperStore <a href="#">Quality of Service</a> 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.</p>

#### 4.8.1.3. 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 install you choose how many service metadata replicas to keep in each data center.

#### 4.8.1.4. S3 Request Traffic Analysis and Visualization

For information on setting up an [Elastic Stack](#) node and streaming request log data to that node to support analysis and visualization of your S3 request traffic, see **"Setting Up Elastic Stack for S3 Request Traffic Analysis"** (page 631).

#### 4.8.1.5. Billing

The HyperStore system maintains comprehensive service usage data for each group and each user in the system. This usage data 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 period. 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.

Cloudian 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 "whitelisted" domains.

**Note** For information on how auto-tiering impacts billing calculations, see **"How Auto-Tiering Impacts Usage Tracking, QoS, and Billing"** (page 179).

#### 4.8.1.5.1. Retention of Usage Data Used for Billing

Billing calculation is derived from hourly rollup usage data. The retention period for 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 144)

#### 4.8.1.5.2. 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](#).

### 4.8.2. Enabling Advanced Usage Reporting Features

This topic describes how to enable advanced HyperStore usage reporting features that are disabled by default: per-user and per-group traffic rates (HTTP request rates and data transfer rates); and per-bucket usage tracking.

#### 4.8.2.1. 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 **Cluster** → **Cluster Config** → **Configuration Settings**, 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 Cassandra, and additional work for the cron jobs that roll up usage data into hourly, daily, and monthly aggregates.

#### 4.8.2.2. 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, in the configuration file [common.csv](#) set `bucketstats_enabled` to "true".

Then push your change out to the cluster and restart the S3 Service. For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

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. 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 data being stored in Cassandra, and additional work for the cron jobs that roll up usage data into hourly, daily, and monthly aggregates.

**Note** The Admin API call that returns the current total number of bytes and total number of objects in a bucket -- `POST /usage/repair/bucket` -- works even if bucket statistics are disabled in the system. The other bucket statistics related API calls only work if bucket statistics are enabled. For more information on the API calls see **"usage"** (page 868).

### 4.8.3. Validating Storage Usage Data

As described in **"Usage Reporting and Billing Feature Overview"** (page 138), 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 keyspace, 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.

#### 4.8.3.1. 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 a configurable maximum number of users (`mts.properties.erb: "usage.repair.maxdirtyusers"` (page 569); default = 1000) 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" keyspace's "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.

#### 4.8.3.2. 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.

#### 4.8.4. Setting Usage Data Retention Periods

Data retention periods are separately configurable for raw, hourly roll-up, daily roll-up, and monthly roll-up usage data. After data has been stored for its configured retention period (also known as its "time-to-live" or TTL), it's automatically deleted from the system.

The relevant settings are all in the configuration template [mts.properties.erb](#):

- **"reports.rolluphour.ttl"** (page 567) (default = 5616000 seconds [65 days])

**IMPORTANT !** The HyperStore system calculates monthly bills for service users by aggregating **hourly** roll-up data. Once hourly data is deleted, you will not be able to generate bills for the service period covered by that data. So be sure to have *reports.rolluphour.ttl* set to a value large enough to accommodate your billing routine.

- **"reports.rollupday.ttl"** (page 568) (default = 5616000 seconds [65 days])
- **"reports.rollupmonth.ttl"** (page 568) (default = 15552000 seconds [180 days])

If you edit any of these settings, push your changes out to the cluster and restart the S3 Service. For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

#### 4.8.5. Generating a Usage Report

You can generate usage report data through either the CMC or the Admin API.

##### 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.

##### 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 a [POST /usage/repair/bucket](#) call that returns the current total bytes count and total objects count for a bucket; and a [POST /usage/bucket](#) call that can retrieve raw usage data for multiple specified buckets at once.

**Note** The Admin API call that returns the current total number of bytes and total number of objects in a bucket -- [POST /usage/repair/bucket](#) -- works even if bucket statistics are disabled in the system. The other bucket statistics related API calls only work if bucket statistics are enabled.

**Note** For information on setting up an [Elastic Stack](#) node and streaming request log data to that node to support analysis and visualization of your S3 request traffic, see "**Setting Up Elastic Stack for S3 Request Traffic Analysis**" (page 631).

#### 4.8.6. Creating Rating Plans for Billing

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 GB of data in storage (based on a calculated average storage volume for the billing period)
- Per GB of data uploaded
- Per GB 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 GB of stored data, per GB 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](#)

## 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.

### 4.8.6.0.1. Storage Charges

- Types: One type only. The average number of GBs of data stored for the month.
- Example:
  - Unit: Dollars per GB-months.
  - Pricing: From 0-1 TB at \$0.14 per GB-month, 1-10 TB at \$0.12 per GB-month, 10+ TB at \$0.10 per GB-month.
  - Usage: Store 0.1 TB for first 10 days of month, then 20 TB for remaining 21 days of month.
  - Sum up usage over month:  $0.1\text{TB} \times 240 \text{ hours} + 20\text{TB} \times 504 \text{ hours} = 10,104 \text{ TB-hours}$ .
  - Convert to GB-months:  $10,104 \text{ TB-hours} \times (1024 \text{ GB/TB}) \times (1 \text{ month}/744 \text{ hours}) = 13,906.58 \text{ GB-months}$
  - Apply tiered pricing:  $(\$0.14 \times 1 \times 1024\text{GB}) + (\$0.12 \times 9 \times 1024\text{GB}) + (\$0.10 \times 3666.58\text{GB}) = \$1615.94$ .

### 4.8.6.0.2. Data Transfer Charges

- Types: 2 types. Data Transfer IN and Data Transfer OUT.
  - Each type (IN, OUT) has own cost.
- Example:
  - Unit: Dollars per GB.
  - Pricing: IN: 0GB+ at \$0.10. OUT: 0-10GB at \$0.20, 10GB+ at \$0.10
  - Usage: Transfer-IN 300GB, Transfer-OUT 100GB.
  - $(300 \times \$0.10) + (10 \times \$0.20 + 90 \times \$0.10) = \$41.00$ .

#### 4.8.6.0.3. 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.
  - $(\$0.20 \times 25,000 / 10,000) + (\$0.01 \times 300,000 / 10,000) + (\$0.00 \times 1000 / 10,000) = \$0.53$ .

#### 4.8.6.0.4. Total Bill for Month

- Storage charges: \$1615.94
- Data transfer charges: \$41.00
- Requests charges: \$0.53
- TOTAL: \$1657.47

### 4.8.7. Assigning Rating Plans to 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 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 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.

#### 4.8.8. Creating a "Whitelist" for Free Traffic

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 a "whitelist", 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 whitelist 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 a whitelisted IP address.

You can create a billing whitelist through either the CMC or the Admin API. But before doing so, you must enable the whitelist feature. It is disabled by default.

**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 whitelist 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 Whitelist Feature

To enable the HyperStore billing whitelist feature:

1. On your Puppet master node, open the following configuration file in a text editor:

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

2. Set `admin_whitelist_enabled` to `true`, then save your change.
3. Use the installer to push your changes to the cluster and to restart the S3 Service and the CMC. For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

#### Creating a Source IP "Whitelist" (CMC)

To create a whitelist through the CMC, use the [Whitelist](#) page. That page shows the ID and name of the default whitelist, which initially has no IP addresses in it. To create a functioning whitelist, you edit the default whitelist by adding IP addresses or subnets to it.

Once created, the whitelist takes effect. From that time forward, the HyperStore billing system will no longer apply traffic charges to users' traffic originating from the whitelisted IP addresses and subnets.

**Note** If you want to see a particular user's whitelisted traffic volume during a given billing period, you can do so as part of the HyperStore system's functionality for **"Generating Billing Data for a User or Group"** (page 149).

### Creating a Source IP "Whitelist" (Admin API)

The HyperStore Admin API provides two different methods for creating a whitelist:

- Use the [POST /whitelist](#) method to post your whitelist as a JSON-encoded request payload.
- Use the [POST /whitelist/list](#) method to post your whitelist as a URI parameter.

The Admin API also supports a [GET /whitelist](#) method for retrieving the contents of your current whitelist.

### 4.8.9. 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 *mts.properties.erb*: **"reports.rolluphour.ttl"** (page 567). The default retention period is 65 days. Once this rollup data is deleted it can no longer be used to generate users' bills.

### 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 [whitelisted 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.

## 4.9. Automated Data Repair

### 4.9.1. Automated Data Repair Feature Overview

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Repair-On-Read"** (page 150)
- **"Proactive Repair"** (page 151)
- **"Scheduled Auto-Repair"** (page 151)
- **"Operator-Initiated Data Repair"** (page 152)

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.

#### 4.9.1.1. 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 (mismatched with the MD5 hash in the [file digest](#)) replicas are automatically replaced or updated. This repair process occurs whether the read succeeds in meeting [consistency requirements](#) or not, so long as at least one valid replica exists in the system.

Repair-on-read is also performed for erasure coded object reads, in the event that 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 5 fragments for the object, the system replaces the object's missing fragment.

**Note** In the case of large objects (over 10MB) that the system has automatically divided into "chunks" before replication or erasure coding is applied, when processing an object read request the system reads an object's chunks sequentially. If a given chunk is unreadable -- that is, if read consistency

requirements cannot be met for that chunk -- then the system does not attempt to read the object's remaining chunks (if any). In this case, repair-on-read is applied only to the chunks that the system read or tried to read, and not to any remaining chunks that the system did not try to read.

#### 4.9.1.2. 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 Cassandra 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 152) .

#### 4.9.1.3. 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, each node is scheduled to have [hsstool repair](#) automatically run on it once every 30 days.
- To repair erasure coded object data, each node is scheduled to have [hsstool repairec](#) automatically run on it once every 29 days.
- To repair metadata in Cassandra, each node is scheduled to have [hsstool repaircassandra](#) automatically run on it once every 7 days.

The repairs are scheduled and launched in such a way that **within a service region only one repair of each type is running at a time**. For each repair type the system maintains a queue of nodes scheduled for auto-repair -- a replicated data repair queue, an erasure coded data repair queue, and a Cassandra metadata repair queue. For each repair type, repair of the node that is next in queue will not start until the repair operation completes on the node on which a repair is currently running. Consequently if you have a lot of nodes and/or a high volume of data in your system, the actual time between repairs of a given node may be larger

than the scheduled interval. This effect is most pronounced with erasure coded data repair (which can be very long-running) and least pronounced with Cassandra repair (which is relatively fast).

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. In a multi- data center HyperStore service region, the erasure coded data auto-repair queue is ordered in such a way that the target nodes alternate among the data centers -- for example after a repair completes on a target node in DC1, then the next target node will be from DC2, and then after that completes the next target node will be from DC3, and then after that completes the next target node will be from DC1 again, and so on.

**Note** The intervals for scheduled auto-repair are configurable as described in **"Configuring Automatic Data Repair"** (page 152).

**Note** The system allows repair operations of different types -- such as an *hsstool repair* operation and an *hsstool repairec* operation -- to run concurrently within the same service region.

#### 4.9.1.4. 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 443).
- 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 453).

For repair command syntax see [hsstool repair](#) and [hsstool repairec](#) and [hsstool repaircassandra](#).

#### 4.9.2. 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)

See **"Auto-Repair Schedule Settings"** (page 350) for important details about how these interval settings are applied.

**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 **"auto\_repair\_computedigest\_run\_number"** (page 518) setting in *common.csv*. By default "computedigest" is not used in auto-repair runs.

The [Proactive Repair](#) feature is configurable by these properties:

- In [hyperstore-server.properties.erb](#) the setting **"hyperstore.proactiverepair.poll\_time"** (page 550) 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 proactiverepairq](#) command with the "-start" option.

- In [mts.properties.erb](#) the setting **"hyperstore.proactiverepair.queue.max.time"** (page 579) 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 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 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 453).

If you edit properties file settings, be sure to push your changes to your cluster and to restart the HyperStore Service (for a *hyperstore-properties.erb* edit) and/or the S3 Service (for an *mts.properties.erb* edit). For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

### 4.9.3. 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).

#### 4.9.3.1. 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.

#### 4.9.4. Disabling or Stopping Data Repairs

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Temporarily Disabling Automatic Data Repairs"** (page 154)
- **"Stopping In-Progress Data Repairs"** (page 156)

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.

##### 4.9.4.1. 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.

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', 'Repair Status', and 'Operation Status'. The 'Advanced' tab (highlighted with a red box and '3') is selected. The 'Command Type' dropdown is set to 'Maintenance'. The 'hstool Command' dropdown is set to 'autorepair'. The 'Target Node' dropdown is set to 'hyperstore-demo1'. The 'Options' section shows 'Enable' and 'Disable' radio buttons, with 'Disable' selected. The 'Type' dropdown is set to 'Replicas'. The '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 156).

After completing the system operation that you are undertaking, be sure to **re-enable automatic data repairs**.

This screenshot is identical to the one above, but the 'Options' section shows 'Enable' selected instead of 'Disable'. The 'EXECUTE' button is at the bottom right.

#### 4.9.4.1.1. 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' (1), 'Alerts', 'Admin', and 'Help'. The main navigation bar has 'Data Centers', 'Nodes' (2), 'Cluster Config', 'Storage Policies', 'Repair Status', and 'Operation Status'. The 'Nodes' page has tabs for 'Node Status', 'Node Activity', and 'Advanced' (3). In the 'Advanced' tab, the 'Command Type' is set to 'Maintenance'. The 'hsstool Command' is set to 'proactiverepair'. The 'Target Node' is set to 'hyperstore-demo1'. Under 'Options', the 'Disable' radio button is selected. A description at the bottom states: '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.

Be sure later to **re-enable whichever type of automatic data repair you disabled**, by using the same interface.

#### 4.9.4.2. 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 CMC's [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.

## 4.10. Automated Disk Management

### 4.10.1. Automated Disk Management Feature Overview

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Automatic Correction of Substantial Disk Usage Imbalance"** (page 157)
- **"Automatic Stop of Writes to a Disk at 90% Usage"** (page 157)
- **"Automatic Stop of Writes to a Node at 90% Usage"** (page 158)
- **"Dynamic Object Routing"** (page 160)
- **"Automatic Disk Failure Handling"** (page 160)

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.

#### 4.10.1.1. 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 160).

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.

#### 4.10.1.2. 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 system moves a 90% full disk's tokens in a manner

that takes into account the current usage levels among the remaining disks on the node: the lower a disk's current usage level, the more likely it is to receive some of the transferred tokens.

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 back below 90% it becomes eligible to receive tokens from any other disks on the node that subsequently reach 90% utilization and go into a stop-write condition. Recall though that when the system moves tokens away from a 90% full disk it prioritizes allotting tokens to disks that have relatively low usage. So a disk that is only a little below 90% usage is less likely to receive tokens during this process than disks that are at lower usage levels on the node.

**Note** When a disk at 90% usage level enters stop-write condition it stops receiving new S3 writes, but object data writes associated with [hsstool repair](#), [hsstool repairec](#), [hsstool rebalance](#), and [hsstool decommission](#) operations continue to be supported until the disk reaches 95% usage. If the disk reaches 95% usage, then subsequent writes that *hsstool* operations target to that disk will fail (and in the operation's status metrics these failures will increment the "Failed count").

**Note** If you want to customize the disk usage check interval (default 30 minutes) or the "stop-write" threshold (default 90%) or the *hsstool* operation write threshold (default 95%) or the "start-write" threshold (default 85%), consult with Clouddian Support.

#### 4.10.1.3. 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](#); 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 71).

#### 4.10.1.3.1. 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 (see "[bucketops](#)" (page 762)). 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 420)). 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.

#### 4.10.1.4. 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.

#### 4.10.1.5. Automatic Disk Failure Handling

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").
- At a configurable interval (default is 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 errors regarding a drive occur in excess of a configurable error rate threshold, 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. For more detail on the configuration options and the disk disabling behaviors see **"HyperStore Disk Failure Action"** (page 342).

**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.

**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 Clouidian 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.

#### 4.10.1.5.1. 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.

### 4.10.2. 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. Both of these factors are configurable.

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 or the delta that triggers disk migration are:

- **"hyperstore\_disk\_check\_interval"** (page 523) in [common.csv](#) (default = 72 hours)
- **"disk.balance.delta"** (page 552) in [hyperstore-server.properties.erb](#) (default = 10 percent)

After editing either of these settings, be sure to push your changes to the cluster and restart the HyperStore Service. For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

**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 Cloudian Support. These settings are not in HyperStore configuration files by default.

### 4.10.3. 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 the [configured maximum delta](#) (10% by default), then tokens will automatically be migrated between disks.

### 4.10.4. Configuring Disk Failure Handling

**IMPORTANT !** The automatic disk failure handling feature does not work correctly in Xen, Logical Volume Manager (LVM), or Amazon EC2 environments. Contact Cloudian Support if you are using any of these technologies.

Several aspects of the HyperStore automated disk failure handling feature are configurable.

In the CMC's [Configuration Settings](#) page, in the "System Settings" section, you can configure a **"HyperStore Disk Failure Action"** (page 342). This is the automated action for the system to take in the event of a detected disk failure, and the options are "Disable Disk + Move Its Tokens" (the default) or "None". If you change this setting in the **Configuration Settings** page, your change is dynamically applied to the cluster — no service restart is necessary.

Additional settings are available in [hyperstore-server.properties.erb](#) on your Puppet master node. These include settings for establishing an error rate threshold for disk-related errors in the HyperStore Service application log (which if exceeded for a given disk will result in the automatic triggering of the Disk Failure Action):

- **"disk.fail.error.count.threshold"** (page 551)
- **"disk.fail.error.time.threshold"** (page 552)

By default the threshold is 10 errors in the space of 5 minutes, for a particular disk.

Also in *hyperstore-server.properties.erb* is this setting for the interval at which to conduct the proactive disk drive audit:

- **"disk.audit.interval"** (page 552)

By default the proactive audit occurs hourly.

After editing any of these *hyperstore-server.properties.erb* settings, push the changes to the cluster and restart the HyperStore Service. For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

**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 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.

### 4.10.5. 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.

### 4.10.6. 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 the [configurable threshold](#).

### 4.10.7. 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 480).

- **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 482).

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 160)).

**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.

## 4.11. Object Metadata

### 4.11.1. Object Metadata Feature Overview

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"System-Defined Object Metadata"** (page 164)
- **"User-Defined Object Metadata Based on x-amz-meta-\* headers"** (page 165)
- **"User-Defined Object Tagging"** (page 165)
- **"Object Metadata and Storage Policies"** (page 165)
- **"Object Metadata, Cross-Region Replication, and Auto-Tiering"** (page 165)

Like Amazon S3, HyperStore allows for rich metadata to be associated with each stored object. There are three categories of object metadata:

- The system itself assigns certain metadata items to objects, having to do with object attributes and status
- Optionally, users can assign metadata to objects by using the S3 *x-amz-meta-\** request headers
- Optionally, users can assign key-value "tags" to objects by using the S3 API methods and headers that implement object tagging

#### 4.11.1.1. System-Defined Object Metadata

For each object, HyperStore maintains a variety of system-defined object metadata including (but not limited to) the following:

- Creation time
- Last modified time
- Last accessed time
- Size
- ACL information
- Version, if applicable
- Public URL, if applicable
- Compression type, if applicable
- Encryption key, if applicable
- Auto-tiering state, if applicable

#### 4.11.1.2. User-Defined Object Metadata Based on `x-amz-meta-*` headers

S3 client applications can create user-defined object metadata by the use of `x-amz-meta-*` request headers that accompany the upload of an object. The client application sets the specific header names and corresponding values (such as `x-amz-meta-project: apollo` or `x-amz-meta-author: ozowa`). This is a standard S3 API feature that is supported by HyperStore's S3 Service. For more information see **"Creating Object Metadata and Tags"** (page 166).

By default the maximum size limit for `x-amz-meta-*` based metadata is 2KB per object. It is possible to increase this size limit by using hidden configuration settings, but to do so could have significant implications for the storage space requirements on the SSDs on which you store Cassandra data. If you are interested in increasing the size limit for `x-amz-meta-*` based object metadata, consult with Clodian Support.

#### 4.11.1.3. User-Defined Object Tagging

S3 client applications can create user-defined object tags -- key-value pairs such as `status=complete` or `confidential=true` -- either as they upload new objects or for objects that are already stored in the system. This is implemented by the use of object tagging request headers (in the case of assigning tags to an object as it's being uploaded) or object tagging API methods (in the case of assigning tags to an object that's already in storage). This is a standard S3 API feature that is supported by HyperStore's S3 Service. For more information see **"Creating Object Metadata and Tags"** (page 166).

Object tags share some common use cases with `x-amz-meta-*` based object metadata -- for example, you could use either a `x-amz-meta-*` header or an object tag to identify an object as belonging to a particular project. But object tags support a wider range of capabilities, including being able to assign tags to an object that's already in storage, and being able to use object tags as criteria in access control policies. (A future release of HyperStore will also support using object tags as criteria for bucket lifecycle policies for auto-tiering or auto-expiration).

An object may have a maximum of 10 tags associated with it, and each tag must have a unique key (for example you can't assign `status:in-progress` and `status:complete` to the same object). Each key can be a maximum of 128 unicode characters in length, and each value can be up to 256 unicode characters.

#### 4.11.1.4. Object Metadata and Storage Policies

HyperStore object metadata -- including object tags -- is stored in Cassandra, and is protected by replication. The degree of replication depends on the type of [storage policy](#) being used. For more information see **"Object Metadata Replication"** (page 80).

#### 4.11.1.5. Object Metadata, Cross-Region Replication, and Auto-Tiering

When the [cross-region replication](#) feature replicates objects from a bucket in one service region to a bucket in a different service region in the same HyperStore system, user-defined metadata in the form of `x-amz-meta-*` headers and object tags is 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 is sent along with the object data if the tiering destination is an S3-compliant system such as Amazon Web Services or Google Cloud Storage. Also, the user-defined object metadata is retained in the local HyperStore system as well.

If the tiering destination is not S3-compliant, such as Microsoft Azure or Spectra Pearl Logic, then the user-defined object metadata is **not** sent along with the object data. The user-defined object metadata will exist only in the HyperStore system, even after the corresponding object data is transitioned to the destination system.

**IMPORTANT !** Even if the destination system is S3-compliant, the auto-tiering feature does not send certain types of object metadata such as ACL data (data regarding access permissions on objects).

### 4.11.2. Creating Object Metadata and Tags

S3 client applications can create user-defined object metadata -- *x-amz-meta-\** based object metadata and/or object tags -- by using standard S3 API methods and headers that the HyperStore S3 Service supports.

**Note** In the current version of HyperStore, the CMC's built-in S3 client does not support creating user-defined object metadata. To create user-defined object metadata you will need to use an S3 client application other than the CMC.

#### 4.11.2.1. Creating x-amz-meta-\* Based Object Metadata

When uploading a new object, S3 clients can create user-defined object metadata by including one or more *x-amz-meta-\** request headers along with the *PUT Object* request. For example, *x-amz-meta-topic: merger* or *x-amz-meta-status: draft*. For HyperStore support of the *PUT Object* method, see [PUT Object](#).

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. For HyperStore support of the *POST Object* method, see [POST Object](#).

#### 4.11.2.2. Creating Object Tags

When uploading a new object, S3 clients can create one or more user-defined object tags by including a single *x-amz-tagging* request header along with the *PUT Object* request. For example, *x-amz-tagging: team-m=Marketing&project=SEO-2018&status=Needs-Review*. For HyperStore support of the *PUT Object* method, see [PUT Object](#).

S3 clients can also create object tags when calling the *POST Object* method, by using the *tagging* form field. For HyperStore support of the *POST Object* method, see [POST Object](#).

The *PUT Object - Copy* 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. For HyperStore support of the *PUT Object -- Copy* method, see [PUT Object - copy](#).

S3 clients can also create object tags for an object that is already stored in the HyperStore system, by using the S3 API method *PUT Object tagging*. For HyperStore support of this API method, see [PUT Object tagging](#).

For a high-level view of object tagging usage and methods, in the Amazon S3 online documentation see [Object Tagging](#).

### 4.11.3. Retrieving Object Metadata and Tags

S3 client applications can retrieve user-defined object metadata -- *x-amz-meta-\** based object metadata and/or object tags -- by using standard S3 API methods and headers that the HyperStore S3 Service supports.

**Note** In the current version of HyperStore, the CMC's built-in S3 client does not support retrieving user-defined object metadata. To retrieve user-defined object metadata you will need to use an S3 client application other than the CMC.

#### 4.11.3.1. Retrieving *x-amz-meta-\** Based 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:

- *GET Object* returns this type of object metadata as response headers, as well as returning the object itself. For HyperStore support of this API method see [GET Object](#).
- *HEAD Object* returns this type of object metadata as response headers, without returning the object itself. For HyperStore support of this API method see [HEAD Object](#).

#### 4.11.3.2. Retrieving Object Tags

The S3 methods *GET Object* and *Head Object* will not return object tags. They will however return a count of the object tags associated with the object (if any), in an *x-amz-tagging-count* response header. For HyperStore support of these API methods see [GET Object](#) and [HEAD Object](#).

To retrieve the object tags associated with an object, use the S3 API method *GET Object tagging*. For HyperStore support of this API method see [GET Object tagging](#).

For a high-level view of object tagging usage and methods, in the Amazon S3 online documentation see [Object Tagging](#).

### 4.11.4. Object Metadata Structure in Cassandra

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Partitioning of a Bucket's Object Metadata Into Multiple Records"** (page 168)
- **"How a Bucket's Data Organization Impacts the Bucket's Metadata Partitioning"** (page 169)
  - **"Avoid: Having No Folder Name Variety Within the First Three Levels of Folder Depth "** (page 169)
  - **"Avoid: Having A Very Large Number of Folders with Very Few Objects in Each Folder"** (page 170)

Metadata for a given object is stored in two different types of record in Cassandra:

- A record exclusively for that object (called "skinny row" object metadata because along with the key the row has just one column, for that one object)

- A bucket-level record that includes metadata for that object as well as for other objects in the bucket (called "wide row" object metadata because along with the key the row has 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 GET Object request the system will retrieve the object-level record. While processing a GET Bucket (List Objects) request the system will retrieve bucket-level records. Bucket-level records are also used during batch delete operations and during some *hsstool* operations.

While the structure of object-level records is simple, the structure of bucket-level records is somewhat more complex and will be impacted by how your users and applications organize data within buckets. Since in some circumstances this can have system performance implications, it's useful to understand how bucket-level object metadata structure is impacted by the organization of data within buckets.

#### 4.11.4.1. Partitioning of a Bucket's Object Metadata Into Multiple Records

In older versions of HyperStore, all object metadata for a given bucket was stored in a single record (one record -- also known as a **row** -- per bucket). While having the appeal of simplicity, this design had certain limitations:

- Imposed a hard limit of two billion objects per bucket (since each object's metadata constitutes a column in the metadata row and a Cassandra row can have a maximum of two billion columns)
- Made for possible performance hot spots on drives that happened to store metadata records for buckets that contained a huge number of objects
- Made for possible problems with excess Cassandra "tombstones", when mass deletes were performed on buckets that contained a huge number of objects

Consequently, for buckets created in HyperStore 7.1 and newer the object metadata for each bucket is partitioned into multiple rows in Cassandra. For each bucket, the bucket's object metadata partitioning follows two rules:

- **For objects in the root level of the bucket**, with no folder structure, object metadata partitioning for the bucket is based on the first character of the object name, with case-sensitivity. Examples:

- Metadata for each of these objects will be stored in a different row:

```
ant.txt
beaver.txt
cat.txt
Caribou.txt
314_animals.xls
```

- Metadata for these objects will be stored in the same row:

```
dog.txt
deer.txt
dolphin.txt
```

- **For objects in folders**, metadata partitioning gives each unique folder path its own row, down to three levels of folder depth. Examples:

- Metadata for each of these objects will be stored in a different row:

```
images/logo.png
images/screenshots/dashboard.png
```

```
2020/06/09/report.xls
2020/06/08/report.xls
2020/06/07/report.xls
```

- Metadata for these objects will be stored in the same row:

```
2020/06/09/report.xls
2020/06/09/objectives.docx
2020/06/09/images/venue.jpg
2020/06/09/images/parts/backplane.png
2020/06/09/images/tradeshows/guestspeakers/duranian.png
```

Each of the object metadata rows for a given bucket has a unique row key that is derived from the bucket name and the partitioning criterion (such as a specific folder path). For each bucket, a special metadata record identifies all the partitioned object metadata rows for that bucket, so that the system can aggregate the metadata when executing operations such as an S3 GET Bucket (List Objects) call.

#### 4.11.4.2. How a Bucket's Data Organization Impacts the Bucket's Metadata Partitioning

In the great majority of use cases HyperStore's bucket-level object metadata partitioning scheme works well to serve the goal of optimizing system performance. There are however two atypical ways of organizing bucket data that can result in sub-optimal performance if the number of objects in the bucket is very large.

##### 4.11.4.2.1. Avoid: Having No Folder Name Variety Within the First Three Levels of Folder Depth

HyperStore's object metadata partitioning for a bucket creates a different row for each folder in the bucket, down to three levels of folder depth. This results in partitioning of the bucket's object metadata only if you have at least some variety within the first three levels of folder depth in the bucket. If you put all of your objects directly under a single one-segment or two-segment folder, or if you put all your objects under a single three-segment folder and its sub-folders, then only one object metadata row will be created for that bucket. To illustrate, all of these examples would result in the creation of just one metadata row for the bucket:

- All objects in the bucket are directly under the same one-segment folder:

```
/images/ant.png
/images/bat.png
/images/caterpillar.jpg
/images/...
```

- All objects in the bucket are directly under the same two-segment folder:

```
/accounting/documents/report1.docx
/accounting/documents/demol.pptx
/accounting/documents/spreadsheet1.xlsx
/accounting/documents/...
```

- All objects in the bucket are under the same three-segment folder and its sub-folders:

```
/backup/cloudian-S3/datacenter1/hostlist.txt
/backup/cloudian-S3/datacenter1/host1/2020-06-11.tar.gz
/backup/cloudian-S3/datacenter1/host2/2020-06-11.tar.gz
/backup/cloudian-S3/datacenter1/host3/2020-06-11.tar.gz
/backup/cloudian-S3/datacenter1/host4/2020-06-11.tar.gz
/backup/cloudian-S3/datacenter1/host5/2020-06-11.tar.gz
```

```
/backup/cloudian-S3/datacenter1/...
```

**Note** Variety in the fourth folder level or deeper does not result in metadata partitioning for the bucket.

If just one metadata row is created for the bucket, and if the number of objects in the bucket is very large, then there are increased risks of Cassandra tombstone issues or performance hotspots, and there is also the hard limit of two billion objects in the bucket.

Consequently, **in a bucket in which there is expected to be a very large number of objects, it's best to have some folder name variety within the first three levels of folder paths** so that object metadata for the bucket is partitioned.

#### 4.11.4.2.2. Avoid: Having A Very Large Number of Folders with Very Few Objects in Each Folder

At the opposite extreme of having a very large number of objects under a single folder path is having a very large number of one-, two-, or three-segment folder paths -- such that each folder is given its own object metadata row in Cassandra -- with only one or a few objects under each folder. This structure may encounter sub-standard performance for GET Bucket (Object List) operations, since there would be an inefficiently high ratio of metadata rows to number of objects (and processing a GET Bucket request entails retrieving all of the bucket's metadata rows).

While it's best to avoid this way of organizing data within a bucket, if your use case **requires** that data be organized this way, you can reconfigure the system such that for a specified bucket or buckets, the system creates a metadata row per folder only to one level of folder depth -- rather than the default behavior of creating a metadata row per folder down to three levels of folder depth. So for example, by default each of these eight folder paths in a bucket would have their own metadata row:

```
/cars/toyota/new/  
/cars/toyota/used/  
/cars/honda/new/  
/cars/honda/used/  
/trucks/ford/new/  
/trucks/ford/used/  
/trucks/chevy/new/  
/trucks/chevy/used/
```

By contrast, with the bucket configured to create a metadata row per folder only to one level of folder depth, that same set of folder paths would result in only two metadata rows (one for "/cars/" and all its sub-folders and one for "/trucks/" and all its sub-folders).

If you want to configure certain buckets to use the single folder depth based approach for partitioning metadata, note that:

- The configuration must be applied **before** you create the bucket(s). You cannot apply this configuration to a bucket that already exists.
- Before making the configuration change please consult with Cloudian Support about your use case and your intention to configure the bucket(s) this way.

To configure one or more buckets to use the single folder depth based approach for partitioning metadata, do the following:

1. On the Puppet master node, **add** the following properties to the configuration file [mts.properties.erb](#) (these properties are not in the file by default):

```
cloudian.s3.buckets.folder.depth.one=bucket1,bucket2
cloudian.s3.folder.depth=1
```

where *bucket1,bucket2* is a comma-separated list of buckets to which you want to apply this special configuration (or it can be just a single bucket rather than a comma-separated list). These must be buckets that **do not yet exist** but which you will create later in this procedure.

After adding these properties, save and close the file.

2. Still on the Puppet master node, use the installer to push your configuration change out to the cluster and to restart the S3 Service. If you need instructions for this see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).
3. Use the CMC or another S3 client application to create the buckets that you specified in Step 1 (*bucket1* and *bucket2* in the example).

For the life of these buckets, their object metadata partitioning will be based on first-level folders within the bucket (for instance */cars/* versus */trucks/*) and not on any deeper level folder organization.

4. After creating all buckets to which you want to apply this special configuration, return to the *mts.properties.erb* file and **remove** this line that you previously had added:

```
cloudian.s3.folder.depth=1
```

**Note** Do not remove the *cloudian.s3.buckets.folder.depth.one=bucket1,bucket2* line.

5. Use the installer to push your configuration change out to the cluster and to restart the S3 Service.

## 4.11.5. Elasticsearch Integration for Object Metadata

*Subjects covered in this section:*

- **"Overview of Elasticsearch Integration for Object Metadata"** (page 171)
- **"Enabling Elasticsearch Integration for Object Metadata"** (page 173)
- **"Using the elasticsearchSync Tool "** (page 174)
- **"Option to Send Metadata to an HTTP Server Rather than to Elasticsearch"** (page 175)

### 4.11.5.1. Overview of Elasticsearch Integration for Object Metadata

HyperStore supports integrating with an [Elasticsearch](#) cluster, so that you can use the Elasticsearch cluster to search through the object metadata associated with the objects stored in your HyperStore system. This could be an Elasticsearch cluster that you already have running in your environment, or an Elasticsearch cluster that you install specifically for the purpose of integrating with HyperStore.

**Note** To work with HyperStore your **Elasticsearch version must be a 6.x version, 6.6 or newer**. For availability and performance Cloudian recommends that you have at least three nodes in your Elasticsearch cluster.

When you configure the HyperStore system to integrate with an Elasticsearch cluster, you can then enable object metadata search on a per storage policy basis (as described in **"Elasticsearch Integration for Object Metadata"** (page 171)). For a storage policy on which you've enabled object metadata search, the following will happen:

- For each object that subsequently gets uploaded into a HyperStore bucket that uses that storage policy, HyperStore will retain the object metadata locally (as it always does) and also transmit a copy of the object metadata into your Elasticsearch cluster (using HyperStore's built-in Elasticsearch REST client).
  - This includes HyperStore system-defined object metadata and user-defined metadata (for more information on these types of metadata see **"Object Metadata Feature Overview"** (page 164) ). It does **not** include object tags.
  - HyperStore will create in your Elasticsearch cluster an index for each bucket that uses a metadata search enabled storage policy. The indexes created in Elasticsearch are named as follows (using lower case only):

*cloudian-<cloudianclustername>-<regionname>-<datacentername>-<bucketname>-<bucketcreationdatetime>*

For example:

*cloudian-cloudianregion1-region1-dc1-bucket2-2017-10-19t18:25:56.955z*

Each index will be created in Elasticsearch with three shards and two replicas.

- For each object uploaded to HyperStore a "document" containing the object metadata will be created in Elasticsearch. Each such document will have a name (ID) in this format:

*<bucketname>/<objectname>*

In the case of versioned objects there will be a separate Elasticsearch document for each object version, named as:

*<bucketname>/<objectname>\u0001u-<versionId>*

The document names will be URL-encoded before transmission to the ES cluster and then URL-decoded upon retrieval from the ES cluster.

**Note** Enabling object metadata search on a storage policy results in HyperStore copying into Elasticsearch any object metadata associated with objects uploaded **from that time forward**. If you also want to load into Elasticsearch the object metadata associated with objects that are already in your HyperStore system, you can use the Elasticsearch synchronization tool that comes with HyperStore -- as described in **"Elasticsearch Integration for Object Metadata"** (page 171).

- When an object is updated (overwritten by a new S3 upload operation) in HyperStore, its metadata will be updated in Elasticsearch; and when a object or object version is deleted in HyperStore (by an S3 delete operation or by execution of a bucket lifecycle policy), its metadata will be deleted in Elasticsearch.
- If objects that have metadata in Elasticsearch get auto-tiered to a remote destination, their metadata will remain in Elasticsearch.
- If a bucket is deleted from HyperStore, its corresponding index will be deleted in Elasticsearch.

- All insertions, updates, and deletes of HyperStore object metadata in your Elasticsearch cluster are implemented by an [hourly cron job](#) in HyperStore. Note that this means that object metadata associated with a new S3 upload will not immediately appear in Elasticsearch.
- If you enable metadata search for a storage policy, and then at a later time disable metadata search on that storage policy, any object metadata that had been copied to Elasticsearch during the period when metadata search was enabled will remain there (it will not be deleted by HyperStore).

Note that when you enable Elasticsearch integration, HyperStore only **writes** to your Elasticsearch cluster. It does not read from the cluster, and you cannot use HyperStore to search through or retrieve object metadata in Elasticsearch. For that you must use Elastic Stack applications, such as Kibana.

**Note** If you also want to use the Elastic Stack for HyperStore S3 request traffic analysis (as described in ["Setting Up Elastic Stack for S3 Request Traffic Analysis"](#) (page 631)), contact Clodian Sales Engineering or Support to discuss using the same Elastic Stack cluster for both object metadata search and S3 request log analysis.

#### 4.11.5.2. Enabling Elasticsearch Integration for Object Metadata

To enable Elasticsearch integration in your HyperStore system:

1. On your Puppet master node, make these configuration file edits:
  - In [common.csv](#), edit the `cloudian_elasticsearch_hosts` setting to equal a comma-separated list of the IP addresses of your Elasticsearch hosts. In a production environment Clodian recommends that you use at least three Elasticsearch hosts.
  - In [mts-ui.properties.erb](#), set the `elasticsearch.enabled` property to `true`. By default it is false.
  - In [mts.properties.erb](#):
    - If your Elasticsearch cluster **does not use the X-Pack extension**, set `cloudian_elasticsearch.xpack.enabled` to `false`. By default this is set to true.
    - If your Elasticsearch cluster **does use the X-Pack extension**:
      - Set `cloudian_elasticsearch.xpack.username` and `cloudian_elasticsearch.xpack.password` to the user name and password that you have already established in your Elasticsearch cluster set-up.
      - Optionally, change the `cloudian_elasticsearch.ssl.*` settings to suit your environment.

SSL can be enabled only when Xpack is enabled. To use SSL, in your ESDB you will first need to create a new keystore: `./keytool -import -alias elasticCA -file /etc/elasticsearch/certs/client/client-ca.cer -keystore truststore.jks`.

Then copy the `truststore.jks` file to each HyperStore node, in the directory path specified by `mts.properties.erb`: `cloudian_elasticsearch.ssl.truststore.path` (the default setting for the path is `/opt/cloudian/conf/certs/truststore.jks`).
2. On the Puppet master node, use the installer to push your configuration changes out to the cluster, restart the S3 Service, and restart the CMC. For more detail see ["Pushing Configuration File Edits to the Cluster and Restarting Services"](#) (page 506).
3. Log into the CMC and go to the [Storage Policies](#) page. Now when you either create a new storage policy or edit an existing storage policy, the interface will display an "Enable metadata search" option. For each storage policy for which you select the "Enable metadata search" option (and **Save** your

change), object metadata from buckets that use that storage policy will be sent to the Elasticsearch cluster.

**Note** Recall that when you enable Elasticsearch for a storage policy, that means that **from that time forward** objects that get uploaded into buckets that use the storage policy will have copies of their object metadata sent to Elasticsearch. Enabling this option does **not** copy into Elasticsearch metadata from objects that are already stored in buckets that use the storage policy.

#### 4.11.5.3. Using the elasticsearchSync Tool

HyperStore includes a tool that you can use to synchronize object metadata in your Elasticsearch cluster to object metadata currently in your HyperStore cluster. To use the tool you must first enable Elasticsearch integration for your HyperStore system and for one or more of your storage policies, as described in **"Elasticsearch Integration for Object Metadata"** (page 171). Then, using different script options you can bring object metadata in Elasticsearch up to date for any of the following:

- All buckets for which Elasticsearch integration is enabled (that is, all buckets that use Elasticsearch-enabled storage policies).
- A single bucket for which Elasticsearch integration is enabled (a bucket that uses an Elasticsearch-enabled storage policy).
- A single object in a bucket for which Elasticsearch integration is enabled.

The primary use case for the synchronization tool is to populate Elasticsearch with object metadata that was already in your HyperStore system before you enabled Elasticsearch integration. For example, suppose you enable Elasticsearch integration for one or more of your existing storage policies that are already being used by existing buckets. HyperStore will automatically copy into Elasticsearch the metadata associated with objects that get uploaded into those buckets from that point forward. But if you want to copy into Elasticsearch **all** the metadata associated with objects that are currently in those buckets -- including objects that were already in the buckets before Elasticsearch integration was enabled -- you need to use the sync tool.

Another use case for the sync tool is if your Elasticsearch cluster has been down or unreachable for more than a few hours. HyperStore uses an [hourly cron job](#) to apply needed updates to object metadata in Elasticsearch. Elasticsearch update requests that fail during one run of the cron job are retried during the next run of the cron job. But if your Elasticsearch cluster is unreachable for more than a few hours, the Cassandra-based queue for unprocessed Elasticsearch update requests can get filled up. Consequently if Elasticsearch has been unreachable for more than a few hours you should use the sync tool when Elasticsearch comes back online (for all Elasticsearch-enabled buckets). This will bring Elasticsearch up to date with HyperStore object uploads and deletions that occurred while Elasticsearch was unavailable.

The sync tool is located on each of your HyperStore nodes, under the `/opt/cloudian/bin` directory. Once you change into that directory, the tool syntax is as follows:

Sync all Elasticsearch-enabled buckets:

```
# ./elasticsearchSync all
```

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

If you are using the [HyperStore Shell \(HSH\)](#), you do not need to change into the `/opt/cloudian/bin` directory or use the leading `./`. Instead, from any directory just run the command as:

```
$ elasticsearchSync all
```

The same points apply to the other command options described below.

Sync one Elasticsearch-enabled bucket:

```
# ./elasticsearchSync bucket <bucketname>
```

Sync one object in an Elasticsearch-enabled bucket:

```
# ./elasticsearchSync object <bucketname>/<objectname>
```

Sync one version of one object in an Elasticsearch-enabled bucket:

```
# ./elasticsearchSync object <bucketname>/<objectname> <versionid>
```

**Note** Along with performing your specified action, the running of the sync tool always triggers the processing of any Elasticsearch update requests that are currently in the retry queue.

#### 4.11.5.4. Option to Send Metadata to an HTTP Server Rather than to Elasticsearch

HyperStore supports an option to send object metadata copies to an HTTP server of your choosing. If you enable this option, then HyperStore will send object metadata to your specified HTTP URI **instead of** sending it to Elasticsearch. This works only for object metadata associated with objects that get uploaded after you enable this feature. It does not work for metadata associated with any objects already in the system.

**Note** The [elasticsearchSync tool](#) -- for transmitting metadata associated with any objects already in the system -- only works with Elasticsearch as the destination. It does not work with an HTTP Server as the destination.

To configure HyperStore to send object metadata to your specified HTTP server:

1. On your Puppet master node, make these configuration file edits:
  - In [common.csv](#), set `cloudian_elasticsearch_hosts` to the IP address of the HTTP server. By default this setting is empty.
  - In [mts-ui.properties.erb](#), set the `elasticsearch.enabled` property to `true`. By default it is false.
  - In [mts.properties.erb](#):
    - Set `cloudian.elasticsearch.process.type` to `http`. By default this is set to `elasticsearch`.
    - Set `cloudian.elasticsearch.http.url` to the HTTP URL that you want metadata to be sent to. For example, `http://<HTTP_server_IP_address>:<port #>`. By default this setting is empty.
2. On the Puppet master node, use the installer to push your configuration changes out to the cluster, restart the S3 Service, and restart the CMC. For more detail see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).
3. Log into the CMC and go to the [Storage Policies](#) page. Now when you either create a new storage policy or edit an existing storage policy, the interface will display an "Enable metadata search" option. For each storage policy for which you select the "Enable metadata search" option (and **Save** your change), object metadata from buckets that use that storage policy will be sent to the HTTP server.

**Note** Recall that when you enable metadata search for a storage policy, that means that **from that time forward** objects that get uploaded into buckets that use the storage policy will have copies of their object metadata sent to the HTTP server. Enabling this option does **not** copy to

the HTTP server metadata from objects that are already stored in buckets that use the storage policy.

Here is an example of a POST request by which HyperStore submits object metadata to an HTTP server. In this example the request body contains the metadata associated with an S3 PUT Object operation that was processed in HyperStore.

```
POST / HTTP/1.1.
Content-type: application/json.
Content-Length: 209.
Host: 10.20.2.64:9000.
Connection: Keep-Alive.
User-Agent: Apache-HttpClient/4.5.2 (Java/1.8.0_172).
Accept-Encoding: gzip,deflate.
.
{"id":"buser1%2Fabc", "bucketName":"buser1", "objectName":"abc",
"lmt":"2019-06-26T08:23:16.224Z", "objectSize":1,
"contentType":"application/octet-stream", "esTime":"Wed Jun 26 16:23:16 CST 2019",
"userMetadata":{}}
```

## 4.12. Auto-Tiering

### 4.12.1. Auto-Tiering Feature Overview

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Tiering Destination Accounts, Credentials, and Buckets"** (page 177)
- **"Bridge Mode"** (page 178)
- **"Option to Retain a Local Copy"** (page 178)
- **"Auto-Tiering Logging"** (page 178)
- **"Auto-Tiering Licensing"** (page 179)
- **"Auto-Tiering of Encrypted Objects"** (page 179)
- **"Auto-Tiering and User-Defined Object Metadata"** (page 179)
- **"How Auto-Tiering Impacts Usage Tracking, QoS, and Billing"** (page 179)

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: Amazon S3, Amazon Glacier, 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 for which the full object name starts with a particular prefix
- 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, 2018
  - Move objects immediately after they're created ("Bridge Mode"). Note that with Bridge Mode, filtering by prefix is not supported -- if Bridge Mode is used, this applies to all objects regardless of prefix.

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 **"Setting Up Auto-Tiering"** (page 180).

**Note Auto-tiering restrictions based on destination type:**

- \* By default the largest object size that can be auto-tiered to Amazon, Google, or other S3-compliant destinations is 100GB. If you want to tier objects larger than this, consult with Cloudian Support. This 100GB limit does not apply to tiering to Azure or Spectra BlackPearl.
- \* Tiering to Azure or Spectra BlackPearl is not supported for source buckets that have versioning enabled or that have had versioning enabled in the past.
- \* When auto-tiering to Spectra BlackPearl is used for a bucket, objects in the bucket will not be auto-tiered unless they are larger than 5MB. Objects 5MB or smaller will remain in HyperStore. To change this limit, consult with Cloudian Support.

#### 4.12.1.1. Tiering Destination Accounts, Credentials, and Buckets

Auto-tiering to Amazon 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 **"Setting Up Auto-Tiering"** (page 180).

HyperStore encrypts supplied tiering account credentials and stores them in the Redis Credentials database, 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 Amazon 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 184).

**Note** In the case of auto-tiering to Amazon Glacier, the HyperStore system creates a bucket in Amazon S3 and configures that remote bucket for immediate transitioning to Glacier. Objects are then auto-tiered from HyperStore to Amazon S3, where they are immediately subject to Amazon's automated mechanism for transitioning objects to Glacier.

#### 4.12.1.2. Bridge Mode

The HyperStore auto-tiering feature supports a "Bridge Mode" 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 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 179) for detail).

If the initial attempt to move an object to the destination system fails with a temporary error, the system will retry once every six hours until either the object is successfully moved or a permanent error is encountered (an example permanent error would be if, outside of HyperStore, someone had deleted the tiering destination bucket). The retries are triggered by a [system cron job](#). 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 local copy of the object will not be deleted until the object is successfully moved to the destination system, as indicated by the destination system returning a success status.

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 Amazon Glacier or Spectra BlackPearl.

#### 4.12.1.3. 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.

#### 4.12.1.4. 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](https://cloudian-tiering-request-info.log). For regular auto-tiering that occurs on a defined schedule this request logging occurs on all nodes in the region in which the cron job primary node is located (since that daily cron job distributes the auto-tiering execution workload across all nodes in the region).

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.

#### 4.12.1.5. Auto-Tiering Licensing

Your HyperStore license may impose a limit on how much tiered data you can have stored in external systems other than HyperStore. For details see **"Licensed Maximum Tiered Storage Usage"** (page 18).

#### 4.12.1.6. Auto-Tiering of Encrypted Objects

For information about how auto-tiering works together with the server-side encryption feature, see **"Encryption and Auto-Tiering"** (page 106). 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.

#### 4.12.1.7. Auto-Tiering and User-Defined Object Metadata

For information about how auto-tiering impacts user-defined object metadata, see **"Object Metadata, Cross-Region Replication, and Auto-Tiering"** (page 165).

#### 4.12.1.8. 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 Amazon 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 184).

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 Amazon, 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 135) and **"Usage Reporting and Billing Feature Overview"** (page 138).

## 4.12.2. Setting Up Auto-Tiering

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Enable and Configure the Auto-Tiering Feature in the CMC"** (page 180)
- **"Configure Tiering Destinations"** (page 181)
- **"Configure Auto-Tiering Rules for Individual Buckets (CMC)"** (page 183)
- **"Configure Auto-Tiering Rules for Individual Buckets (S3 API and Admin API)"** (page 183)
- **"Tiering Statistics for Spectra BlackPearl"** (page 184)

To use the CMC to set up buckets for auto-tiering, follow the high-level instructions below. First you will enable and configure the auto-tiering feature in the CMC, and then you or your users can use the CMC to configure individual buckets for auto-tiering. (If you want to use an S3 client application other than the CMC, jump down to **"Configure Auto-Tiering Rules for Individual Buckets (S3 API and Admin API)"** (page 183)).

### 4.12.2.1. Enable and Configure the Auto-Tiering Feature in the CMC

**Note** None of the settings described below is applicable to using a third party S3 client application to invoke the HyperStore S3 Service's auto-tiering feature. These settings are applicable only to using the auto-tiering feature through the CMC.

By default auto-tiering functionality is **disabled** in the CMC, such that CMC users when configuring bucket life-cycle properties will not see an option for auto-tiering. Do the following to enable and configure the auto-tiering feature so that CMC users can apply auto-tiering to their buckets:

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, Google, and Azure by default) and no other destinations, 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 tiering destinations that will display for users, and the endpoints for those destinations, as described in **"Configure Tiering Destinations"** (page 181) 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 a Glacier, Azure, or Spectra BlackPearl endpoint.
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.

#### 4.12.2.2. 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, Google Cloud Storage, and Azure; and by default the endpoints for those destinations are as follows:

Destination	Default Endpoint
AWS S3	<a href="https://s3.amazonaws.com">https://s3.amazonaws.com</a>
AWS Glacier	<a href="https://s3.amazonaws.com">https://s3.amazonaws.com</a>
Google Cloud Storage	<a href="https://storage.googleapis.com">https://storage.googleapis.com</a>
Azure	<a href="https://blob.core.windows.net">https://blob.core.windows.net</a>

**Note** If your original HyperStore version was older than 7.1.4, then after upgrade to 7.1.4 or later the default list of tiering destinations will also include Spectra BlackPearl with endpoint <https://b-plab.spectrallogic.com>.

If you want to change this list -- by changing the endpoint for any of the destinations above, or by adding or removing destinations -- you can do so by editing the **"cmc\_bucket\_tiering\_default\_destination\_list"** (page 541) setting in [common.csv](#). The setting is formatted as a quote-enclosed list, with comma-separation between destination attributes and vertical bar separation between destinations, like this:

```
"<name>,<endpoint>,<protocol>|<name>,<endpoint>,<protocol>|..."
```

This can be multiple destinations (as it is by default), or you can edit the setting to have just one destination in the "list" if you want your users to only use that one destination.

The *<name>* will display in the CMC interface that bucket owners use to configure auto-tiering, as the auto-tiering destination name. The *<protocol>* must be one of the following:

- s3
- glacier
- azure
- spectra

If you wish you can include multiple destinations of the same type, if those destinations have different end-points. For example, "Spectra 1,<endpoint1>,spectra|Spectra 2,<endpoint2>,spectra". Each such destination will then appear in the CMC interface for users configuring their buckets for auto-tiering.

If you make any changes to this setting, push your changes to the cluster and restart the CMC. For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

#### **Note About Tiering to a Different HyperStore Region or System**

In the CMC configuration set-up described in **"Enable and Configure the Auto-Tiering Feature in the CMC"** (page 180) (above), there are two ways in which your users can use a HyperStore service region as the tiering destination:

- You configure the system so that all users tier to a system default tiering destination, and you define that default tiering URL to be a HyperStore service region endpoint.
- You configure the system so that users can enter a user-defined endpoint, and then individual users can specify a HyperStore service region as their user-defined tiering endpoint.

The HyperStore region that's being used as the tiering destination could be a different region in the same HyperStore system (for example users are tiering from Region1 to Region2 within the same HyperStore system); or it could be a service region within a different HyperStore system altogether (users are tiering from HyperStore system A to a region in HyperStore system B). For background on the distinction between a HyperStore region and a HyperStore system, see **"System Levels"** (page 30).

If your users will be tiering to a region in a **different HyperStore system**, then for tiering to that region to work "out of the box" the endpoint for that region must be specified in this format:

`s3-<region>.<domain>`

For example:

`s3-boston.company.com`

where "boston" is the actual region name in the destination HyperStore system's own system configuration.

If the endpoint for the external HyperStore system is in any format other than the above -- if, for example, the endpoint is simply `boston.company.com` rather than `s3-boston.company.com` -- then for tiering to work you must first make the following configuration change in your own HyperStore system (the source system from which the tiering will originate):

1. On the Puppet Master node, open the following file in a text editor:

```
/etc/cloudian-<version>-puppet/modules/cloudians3/templates/  
tiering-regions.xml.erb
```

2. Copy the sample "Region" block at the top of the `tiering-regions.xml.erb` file and paste it toward the end of the file after the existing "Region" blocks (but before the closing "</XML>" tag that's at the very end of the file).
3. Edit the block as follows:
  - Use the "Name" element to specify the region name of destination system region that you will tier to. Enter the region name exactly as it is defined in the destination HyperStore system.
  - Leave the "ServiceName", "Http", and "Https" elements at their default values.
  - Use the "Hostname" element to specify the service endpoint that you will tier to.

For example, if the region name is "boston" and the service endpoint is "boston.company.com", then your edited Region block would look like this:

```
<Region>
  <Name>boston</Name>
  <Endpoint>
    <ServiceName>s3</ServiceName>
    <Http>true</Http>
    <Https>true</Https>
    <Hostname>boston.company.com</Hostname>
  </Endpoint>
</Region>
```

**Note** Make sure that the service endpoint that you will tier to is resolvable in your DNS system.

**Note** Do not have two instances of "s3" in the endpoint, like *s3-tokyo.s3.enterprise.com*. This may cause auto-tiering errors (and cross-region replication errors, if you use the cross-region replication feature).

4. Save your change and close the *tiering-regions.xml.erb* file.
5. Push your changes to the cluster and restart the S3 Service. For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

#### 4.12.2.3. Configure Auto-Tiering Rules for Individual Buckets (CMC)

HyperStore service users can configure auto-tiering for their own buckets through the CMC's [Bucket Properties](#) page. Here users can select the tiering destination (within the limits of the system configuration as described in the preceding section), the tiering schedule (including, if desired, having different schedules for different sets of objects based on object name prefix), 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 that user's data.

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 for Object Auto-Tiering or Expiration"** (page 227)

**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.

#### 4.12.2.4. Configure Auto-Tiering Rules for Individual Buckets (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 *PUT Bucket lifecycle*. The extensions take the form of request headers to specify the bucket's auto-tiering destination and mode (*x-gmt-tieringinfo*), whether to base auto-

tiering timing on object creation time or last access time (*x-gmt-compare*), and whether to retain a local copy of the object after auto-tiering occurs (*x-gmt-post-tier-copy*). For details about these API extensions see [PUT Bucket lifecycle](#).

If you plan to configure auto-tiering on a bucket by calling the *PUT Bucket lifecycle* 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 *PUT Bucket lifecycle* 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 such as Spectra BlackPearl. For details about the relevant Admin API methods, see **"tiering"** (page 861).

Also, if you want to support auto-tiering to an external HyperStore system -- not a different region within the same HyperStore system but rather a different HyperStore system altogether -- see **"Note About Tiering to a Different HyperStore Region or System"** (page 182), in regard to the format requirements for the tiering endpoint.

#### 4.12.2.5. 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.

```
# 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.

### 4.12.3. 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
- Direct access through the destination system

## 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, or Azure (not Glacier or Spectra Black-Pearl)
- The bucket's auto-tiering is configured to support Streaming of auto-tiered objects.

If you're uncertain regarding 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.

**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](#).

**Note** As an alternative to accessing auto-tiered objects through the CMC, you can use a third party S3 client application to submit [POST Object restore](#) requests (for any auto-tiered objects) or [GET Object](#) requests (for auto-tiered objects that support streaming) to the HyperStore system's S3 Service. You can delete auto-tiered objects by submitting [DELETE Object](#) requests to the HyperStore system's S3 Service.

## Direct Access Through the Destination System

Objects auto-tiered to a destination system can be accessed 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 177) for detail). After objects have been auto-tiered from Hyper-

Store 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.

**IMPORTANT !** Users should **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 to another HyperStore region, any overwriting or deleting of objects should be done through the source bucket not the destination bucket.

## 4.13. Cross-Region Replication

### 4.13.1. Cross-Region Replication Feature Overview

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Bi-Directional Replication"** (page 187)
- **"Replication Timing and Error Handling"** (page 187)
- **"Cross-Region Replication Versus Auto-Tiering"** (page 188)
- **"Cross-Region Replication Impact on Usage Tracking"** (page 188)
- **"Cross-Region Replication of Encrypted Objects"** (page 189)
- **"Cross-Region Replication and WORM"** (page 189)

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 as well as the object data itself.

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.
- 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 **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.

- In the case of an object deletion request that **does specify the object version**, the object version is deleted from the source bucket but is **not** deleted from the destination bucket.

#### 4.13.1.1. 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.

#### 4.13.1.2. Replication Timing and 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 622) 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.

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 region -- 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).

#### 4.13.1.3. Cross-Region Replication Versus Auto-Tiering

The key difference between cross-region replication and auto-tiering from one region to another is that with cross-region replication each replicated object is stored in both the source region **and** the destination region. By contrast, after an object is auto-tiered from a source region to a destination region the object data is (by default) stored **only** in the destination region.

Also, auto-tiering is supported for destinations that are outside the HyperStore system -- for example you can auto-tier objects to Amazon S3 or other S3-compatible storage clouds; and you can auto-tier objects from one HyperStore system to a completely different HyperStore system.

For more information on auto-tiering see "[Auto-Tiering Feature Overview](#)" (page 176)

#### 4.13.1.4. 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 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.

#### 4.13.1.4.1. 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.

#### 4.13.1.5. Cross-Region Replication of Encrypted Objects

Whether or not objects encrypted by server-side encryption are replicated from a source bucket to the destination bucket depends on the server-side encryption method used. For details see **"Encryption and Cross-Region Replication"** (page 106).

#### 4.13.1.6. Cross-Region Replication and WORM

For information on this topic see **"WORM (Object Lock)"** (page 121).

### 4.13.2. Configuring 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.

#### Configure Cross-Region Replication for a Bucket (CMC)

**Note** Before configuring cross-region replication, 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 **"Set Versioning for a Bucket"** (page 239).

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 237). 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*).

Also in the CMC, you can disable CRR on a source bucket for which it is currently enabled.

#### Configure Cross-Region Replication for a Bucket (S3 API)

S3 applications can invoke the HyperStore implementation of the S3 API to configure a source bucket for cross-region replication. For detail see [PUT Bucket replication](#).

As with Amazon S3, the HyperStore implementation of the S3 API 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 [PUT Bucket versioning](#) API.

HyperStore also supports the [GET Bucket replication](#) method for retrieving a bucket's current CRR configuration, and the [DELETE Bucket replication](#) method for disabling CRR on a source bucket for which it is currently enabled.

## 4.14. Smart Support

### 4.14.1. Smart Support and Diagnostics Feature Overview

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Smart Support"** (page 190)
- **"On-Demand Node Diagnostics"** (page 191)

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.

#### 4.14.1.1. 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 with 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 and routing table information for each node
- Storage capacity utilization information for each node
- Disk diagnostics data for each node
- Service status (service UP/DOWN) history for each node
- 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 on 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
- Configuration files 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

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). Under `/var/log/cloudian` on that 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 Cloudian Support once each day. By default 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.

#### 4.14.1.1.1. Automatic Support Case Creation for Failed Disks

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 Cloudian 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 Management Feature Overview"** (page 157).

#### 4.14.1.2. On-Demand Node Diagnostics

HyperStore also supports a mechanism that allows you to easily collect deep diagnostic data for a specific node or nodes 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 Cloudian 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 on the target node(s)
- Configuration files from the target node(s)
- Puppet and Salt configuration files and logs from the system

On the target node(s), under the directory `/var/log/cloudian/cloudian_sysinfo`, the Node Diagnostics mechanism packages all this data into a file named `<hostname>_<YYYYMMDDhhmm>.tar.gz`. Optionally you can have the system also automatically send a copy of the package(s) to Cloudian Support.

**Note** The system creates the `/var/log/cloudian/cloudian_sysinfo` directory on a node the first time you use the Collect Diagnostics feature for that node.

### 4.14.2. 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 setting the `phonehome_uri`, `phonehome_bucket`, `phonehome_access_key`, and `phonehome_secret_key` settings in [common.csv](#). For more information see **"phonehome\_uri"** (page 524) and the subsequent setting descriptions.
- Have the daily Smart Support upload use a local forward proxy by setting the `phonehome_proxy_host`, `phonehome_proxy_port`, `phonehome_proxy_username`, and `phonehome_proxy_password` settings in `common.csv`. For more information see **"phonehome\_proxy\_host"** (page 523) and the subsequent setting descriptions.
- Disable the daily Smart Support upload by setting `phonehome.enabled` in [mts.properties.erb](#) to false. **This is not recommended.**
- Have on-demand Node Diagnostics uploads (triggered by your using the CMC's **"Collect Diagnostics"** (page 328) function) go to an S3 destination other than Cloudian Support, by setting the `sysinfo.uri`, `sysinfo.bucket`, `sysinfo.accessKey`, and `sysinfo.secretKey` properties in `mts.properties.erb`. For more information see **"sysinfo.uri"** (page 575) and the subsequent property descriptions.

**Note** Be sure to do a Puppet push and restart the S3 Service if you edit any of these configuration settings. For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

#### 4.14.2.1. Other Configuration Considerations

##### *Changing the timing of the daily diagnostics upload*

When enabled, the daily diagnostics upload to an S3 URI is triggered by a HyperStore cron job. The timing of the cron job is configured in `/etc/cloudian-<version>-puppet/modules/cloudians3/templates/cloudian-crontab.erb` on the Puppet master. If you want to edit the cron job configuration, look for the job that includes the string `"phoneHome"`. If you edit the crontab, do a Puppet push. No service restart is necessary.

##### *Deletion of old daily diagnostics and node diagnostics packages*

The deletion of old diagnostics packages is managed by Puppet, as configured in `common.csv` by the **"cleanup\_directories\_byage\_withmatch\_timelimit"** (page 515) setting (for daily system diagnostics packages associated with the Smart Support feature) and the **"cleanup\_sysinfo\_logs\_timelimit"** (page 516) setting (for on-demand Node Diagnostics packages that you've generated). By default these settings have Puppet delete the diagnostics packages after they are 15 days old. This presumes that you have left the

Puppet daemons running in your HyperStore cluster, which is the default behavior. If you do not leave the Puppet daemons running the diagnostics logs will not be automatically deleted. In that case you should delete the old packages manually, since otherwise they will eventually consume a good deal of storage space.

If you edit either of these settings in *common.csv*, do a [Puppet push](#). No service restart is necessary.

#### *Compliance with the European Union's General Data Protection Regulation (GDPR)*

As part of a GDPR compliance program, you can have HyperStore remove personally identifiable user information -- such as user IDs and client IP addresses -- from the S3 request log copies and S3 application log copies that get uploaded to Cloudian Support as part of the Smart Support and Node Diagnostics features. To do so, set **"phonehome\_gdpr"** (page 525) to *true* in [common.csv](#) (by default it's set to *false*). Setting this parameter to *true* will not remove the user IDs and client IP addresses from the original S3 request logs on your HyperStore nodes -- just from the log file copies that get sent to Cloudian. In the log file copies that get sent to Cloudian the user ID and IP address fields will say "Not Available".

After editing *common.csv* do a Puppet push and restart the S3 Service. For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

### 4.14.3. 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 328) 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) automatically uploaded to Cloudian Support (or an alternative S3 destination, if you have [configured the system to support this option](#)).

Each node's diagnostics package will be created on the node itself, at path */var/log/cloudian/cloudian\_sysinfo/<hostname>\_<YYYYMMDDhhmm>.tar.gz*.

This page left intentionally blank



## Chapter 5. 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** The CMC supports Firefox and Chrome browsers. It is recommended that you use a recent version of one of those browsers when accessing the Console. Internet Explorer is not supported — you may experience display problems with some CMC pages if you use IE.

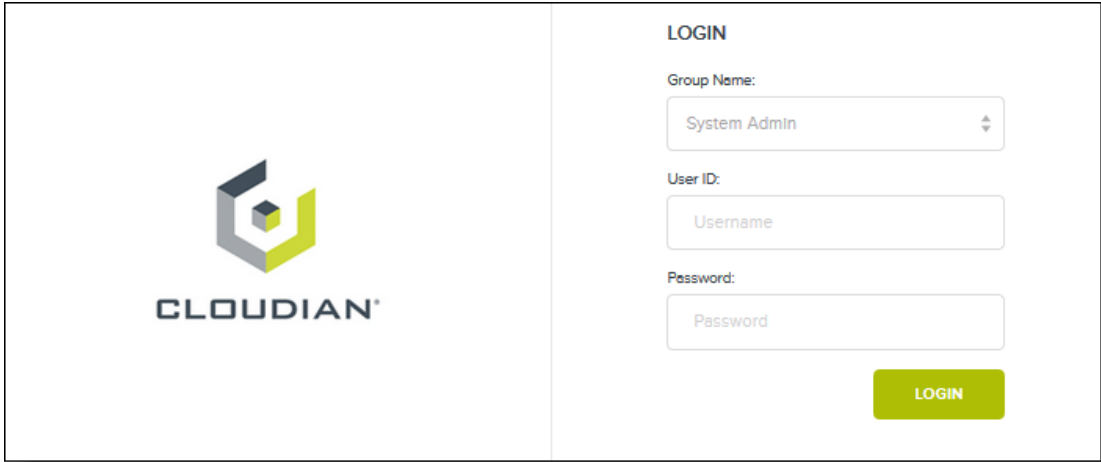
To access the CMC for the first time, follow these steps:

1. Point a browser to `https://<CMC_host>:8443/Cloudian`

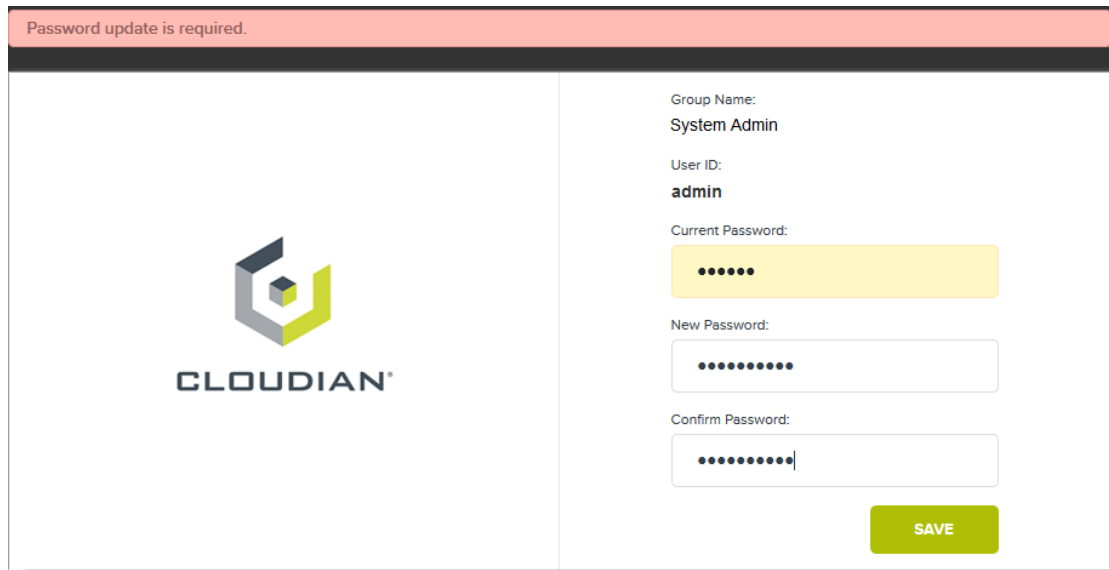
Since the CMC runs on all of your HyperStore nodes, for `<CMC_host>` you can use the fully qualified domain name (FQDN) or IP address of any node.

**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.



3. 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..



The screenshot shows the Cloudian CMC login interface. At the top, a red banner reads "Password update is required." Below this, the Cloudian logo is on the left. On the right, the login form is displayed with the following fields: "Group Name:" with the value "System Admin", "User ID:" with the value "admin", "Current Password:" with a masked input field, "New Password:" with a masked input field, and "Confirm Password:" with a masked input field. A green "SAVE" button is located at the bottom right of the form.

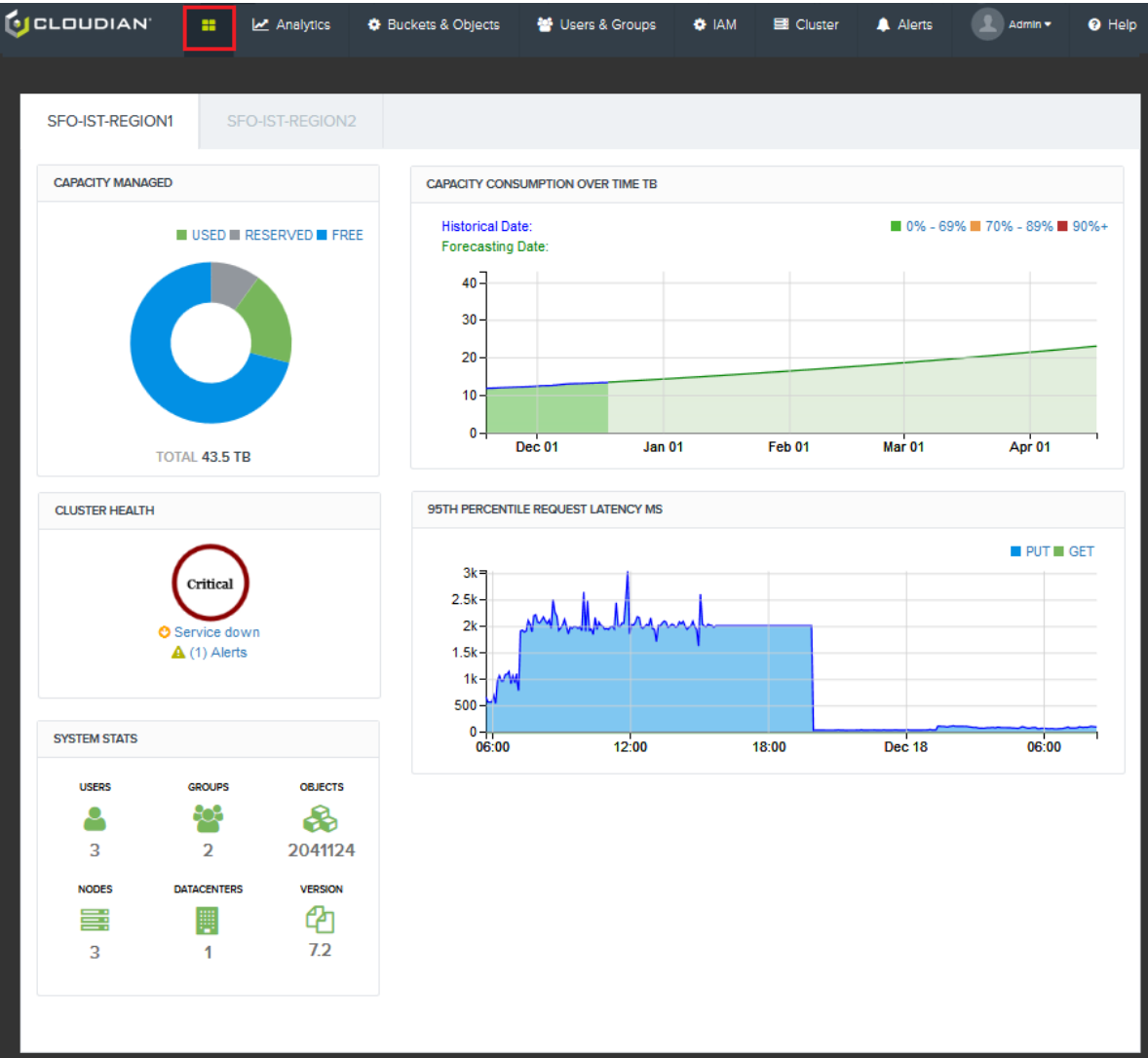
**Note** The first time you try to log into the CMC the system requires you to create a new password for the *admin* user. On subsequent logins to the CMC as the *admin* user, use the password that you created.

**Note** All user logins to the CMC are recorded in the CMC application log [cloudian-ui.log](#).

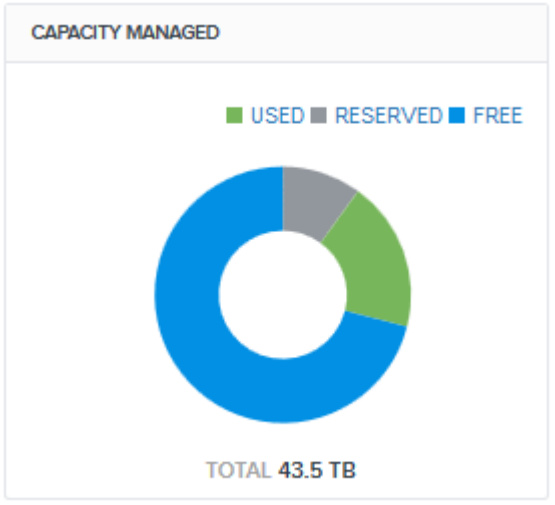
## 5.1. Dashboard

### 5.1.1. 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, Reserved, 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 your total system capacity is consumed by stored 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 1MB object that's replicated three times in the system counts as 3MB toward the Used capacity measurement.

- The **Reserved** segment indicates the portion of total system storage capacity that is reserved and cannot be used for data storage. The Reserved portion consists of the Linux "reserved blocks percentage" plus the HyperStore stop-write buffer:
  - By default in CentOS/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 157).



The Reserved segment always displays in **gray**.

- The **Free** segment indicates the portion of total system storage capacity that is neither used nor reserved, and is therefore available for storing new 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	<div></div>	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

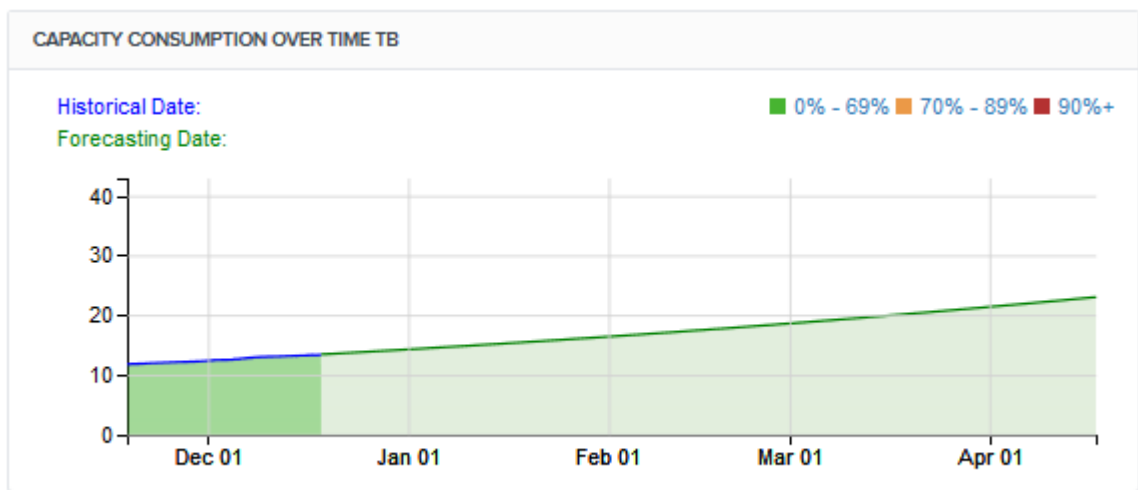
- Add Region
- Uninstall Node
- hssstool rebalance
- hssstool repair or repaired
- hssstool cleanup or cleanuped

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 GBs, TBs, PBs, or EBs (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 198).

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.

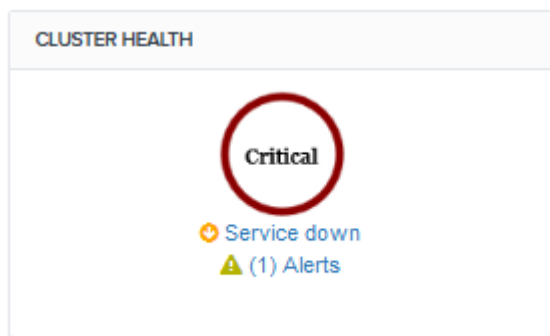
Links to the **Cluster Usage** page, where you can break this capacity consumption data down by data center and node.

**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 71).


**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.

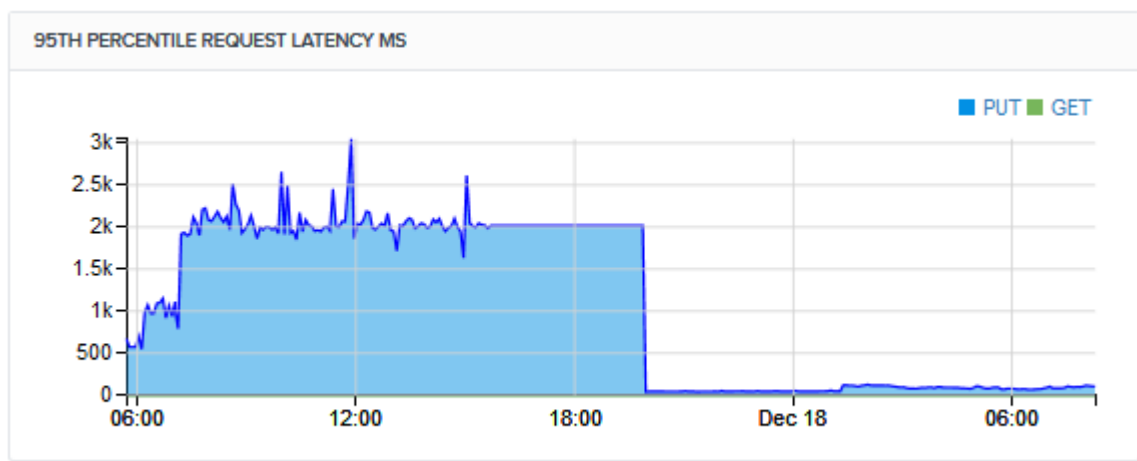
Cluster Status Icon	Condition Message(s)	Action to Take
	"System is X% full" (where X is 90 or more)	Click the condition message to go to the <a href="#">Capacity Explorer</a> page for more detail regarding system capacity usage. At this level of storage capacity utilization the system may soon stop 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 " <b>Capacity Monitoring and Expansion</b> " (page 71).

Cluster Status Icon	Condition Message(s)	Action to Take
		<p><b>Note</b> The percentage shown in this message is the used percentage plus the reserved percentage. For more information on "reserved" capacity see <b>"Capacity Managed"</b> (page 198).</p>
	"Hardware issues"	This message indicates that a data disk has errors or has been disabled. Click the condition message to go to the <a href="#">Data Centers</a> 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 <a href="#">Node Status</a> page to determine which disk is bad. You may need to replace the disk. For more information see <b>"Replacing a HyperStore Data Disk"</b> (page 482).
	"Service down"	A HyperStore service is down on a node. Click the condition message to go to the <a href="#">Data Centers</a> 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 <a href="#">Node Status</a> 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 <b>"Starting and Stopping Services"</b> (page 415)).
	"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 <b>"Automatic Stop of Writes to a Node at 90% Usage"</b> (page 158)
	"License usage X%" (where X is 90 or more)	<p>Contact Cloudian to request a new license. For more information see <b>"Licensing and Auditing"</b> (page 15).</p> <p><b>Note</b> Only your actually used capacity counts toward this percentage -- "reserved" capacity does not count toward this.</p>
	None	This status icon displays if the system has been disabled 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 <b>"Licensing and Auditing"</b> (page 15).

Cluster Status Icon	Condition Message(s)	Action to Take
	"System forecast 90% full in X days" (where X is 90 or less)	<p>Aggregate disk usage in the region is forecast to reach 90% of 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 <a href="#">Cluster Usage</a> 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 <b>"Capacity Monitoring and Expansion"</b> (page 71).</p> <div> <b>Note</b> The percentage shown in this message is the used percentage plus the reserved percentage. For more information on "reserved" capacity see <b>"Capacity Managed"</b> (page 198).         </div>
	"Node's Disks X% Full" (where X is 80 or more)	<p>A node's data disks are nearing capacity. Click the condition message to go to the <a href="#">Capacity Explorer</a> 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 <b>"Capacity Monitoring and Expansion"</b> (page 71).</p> <div> <b>Note</b> The percentage shown in this message is the used percentage plus the reserved percentage. For more information on "reserved" capacity see <b>"Capacity Managed"</b> (page 198).         </div>
	"Long proactive repair queue"	<p>A node has a long <b>"Proactive Repair"</b> (page 151) 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 <a href="#">Repair Status</a> 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 <a href="#">Node Status</a> 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 <a href="#">maintenance mode</a> or in a <a href="#">stop-write condition</a>; if so then in the <a href="#">Data Centers</a> page the node's icon will be blue.</p>
	"License usage X%" (where X is from 70 to 89)	<p>Storage usage in the system as a whole is at 70% of licensed usage or higher, but still below 90%. Contact Cloudbian to request a new license soon. For more information see <b>"Licensing and Auditing"</b> (page 15).</p>

Cluster Status Icon	Condition Message(s)	Action to Take
		<b>Note</b> Only your actually used capacity counts toward this percentage -- "reserved" capacity 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 <a href="#">Alerts</a> 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.

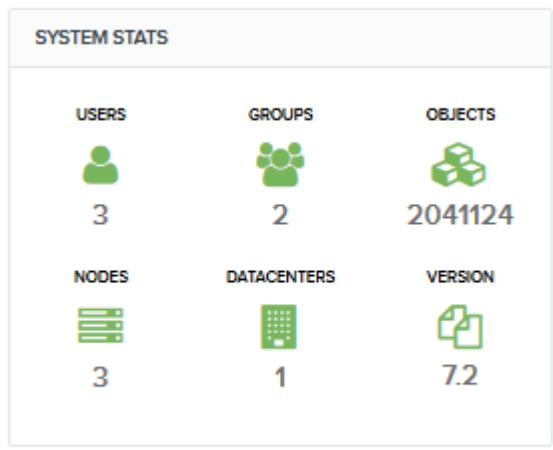
PUT latency is shown in blue and GET latency is shown in green. Hold your cursor over any point in time to view the specific PUT and GET latency 95th percentile figure for that point in time.

**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 Amazon S3, Amazon Glacier, or other HyperStore regions or systems do 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.

#### *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.

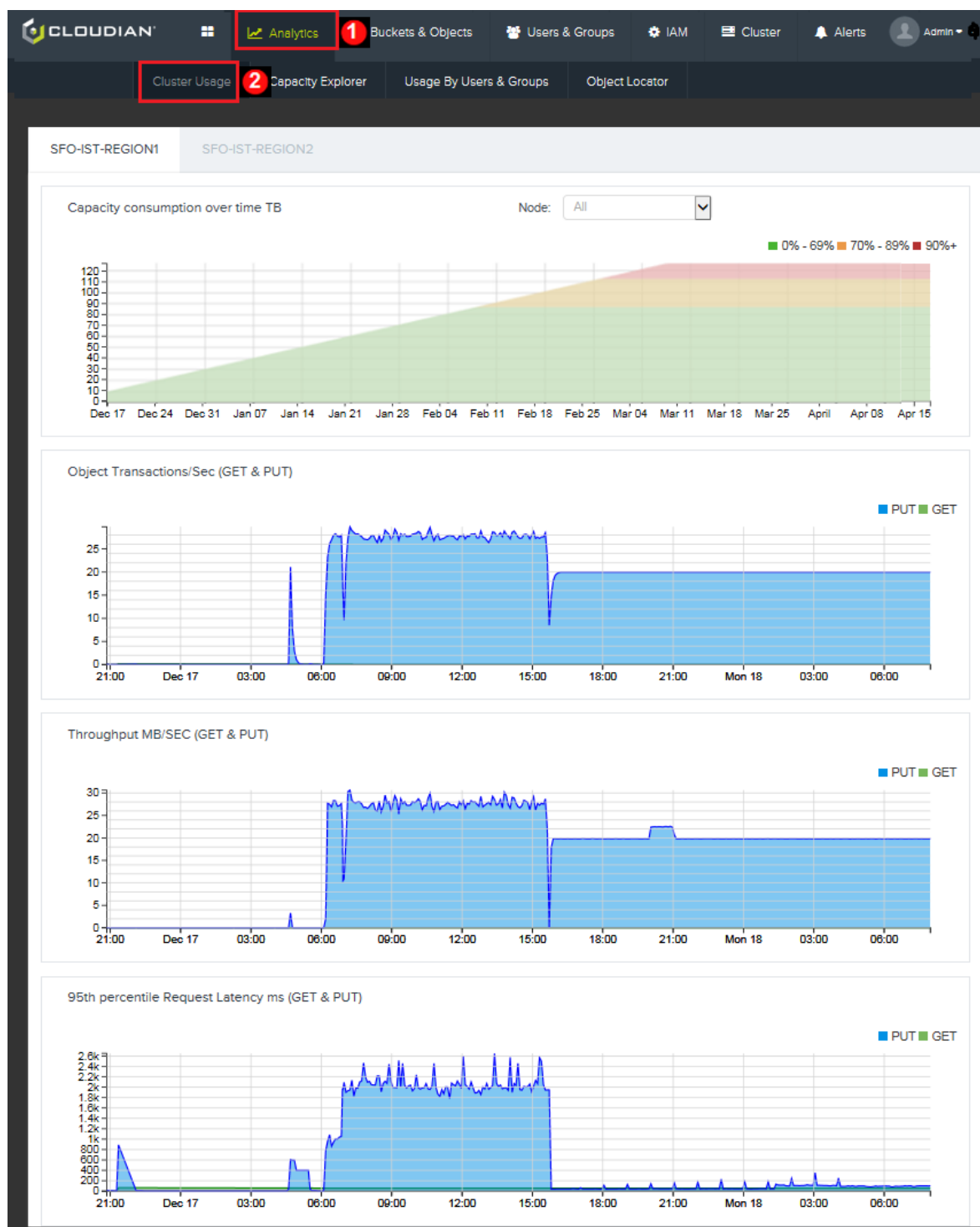
## 5.2. Analytics

The CMC's **Analytics** tab contains the following functions:

- [Cluster Usage](#)
- [Capacity Explorer](#)
- [Usage By Users & Groups](#)
- [Object Locator](#)

### 5.2.1. 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 71) 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 GBs, TBs, PBs, or EBs (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 198).

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 71).

#### *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 MB/sec (GET & PUT)*

This graph shows the S3 data throughput as KB or MB or GB 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.

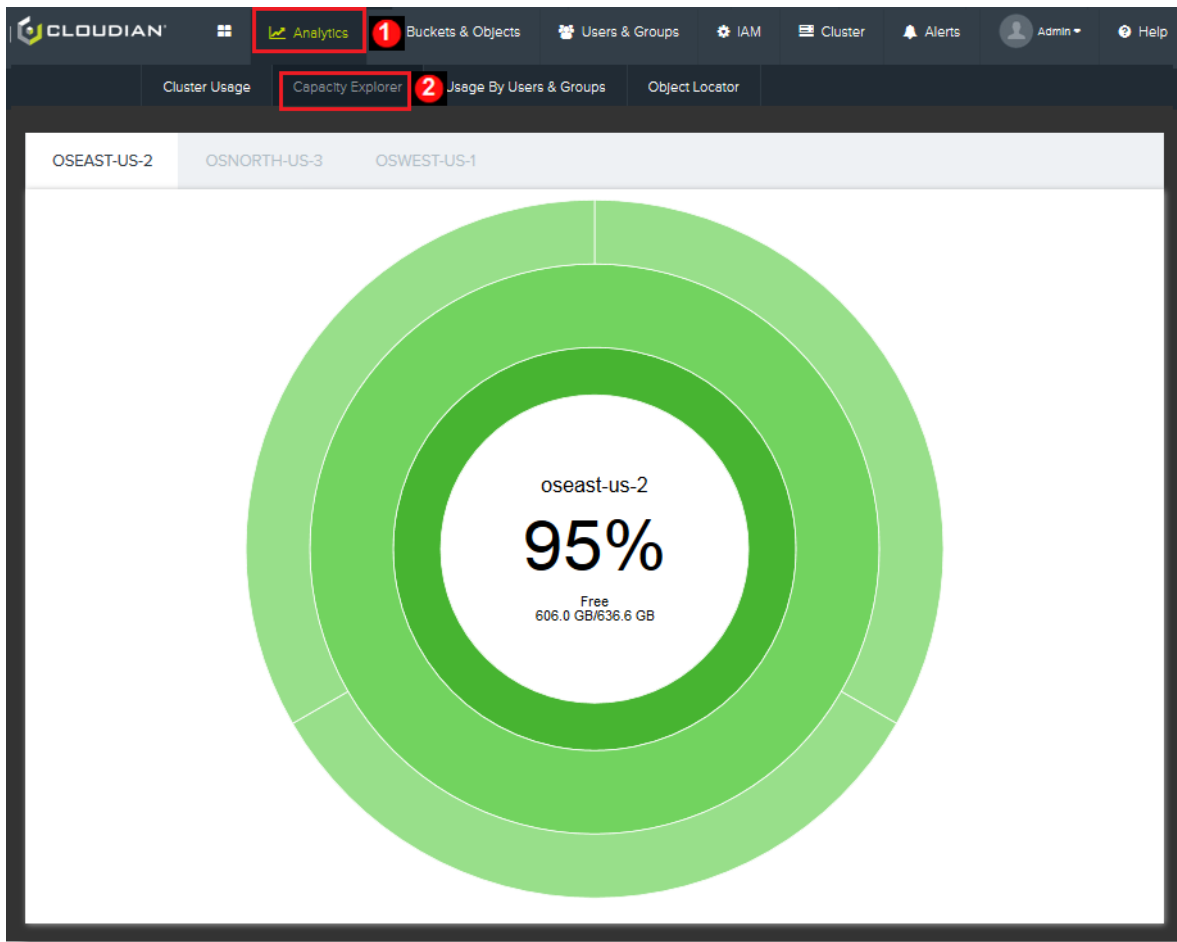
**Note** HEAD transactions are counted toward the GET stat, and POST transactions are counted toward the PUT stat.

**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.

### 5.2.2. 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 198).

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.

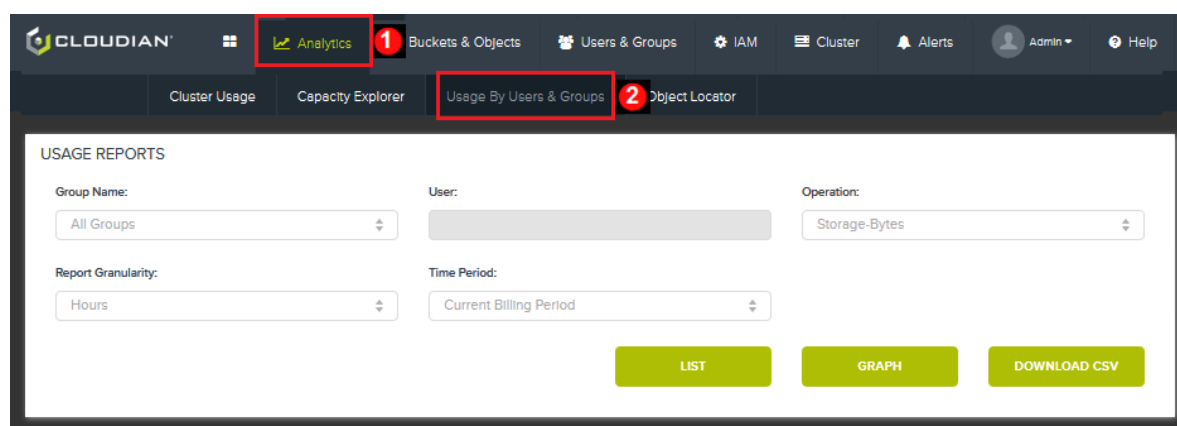
The measurements in the **Capacity Explorer** page are exclusively for HyperStore data disks -- on which S3 object data is being stored -- and do not address drives that are storing only the OS and the Cassandra and Redis databases (for system metadata). The HyperStore data disk utilization measured here is **raw** utilization and includes overhead for replication or erasure coding. For example a 1MB object that's replicated three times in the system results in 3MB of disk usage for the system.

**IMPORTANT !** See **"Capacity Monitoring and Expansion"** (page 71) 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.

### 5.2.3. Usage By Users & Groups

Path: **Analytics** → **Usage By Users & Group**



Supported tasks:

- **"Create a Usage Report"** (page 211)
- **"Review Usage Report Output"** (page 215)

**Note** For an overview of how the HyperStore system tracks service usage by groups and users, see **"Usage Reporting and Billing Feature Overview"** (page 138)

#### 5.2.3.1. 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.

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.

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, if you want the report to be limited 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 1MB object that's protected by 3X replication, this counts as 1MB toward Storage-Bytes -- not 3MB.
- *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"** (page 344) 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 Clouddian HyperStore 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.

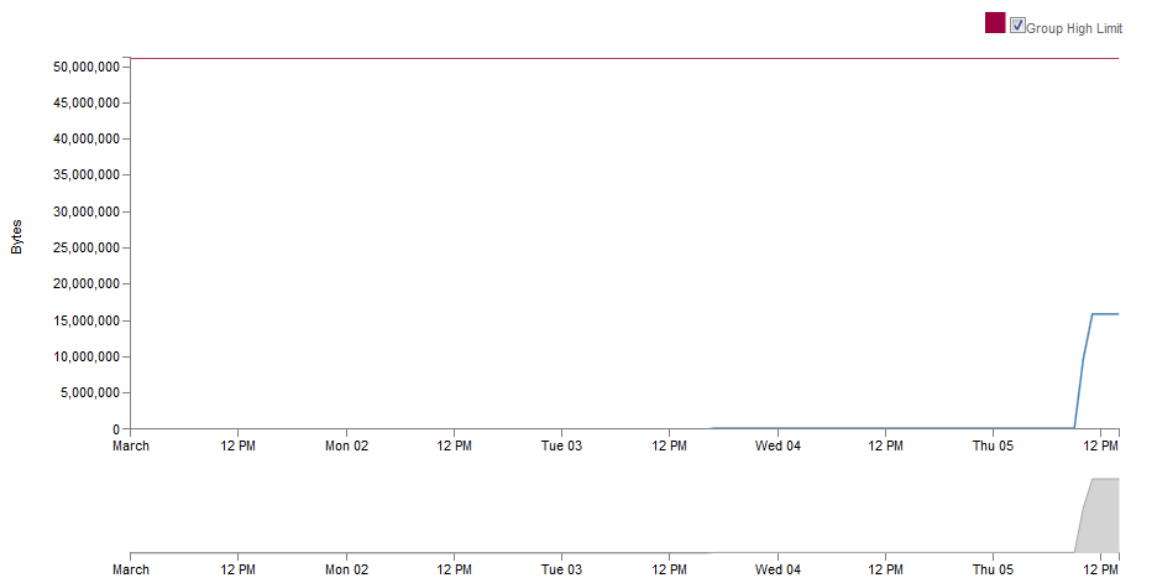
**Note** Graphs are not supported for reports for which the selected Operation category is "All Requests", reports for which the selected Report Granularity is "Raw", or reports for which the selected Region is "All". For more on graphing functionality see Manipulating Graph Reports.

- **Download CSV** to download a comma-separated value version of the report to your computer.

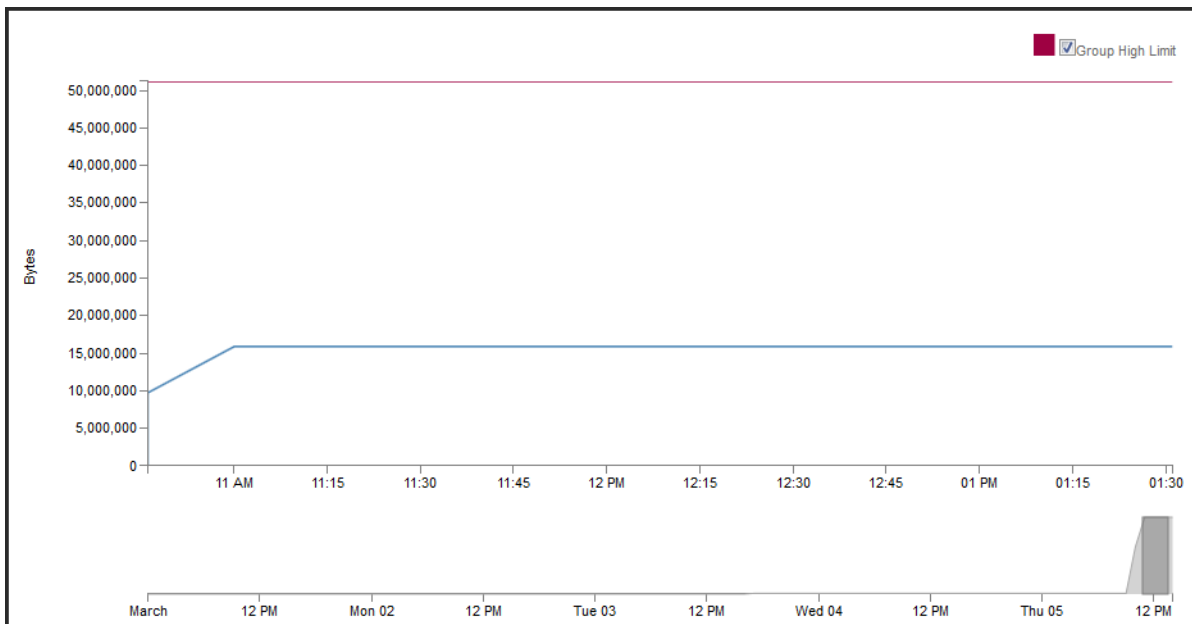
**Note** In CSV-formatted report output, all data size values are expressed in bytes (rather than KBs, MBs, or GBs).

#### 5.2.3.1.1. 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, KBs, MBs, or GBs, 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.

### 5.2.3.2. Review Usage Report Output

This topic clarifies the meaning of the data that you will see in your Cloudfire HyperStore S3 usage reports. Below is an example of a "List" style report, with hourly granularity.

RESULTS						
REGION	DATE/TIME	USER	GROUP	OPERATION	AVERAGE	MAXIMUM
region1	Sep-19-2015 10:00 -0400	PubsUser1	Pubs	Storage Bytes	136.1 K	136.1 K
region1	Sep-19-2015 09:00 -0400	PubsUser1	Pubs	Storage Bytes	136.1 K	136.1 K
region1	Sep-19-2015 08:00 -0400	PubsUser1	Pubs	Storage Bytes	136.1 K	136.1 K
region1	Sep-19-2015 07:00 -0400	PubsUser1	Pubs	Storage Bytes	161.9 K	170.5 K
region1	Sep-19-2015 06:00 -0400	PubsUser1	Pubs	Storage Bytes	170.5 K	170.5 K
region1	Sep-19-2015 05:00 -0400	PubsUser1	Pubs	Storage Bytes	170.5 K	170.5 K
region1	Sep-19-2015 04:00 -0400	PubsUser1	Pubs	Storage Bytes	170.5 K	170.5 K
region1	Sep-19-2015 03:00 -0400	PubsUser1	Pubs	Storage Bytes	170.5 K	170.5 K
region1	Sep-19-2015 02:00 -0400	PubsUser1	Pubs	Storage Bytes	170.5 K	170.5 K
region1	Sep-19-2015 01:00 -0400	PubsUser1	Pubs	Storage Bytes	170.5 K	170.5 K
region1	Sep-19-2015 00:00 -0400	PubsUser1	Pubs	Storage Bytes	170.5 K	170.5 K
region1	Sep-18-2015 23:00 -0400	PubsUser1	Pubs	Storage Bytes	170.5 K	170.5 K
region1	Sep-18-2015 22:00 -0400	PubsUser1	Pubs	Storage Bytes	170.5 K	170.5 K
region1	Sep-18-2015 21:00 -0400	PubsUser1	Pubs	Storage Bytes	170.5 K	170.5 K
region1	Sep-18-2015 20:00 -0400	PubsUser1	Pubs	Storage Bytes	170.5 K	170.5 K
region1	Sep-18-2015 19:00 -0400	PubsUser1	Pubs	Storage Bytes	170.5 K	170.5 K
region1	Sep-18-2015 18:00 -0400	PubsUser1	Pubs	Storage Bytes	170.5 K	170.5 K
region1	Sep-18-2015 17:00 -0400	PubsUser1	Pubs	Storage Bytes	170.5 K	170.5 K
region1	Sep-18-2015 16:00 -0400	PubsUser1	Pubs	Storage Bytes	170.5 K	170.5 K
region1	Sep-18-2015 15:00 -0400	PubsUser1	Pubs	Storage Bytes	170.5 K	170.5 K

20

NEXT

Starting from the left of "List" style reports, the first several columns have consistent meaning across the different types of usage reports:

#### *Region*

The HyperStore 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-14-2019 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-14-2019 <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-2019 <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 1MB object that's protected by 3X replication, this counts as 1MB toward Storage-Bytes -- not 3MB.

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.

## 5.2.4. Object Locator

Path: **Analytics** → **Object Locator**

The screenshot shows the Cloudian Object Locator interface. The top navigation bar has several tabs: 'Analytics' (highlighted with a red box and a red circle with '1'), 'Buckets & Objects', 'Users & Groups', 'IAM', 'Cluster', 'Alerts', 'Admin', and 'Help'. Below the navigation bar, there are sub-tabs: 'Cluster Usage', 'Capacity Explorer', 'Usage By Users & Groups', and 'Object Locator' (highlighted with a red box and a red circle with '2'). The main content area 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 S3 object

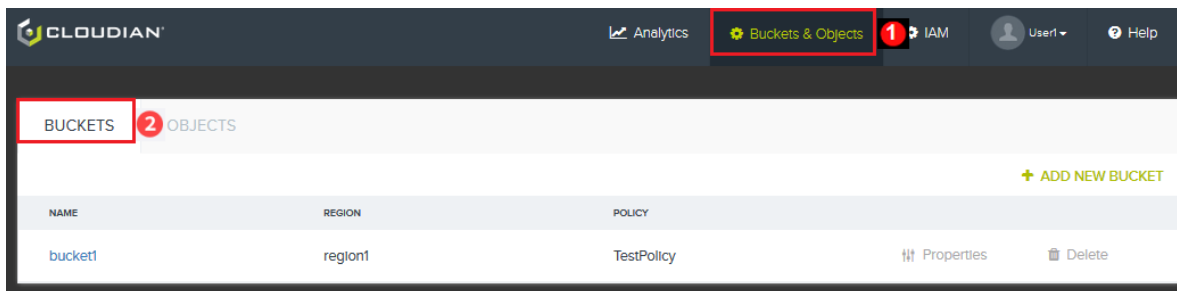
To view storage location information for an object:

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-2014.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](#).

## 5.3. Buckets

Path: **Buckets & Objects** → **Buckets**



Supported tasks:

- **"Add a Bucket"** (page 218)
- **"Set Bucket Properties"** (page 221)
- **"Delete a Bucket"** (page 244)

**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. 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.

### 5.3.1. 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.

**IMPORTANT !** Some atypical ways of organizing data within a bucket can result in sub-optimal performance for certain S3 operations on that bucket. For detail see "**Object Metadata Structure in Cassandra**" (page 167).

**Note** By default each user is allowed a maximum of 100 buckets. You can change this setting in the CMC's [Configuration Settings](#) page.

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 HyperStore service region in which you want the bucket to be created. If you have 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. Object Lock can guard against accidental or malicious deleting of objects after the objects have been uploaded to the bucket.

#### *Details about the Object Lock option*

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.
- 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 242).
- 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 255).
- 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 "**Setting Up Object Lock**" (page 123).
- 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

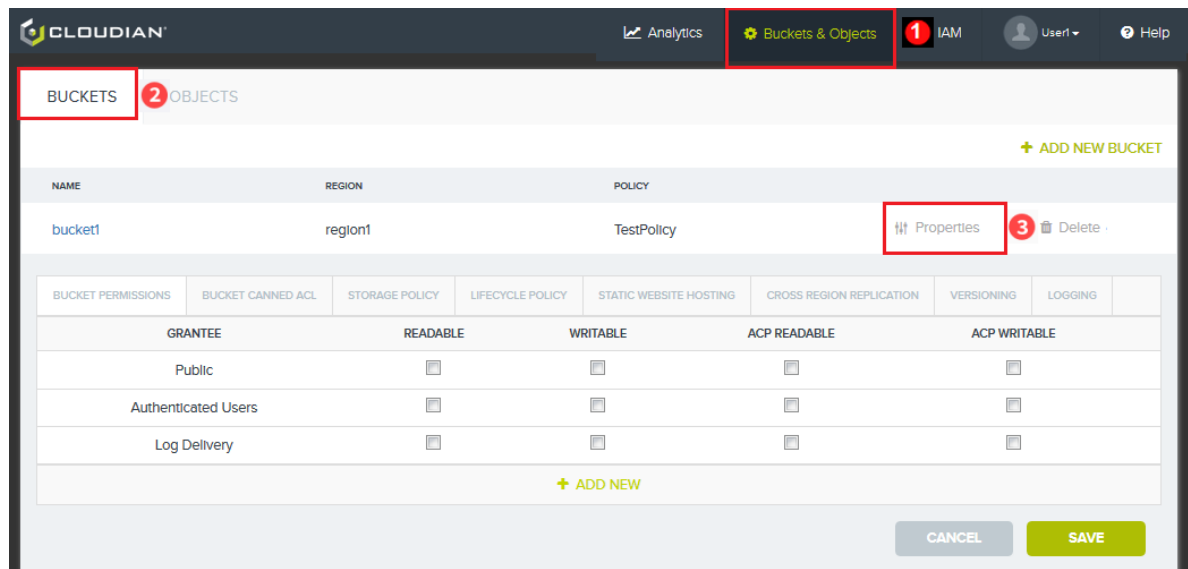
OBJECTS

+ ADD NEW BUCKET

NAME	REGION	POLICY		
<div><div> objectlockbucket2</div></div>	ks-region1	testp	<div> Properties</div>	<div> Delete</div>
unlockedbucket	ks-region1	testp	<div> Properties</div>	<div> Delete</div>

### 5.3.2. Set Bucket Properties

Path: **Buckets & Objects** → **Buckets** → **Properties**



In the **Buckets** page click the **Properties** link for the bucket that you want to work with.

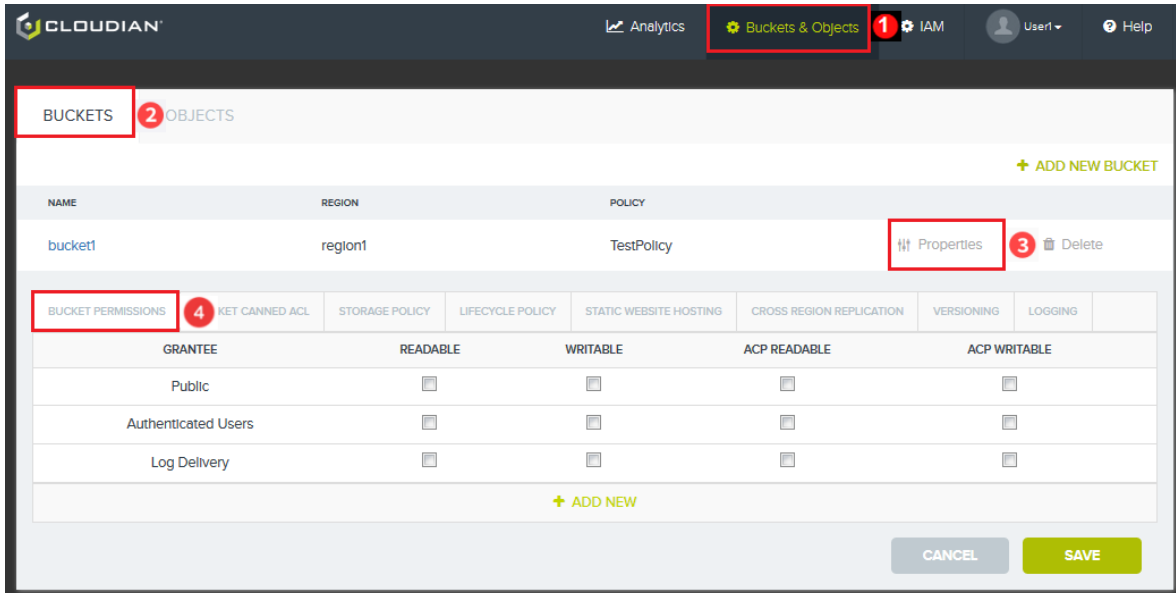
Supported tasks:

- **"Set Custom S3 Permissions for a Bucket"** (page 222)
- **"Set "Canned" S3 Permissions for a Bucket"** (page 224)
- **"View a Bucket's Storage Policy Information"** (page 226)
- **"Configure a Bucket Lifecycle Policy for Object Auto-Tiering or Expiration"** (page 227)
- **"Configure a Bucket as a Static Website"** (page 235)
- **"Configure Cross-Region Replication for a Bucket"** (page 237)

- **"Set Versioning for a Bucket"** (page 239)
- **"Set Logging for a Bucket"** (page 240)
- **"Configure Object Lock Properties for a Bucket"** (page 242)

### 5.3.2.1. Set Custom S3 Permissions for a Bucket

Path: **Buckets & Objects** → **Buckets** → **Properties** → **Bucket Permissions**



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 289).
- The CMC does not support creating bucket policies, but the HyperStore S3 Service API does support this, so you can use a third party S3 client application to create a bucket policy for a HyperStore bucket if you wish.
- The CMC supports configuring S3 ACLs for a bucket. You can do so in the **Bucket Properties** interface, by using either:
  - The [Bucket Canned ACL tab](#), to select from a small set of pre-defined S3 ACL packages for a bucket.
  - The **Bucket Permissions** tab, for more granular, customizable configuring of S3 ACLs. This is described in the instructions below.

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 S3 ACL settings

To configure custom S3 ACLs that grant bucket permissions to other users, follow the instructions below.

**Note** You can use the CMC to configure S3 ACL permissions on your bucket, but your grantees -- users to whom you grant S3 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 S3 ACL permissions by the bucket owner. So, your grantees can use S3 applications other than the CMC to exercise S3 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 bucket.

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 Cloudian HyperStore service endpoint (URL) and the bucket name.
Authenticated Users	All registered users of the Cloudian HyperStore service.
Log Delivery	Give this grantee the "Writable" permission if you want the Cloudian 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	<p>If you want to grant permissions to a specific Cloudian HyperStore group or user, click <b>Add New</b>. A new grantee row appears, with an empty grantee name box. In the name box do either of the following:</p> <ul style="list-style-type: none"> <li>• If the new grantee is a <b>group</b>, 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.</li> <li>• If the new grantee is a <b>user</b>, 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 -- <b>it cannot be an IAM user</b>.</li> </ul> <p><b>Note</b> Group and user name matching are case-sensitive, so be sure to use the correct case when specifying the grantee.</p> <p>To configure permissions for multiple group or user grantees, click <b>Add New</b> for each one.</p>

For each grantee you can select one or more of the following bucket permissions:

Permission	Description
Readable	<p>With this bucket permission, the grantee can read a list of files that are in the bucket.</p> <p><b>Note</b> The bucket permission "Readable" is not inherited by objects</p>

Permission	Description
	<p>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.</p> <p>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 S3 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 <b>Object Properties</b> interface to set per-object S3 ACL permissions (see "<b>Set Object Properties</b>" (page 248)).</p> <p>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 "<b>IAM</b>" (page 289)) or bucket policies (which are not supported by the CMC, but are supported by the HyperStore S3 Service API and by most third party S3 client applications.)</p>
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**.

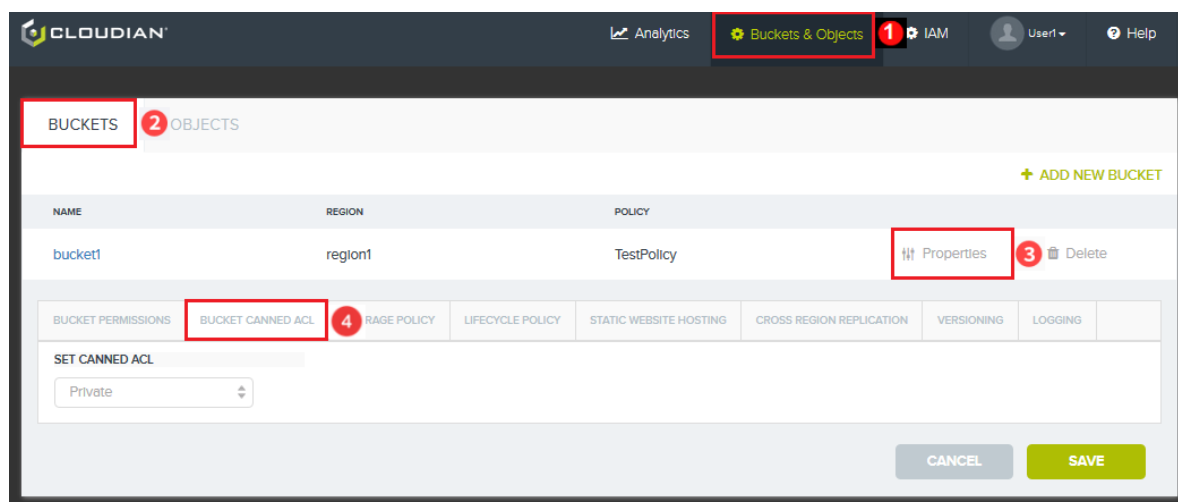
#### 5.3.2.1.1. Editing Custom Bucket Permissions

After you've configured custom S3 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).

#### 5.3.2.2. Set "Canned" S3 Permissions for a Bucket

Path: **Buckets & Objects** → **Buckets** → **Properties** → **Bucket 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 289).
- The CMC does not support creating bucket policies, but the HyperStore S3 Service API does support this, so you can use a third party S3 client application to create a bucket policy for a HyperStore bucket if you wish.
- The CMC supports configuring S3 ACLs for a bucket. You can do so in the **Bucket Properties** interface, by using either:
  - The **Bucket Canned ACL** tab, to select from a small set of pre-defined S3 ACL packages for a bucket. This is described in the instructions below.
  - The [Bucket Permissions tab](#), for more granular, customizable configuring of S3 ACLs.

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 S3 ACL settings

To set canned S3 ACLs that grant bucket permissions to other users, follow the instructions below.

**Note** You can use the CMC to configure S3 ACL permissions on your bucket, but your grantees -- users to whom you grant S3 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 S3 ACL permissions by the bucket owner. So, your grantees can use S3 applications other than the CMC to exercise S3 ACL permissions that you grant them.

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,

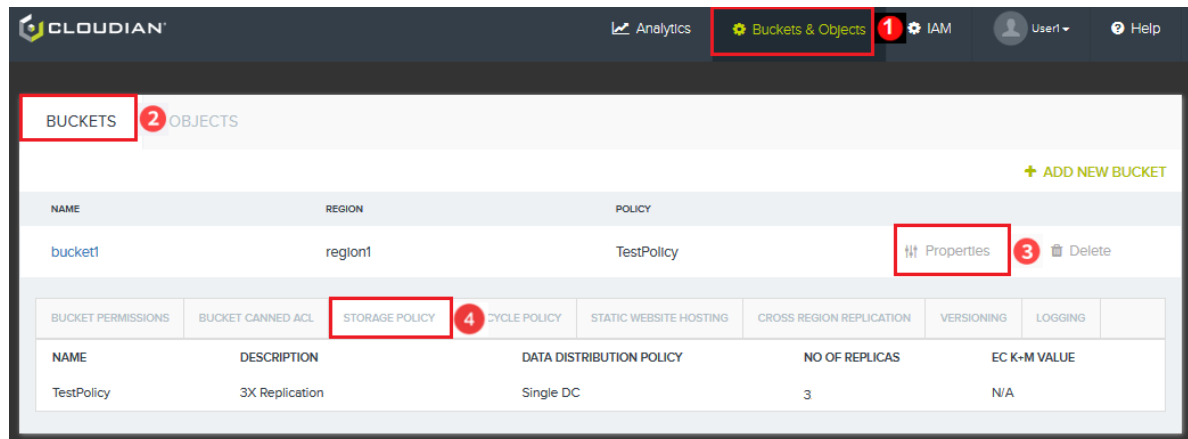
Canned ACL	Description
	<p>regardless of whether or not the user is a registered user of the Cloudian HyperStore service -- permission to read a list of files that are in the bucket.</p> <div> <p><b>Note</b> The bucket read permission is not inherited by objects within the bucket. Giving a grantee the 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.</p> <p>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 S3 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 <b>Object Properties</b> interface to set per-object S3 ACL permissions (see <b>"Set Object Properties"</b> (page 248)).</p> <p>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 <b>"IAM"</b> (page 289)) or bucket policies (which are not supported by the CMC, but are supported by the HyperStore S3 Service API and by most third party S3 client applications.)</p> </div>
Public Read Write	<p>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.</p> <p>See the Note in the "Public Read" description regarding the limits of the bucket read permission.</p>
Authenticated Read	<p>This bucket canned ACL grants all registered Cloudian HyperStore users permission to read a list of files that are in the bucket.</p> <p>See the Note in the "Public Read" description regarding the limits of the bucket read permission.</p>
Log Delivery Write	<p>This canned ACL enables the Cloudian HyperStore system to store the bucket's server access logs in the bucket.</p>

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.

### 5.3.2.3. View a Bucket's Storage Policy Information

Path: **Buckets & Objects** → **Buckets** → **Properties** → **Storage Policy**



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.

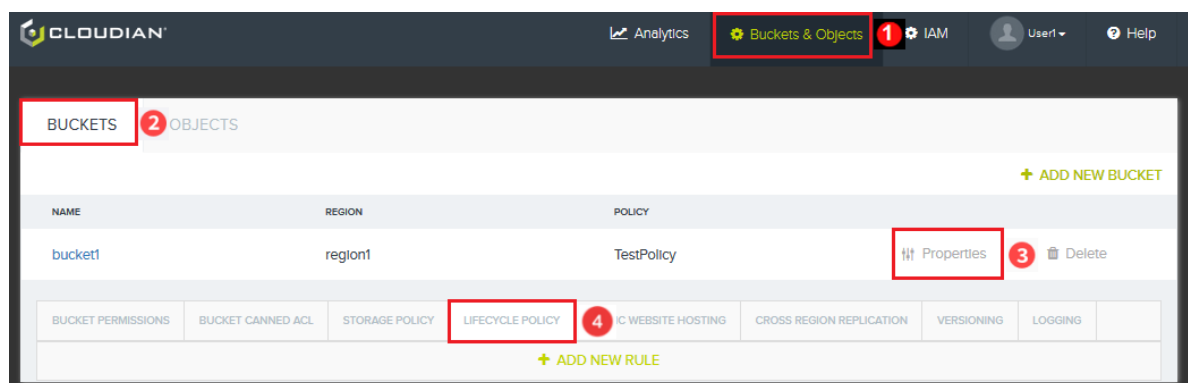
The display includes the storage policy name and description, whether data in the bucket is stored in one data center or multiple data centers, and whether data in the bucket is replicated or erasure coded.

You can control which user types can view this tab -- system admins, group admins, and/or regular users -- with the **"bucket.storagepolicy.showdetail.enabled"** (page 592) setting in *mts-ui.properties.erb*. By default all user types can view this tab.

**Note** You cannot change a bucket's storage policy.

#### 5.3.2.4. Configure a Bucket Lifecycle Policy for Object Auto-Tiering or Expiration

Path: **Buckets & Objects** → **Buckets** → **Properties** → **Lifecycle Policy**



The HyperStore system supports configuring bucket lifecycle policies so that objects are automatically moved (auto-tiered) to a different object storage system on a defined schedule, or automatically deleted (expired) on a defined schedule. You can also create combination lifecycle policies such that objects are auto-tiered to a different storage system and then later deleted from that system.

A lifecycle policy can apply to all objects in a bucket, or to a subset of objects as identified by object name prefix. You also have the option of creating multiple lifecycle rules for a bucket, with each rule applied to a different subset of objects based on object name prefix.

**Note** By default system configuration, the bucket lifecycle auto-tiering functions are disabled in the CMC. For instructions on enabling these functions, and controlling which auto-tiering options are presented to users, see **"Setting Up Auto-Tiering"** (page 180).

**Note** Auto-tiering is not supported for:

- \* Buckets that have an underscore in their name.
- \* Buckets for which [object lock](#) is enabled.
- \* Buckets that are source buckets for [cross-region replication](#).

To create 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/2016/*

This object prefix would apply to all objects with names that start with *Projects/archived/2016/*, such as *Projects/archived/2016/01/JanuaryOverview.docx* and *Projects/archived/2016/02/FebruaryOverview.docx*.

4. Select:
  - The "Enable Tiering" checkbox, if you want objects to be moved to a different 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 moves objects to a different storage system and then later deletes them from that system.
5. Configure the rule parameters:

## Object Tiering

OBJECT TIERING

SCHEDULE

☐ CURRENT VERSION
 ☐ PREVIOUS VERSION

OBJECT TIERING BUCKET LEVEL SETTING

DESTINATION

☒ Tier to AWS S3
 ☐ Tier to AWS GLACIER
 ☐ Tier to Google
 ☐ Tier to Azure
 ☐ Tier to Spectra
 ☐ Tier to Custom Endpoint

TIERING CREDENTIAL

Endpoint 
  
 Access Key:  Secret Key: 
  
 Bucket Name:

☐ Retain Local Copy
 ☐ Bridge Mode (Proxy)

GET REQUEST HANDLING

☒ Stream
 ☐ Require Restore

LIFECYCLE RULE BUCKET LEVEL SETTING

☐ Use Creation Date/Time
 ☒ Use Last Access Time

### Current Version

Select this 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 objects to occur on a particular date, choose the "After Date" radio button and enter the desired tiering date.
- If you want tiering of the objects to occur a certain number of days after the objects were last accessed (retrieved or modified) select the "Days After Last Access Date/Time" radio button and enter the number of days.
- If you want tiering of the objects to occur a certain number of days after the objects were created, at the bottom of the page choose the "Use Creation Date/Time" radio button, then back up in the scheduling section choose the "Days After Creation Date" radio button and enter the number of days.

### Previous Version

This option is applicable only if the bucket uses [versioning](#). Select this 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, 2018, 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).

### Destination

The destination that objects will be transitioned to.

Depending on system configuration (as described in **"Setting Up Auto-Tiering"** (page 180)), 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 S3
- AWS Glacier
- Google Cloud Storage
- Microsoft Azure
- Spectra BlackPearl

**Note** The destinations and corresponding endpoint URLs that the interface displays here are configurable in *common.csv*. For details see **"Configure Tiering Destinations"** (page 181).

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 an Amazon S3 or Google URL, or the URL of a different HyperStore region or system. Note that the system does not support specifying a Glacier, Azure or Spectra BlackPearl URL as a Custom endpoint -- the tiering operations will fail. To tier to one of those destination types select the radio button for that destination type, not the Custom Endpoint radio button.

If you enter a custom tiering URL be sure to include the *https://* part (or *http://* if the system does not support SSL).

**IMPORTANT !** After you Save your lifecycle policy, you will not be able to change the tiering destination for this bucket. If you create multiple auto-tiering rules for a single bucket (for different object prefixes), 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.

**IMPORTANT !** If you use a third party cloud service such as Amazon, Google, or Azure for storage of auto-tiered objects you (or your organization) will incur charges from that provider per the terms of your service agreement with them..

**Note Auto-tiering restrictions based on destination type:**

- \* The largest object size that can be auto-tiered to Amazon, Google, or other S3-compliant destinations is 100GB. If you want to tier objects larger than this, consult with Clouidian Support. This 100GB limit does not apply to tiering to Azure or Spectra BlackPearl.
- \* Tiering to Azure or Spectra BlackPearl is not supported for source buckets that have versioning enabled or that have had versioning enabled in the past.
- \* When auto-tiering to Spectra BlackPearl is used for a bucket, objects in the bucket will not be

auto-tiered unless they are larger than 5MB. Objects 5MB or smaller will remain in HyperStore. To change this limit consult with Cloudian Support.

#### *Tiering Credential*

For any tiering destination other than a system-default endpoint, you must enter your account security credentials for accessing the destination system (your Access Key and Secret Key -- or if the destination is Azure, your Account Name and Account Key). The system will then use these account credentials when it transitions objects from the bucket to the destination account.

**Note** If you create multiple auto-tiering lifecycle rules for a single bucket (for different object prefixes), all such rules must use the same destination account credentials.

#### *Bucket Name (or Container Name, for 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. In this case HyperStore will use this existing bucket as the tiering destination.
- The **name of a bucket that you want HyperStore to create in the destination system**, to use as the tiering destination. Be sure to choose a bucket name that is very likely to be unique in the destination system. If your supplied bucket name is not unique in the destination system, HyperStore will be unable to create the bucket and 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>`

**IMPORTANT !** After you Save your lifecycle policy, you will not be able to change the tiering destination bucket. If you create multiple auto-tiering rules for a single source bucket (for different object prefixes), 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** Azure uses the term "Container" rather than bucket, and in the CMC the field is "Container Name" rather than Bucket Name -- but the role of the field is the same as described above.

#### *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 30 days the local copy of the object will be automatically deleted.

If you do not select the "Retain Local Copy" checkbox, then 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 to retrieve a copy of the auto-tiered objects from the destination system upon your request.

*Bridge Mode (Proxy) (applicable only to 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 **all 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" 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.

For more information on bridge mode, see **"Bridge Mode"** (page 178).

*GET Request Handling (applicable only to AWS S3, Google, Azure, or Custom S3 destinations)*

If your tiering destination is AWS S3, Google, or Azure, 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 HyperStore system GETs the object from the tiering destination system and streams it through to the client. This is the default for tiering to AWS S3, Google, or Azure.
- 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"** (page 988) 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 **"Restore an Auto-Tiered Object"** (page 258).

**Note** If the bucket uses [versioning](#) you should use Stream 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 Auto-Tiered Object Versions"** (page 260)

**Note** If you create multiple auto-tiering lifecycle rules for a single bucket (for different object prefixes), all such rules must use the same GET request handling method.

For objects tiered to destinations other than AWS S3, Google, and Azure, the "Require Restore" method is always used.

## Object Expiration

OBJECT EXPIRATION
<b>SCHEDULE</b> <input type="checkbox"/> CURRENT VERSION <input type="checkbox"/> PREVIOUS VERSION <b>CLEAN UP EXPIRED OBJECT DELETE MARKERS AND INCOMPLETE MULTIPART UPLOADS</b> <input type="checkbox"/> CLEAN UP INCOMPLETE MULTIPART UPLOADS <input type="checkbox"/> CLEAN UP EXPIRED OBJECT DELETE MARKERS
<b>LIFECYCLE RULE BUCKET LEVEL SETTING</b> <input type="radio"/> Use Creation Date/Time <input checked="" type="radio"/> Use Last Access Time

### Current Version

Select this 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, current versions of 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 it will become a non-current version, and a "delete marker" will be created for the object (such that there is no longer a "current version" of the object). If you want to set a schedule for deleting non-current object versions from storage, you can do so by scheduling "Previous Version" expiration (described further below).

When setting a Current Version expiration schedule:

- If you want deletion of the objects to occur on a particular date, choose the "After Date" radio button and enter the desired deletion date.
- If you want expiration of the objects to occur a certain number of days after the objects were last accessed (retrieved or modified) select the "Days After Last Access Date/Time" radio button and enter the number of days.
- If you want expiration of the objects to occur a certain number of days after the objects were created, at the bottom of the page choose the "Use Creation Date/Time" radio button, then back up in the scheduling schedule choose the "Days After Creation Date" radio button and enter the number of days.

**Note** If you are configuring a lifecycle rule that includes both auto-tiering and expiration, you cannot have one action use Last Access Date/Time as the basis for scheduling while the other uses Creation Date/Time. This is true too if you plan to configure multiple lifecycle rules for a bucket (applying to different object prefixes): all lifecycle rules for a given bucket must use the same type of time reference point for objects -- either Last Access Date/Time or Creation Date/Time.

*Previous Version*

This option is applicable only if the bucket uses [versioning](#). Select this 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.

**Note** If the bucket uses "Object Lock", non-current object versions will not be deleted prior to the completion of their defined retention period. If a non-current object version's expiration date (based on your configured expiration schedule) falls before the end of the object version's lock period, the system will retain the non-current object version until the end of its lock 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 the HyperStore S3 storage system, 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.

*Clean Up Expired Object Delete Markers*

This option is applicable only if the bucket uses [versioning](#) and only if you haven't set a Current Version expiration schedule. If in a versioning-enabled bucket you delete the current version of an object, a "delete marker" takes the place of that object version while all of the object's older versions are retained in the system (and are retrievable). If all the older versions are subsequently expired (by execution of your expiration rule for Previous Versions), that orphaned delete marker remains. Select the "Clean Up Expired Object Delete Markers" option to have the system automatically delete a "delete marker" if all older versions of the object have been expired.

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.

**Note** You can create multiple lifecycle rules for a bucket (using the **Add New Rule** function each time), with each rule applying to a different object prefix. You cannot, however, have more than one lifecycle rule for the same object prefix.

**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 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 to another HyperStore region, any overwriting or deleting of objects should be done through the source bucket not the destination bucket.

#### 5.3.2.4.1. Editing, Disabling, or Deleting a Lifecycle Rule

To edit, 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.

BUCKETS		OBJECTS	
+ ADD NEW BUCKET			
NAME	REGION	POLICY	
bucket1	region1	Replication-3X	Properties Delete
<div> <div>BUCKET PERMISSIONS</div> <div>BUCKET CANNED ACL</div> <div>STORAGE POLICY</div> <div>LIFECYCLE POLICY</div> <div>STATIC WEBSITE HOSTING</div> <div>CROSS REGION REPLICATION</div> <div>VERSIONING</div> </div>			
ENABLE	RULE NAME		ACTION
<input checked="" type="checkbox"/>	Delete Logs After 90 Days		<a href="#">edit</a> <a href="#">delete</a>
<input checked="" type="checkbox"/>	Auto-Tier Documents After 1 Year		<a href="#">edit</a> <a href="#">delete</a>
+ ADD NEW RULE			
		<a href="#">CANCEL</a>	<a href="#">SAVE</a>

Here you can click the appropriate button or text to add a new rule for the bucket, or to edit, disable, or delete an existing rule for the bucket.

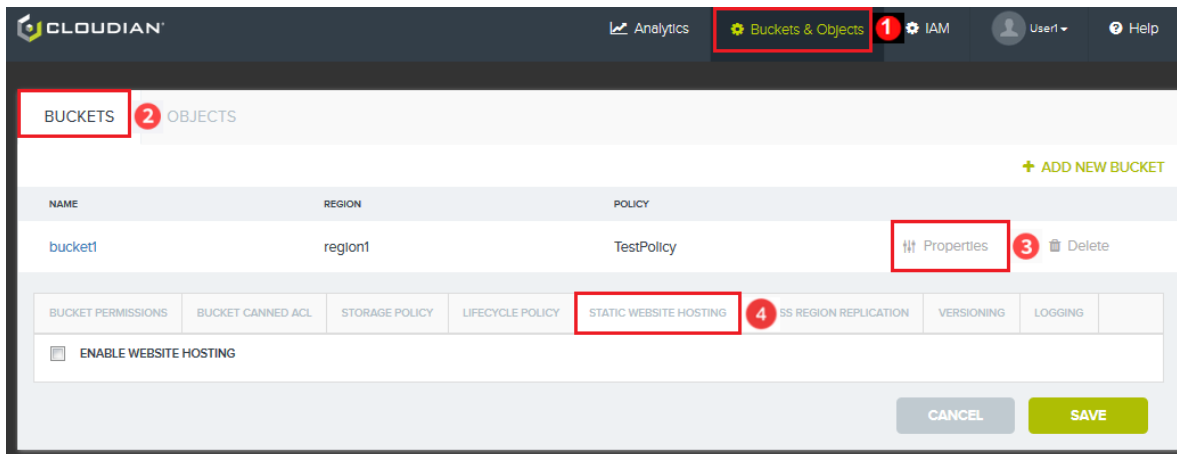
Note that adding, editing, disabling, or deleting a lifecycle rule changes the behavior of the bucket **from that point forward** — it doesn't change the past behavior. So for example, if you disable an auto-tiering rule that had been in place for your bucket, any objects that have already been transitioned to a tiering destination system will remain in that system.

Also, there are these restrictions:

- You cannot change the tiering destination system or tiering destination bucket for an existing rule. While other attributes of an existing rule are editable (such as the schedule), the **tiering destination system and destination bucket are not editable**.
- If you add a new lifecycle rule for a bucket, it must apply to a different object prefix than any existing rules. You **cannot have multiple rules for the same object prefix**.
- 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**.

#### 5.3.2.5. 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 (as opposed to 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" in the *HyperStore Installation Guide*.

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 248) 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 file is typically named "index.html".
4. Complete the "Error Document" field with the name of the file that should be served when site visitors trigger an HTTP 4xx error (such as by requesting an object that does not exist). You might name this file "404.html", for instance.
5. Click **Save**.

**Note** If you have not yet done so, you must upload both the index document file and the error document file into the **root level** of your bucket, and make sure their names correspond exactly to those you specified in the **Static Website Hosting** interface.

For a bucket configured as a static website, the root URL will be:

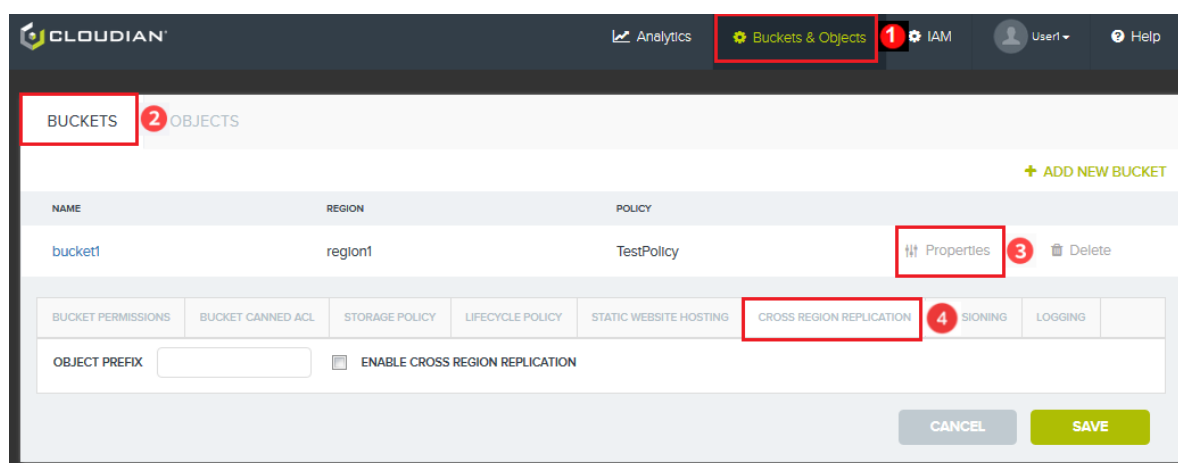
```
http://<bucketName>.<regionalS3WebsiteEndpoint>
```

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.

### 5.3.2.6. 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.

HyperStore also allows you to replicate to a destination bucket that's in the same HyperStore service region as the source bucket, if you want to. Note however that replicating from a source bucket to another bucket in the same region will not provide the geographical dispersion of data copies that replicating across regions does.

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.
- 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 bucket that has [object lock](#) enabled.
- Cross-region replication is not supported for a bucket that has a bucket [lifecycle policy](#) 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\_04-17-2018" will be replicated as "legal/docs/briefing\_04-17-2018" in the destination bucket).

3. Specify the "Destination Bucket" name. This must be a bucket that you own, within the HyperStore system.
4. Click **Save**.

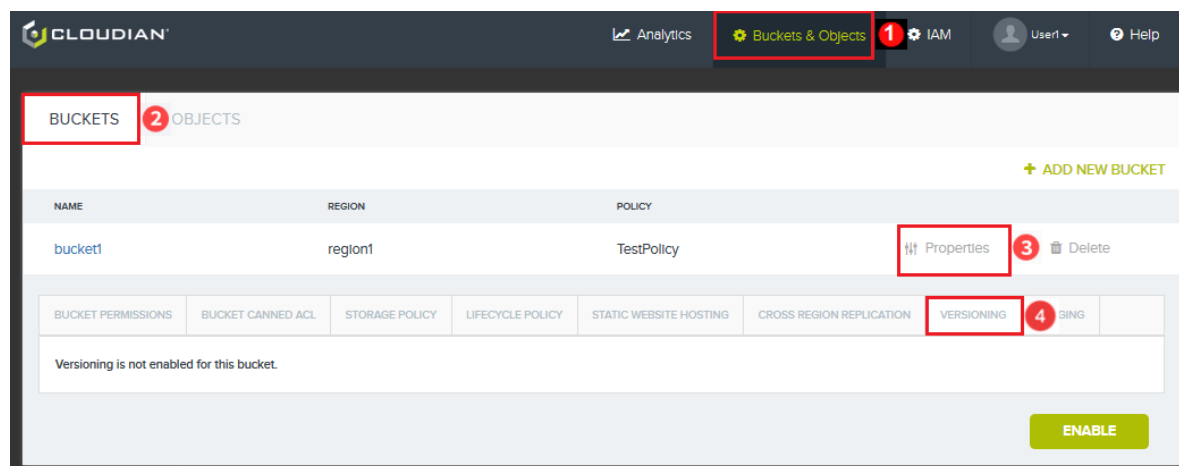
After cross-region replication is enabled on a source bucket and has been active, the system does support the option of editing the source bucket's configuration so that a different destination bucket is used. After that change, new data uploaded to the source bucket would replicate to the new destination 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.

**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.

### 5.3.2.7. Set 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 258).

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** You cannot enable versioning on a bucket that is configured for auto-tiering to Amazon Glacier or Spectra BlackPearl.

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 258) and **"Delete an Object"** (page 261).

### 5.3.2.7.1. 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. 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 for Object Auto-Tiering or Expiration"** (page 227).

### 5.3.2.7.2. 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:

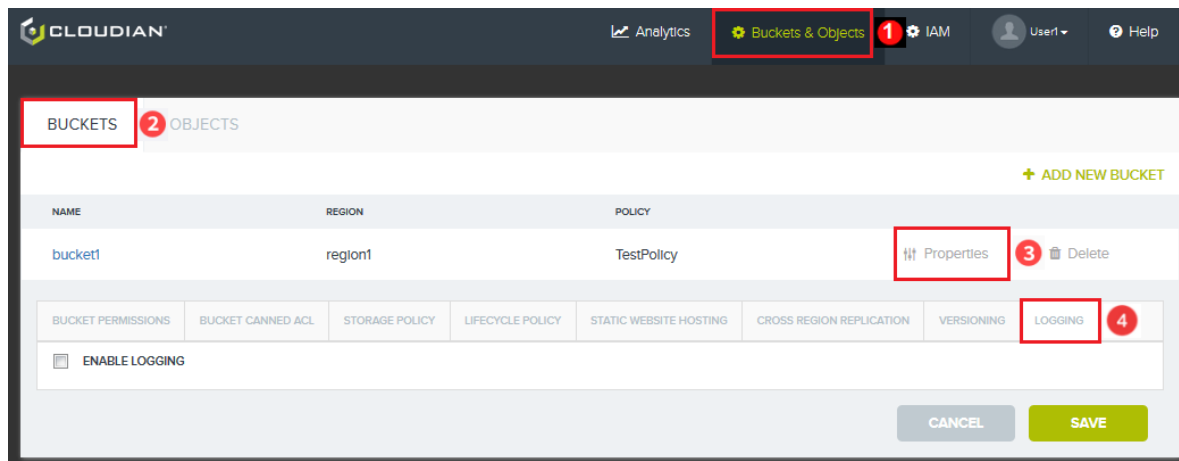
- For **new objects uploaded for the first time after versioning is suspended**, no versioning is applied. For example, if after suspending versioning you upload a new document *TerribleIdeas.docx*, and then later you revise that doc and upload the revised version, only the most current version will 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**.

### 5.3.2.8. Set Logging for a Bucket

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. You must be the owner of the destination bucket.

The content of the bucket log files is compliant with Amazon S3's bucket logging feature. For detail about the bucket log content, see the Amazon documentation topic [Server Access Log Format](#).

**Before you enable logging for a source bucket, you must establish the necessary access rights on the destination bucket** so that the HyperStore 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, 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** 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 CMC's [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.

### 5.3.2.8.1. 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:

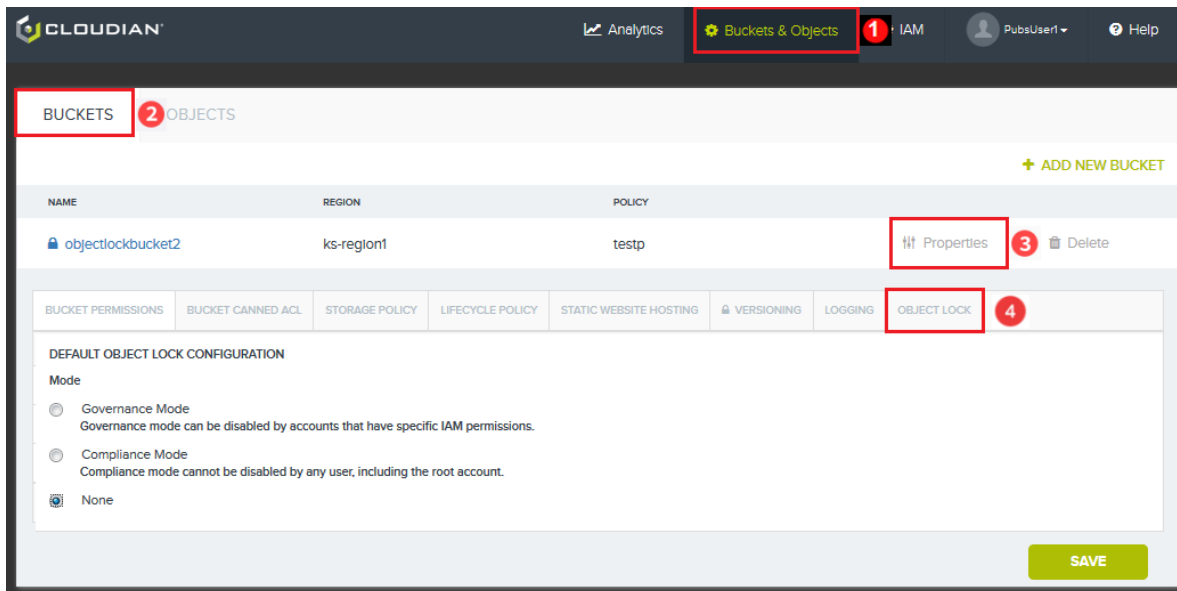
- Edit the bucket logging configuration (by changing the destination bucket or the log file name prefix)
- Disable logging for the bucket (by deselecting the "Enable Logging" checkbox)

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.

### 5.3.2.9. 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 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 (as described in "[Set Object Lock Attributes on an Object](#)" (page 255)).

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.

**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 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 255).

**Note** If IAM users that you have created will use third party S3 applications -- applications other than the CMC -- to access this bucket, then Governance Mode will allow locked objects to be deleted by IAM users to whom you have granted special permissions in regard to Object Lock. For detail see **Object Protection Under Governance Retention, Compliance Retention, and Legal Hold**. Note that the CMC does not allow IAM users to log in and access buckets or objects.

- **Compliance Mode:** With this mode, you cannot remove the lock from a locked object before the end of its retention period. Locked objects cannot be deleted by anyone -- not even by you as the bucket owner -- until the end of their retention period. Also with this mode, you cannot shorten the retention period for a locked object.

2. 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.

3. 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.

The screenshot shows the Cloudian console interface. The top navigation bar includes 'Analytics', 'Buckets & Objects', 'IAM', and a user profile 'PubsUser1'. The main content area is divided into 'BUCKETS' and 'OBJECTS' tabs. Under the 'OBJECTS' tab, the bucket 'objectlockbucket2' is selected. The table below lists objects in the bucket 'ks-region1 : objectlockbucket2'. Each object is preceded by a padlock icon, indicating it is locked. The objects are:

NAME	SIZE	LAST MODIFIED
EncryptionTiering.png	--	--
fe155c0a-d053-f75f-af1b-525400af5cfc	45.0 KB	Jun-01-2020 03:46 AM -0700
FirewallLogRotation.png	--	--
fe155c0a-9ddd-1ccf-af1b-525400af5cfc	14.2 KB	Jun-01-2020 03:48 AM -0700
fe155c0a-d048-d3cf-af1b-525400af5cfc	14.2 KB	Jun-01-2020 03:46 AM -0700
HardwareRequirements.png	--	--
fe155c0a-d001-6a8f-af1b-525400af5cfc	60.2 KB	Jun-01-2020 03:46 AM -0700

A 'DELETE' button is located at the bottom right of the object list.

**Note** Once you've set a default Object Lock configuration on the bucket, you still have the ability to later change that default configuration if you wish (by once again using the **Object Lock** tab will appear 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.

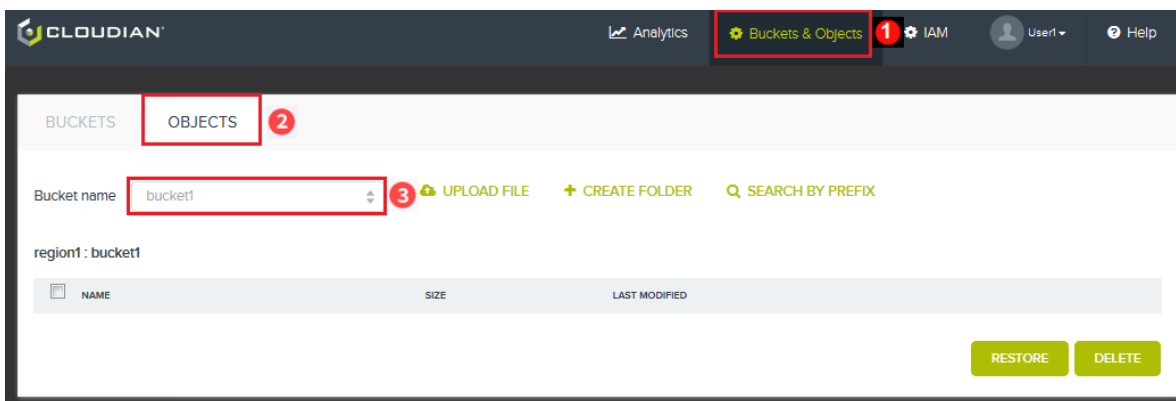
### 5.3.3. Delete a Bucket

To delete a bucket, first go to the [Objects](#) view and delete all the objects in the bucket. The system will not let you delete a bucket that has objects in it.

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.

## 5.4. 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, in the **Buckets** interface, clicking on a bucket name will bring you to the **Objects** interface for that bucket.)

**Note** The **Objects** tab does not display until you've created a bucket.

Supported tasks:

- "Create or Delete a "Folder"" (page 245)
- "Upload an Object" (page 246)
- "Set Object Properties" (page 248)
- "List or Search for Objects" (page 257)
- "Download an Object" (page 258)
- "Restore an Auto-Tiered Object" (page 258)
- "Delete an Object" (page 261)

**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 245).

### 5.4.1. 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 248). 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.

#### 5.4.1.1. 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*

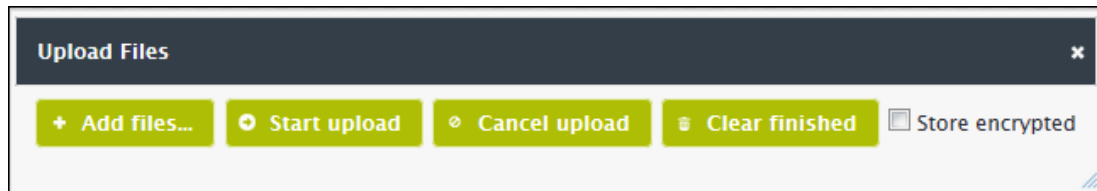
The CMC makes it easy to create such hierarchies by enabling you to create "folders". From the example above, through the CMC interface you could create a folder named "Documents", and then drill down into that folder and upload files named "resume.docx" and "schedule.docx". Behind the scenes, in the 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.

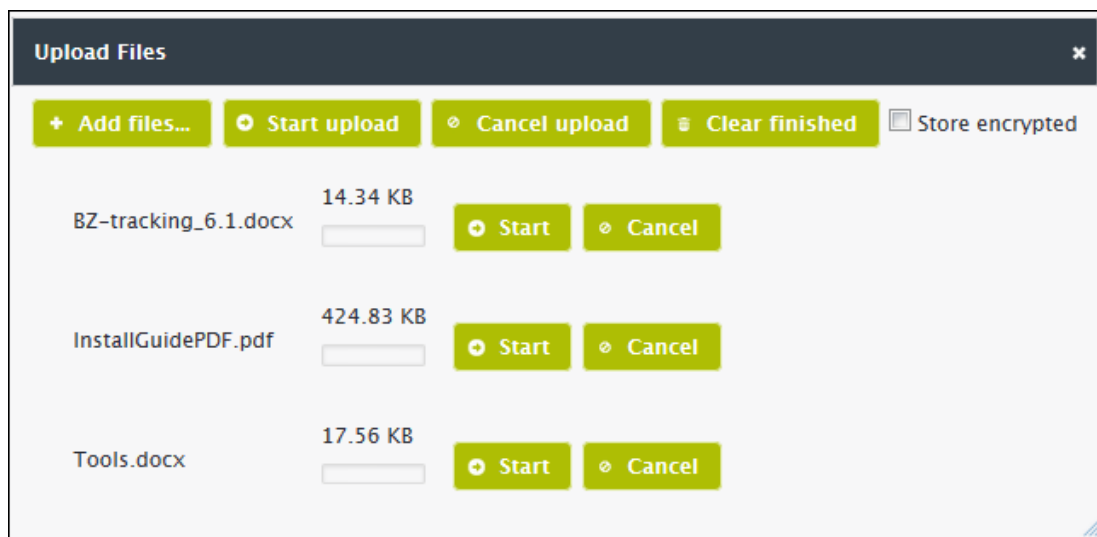
### 5.4.2. Upload an Object

**Note** If you haven't yet done so, you must **"Add a Bucket"** (page 218) 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.

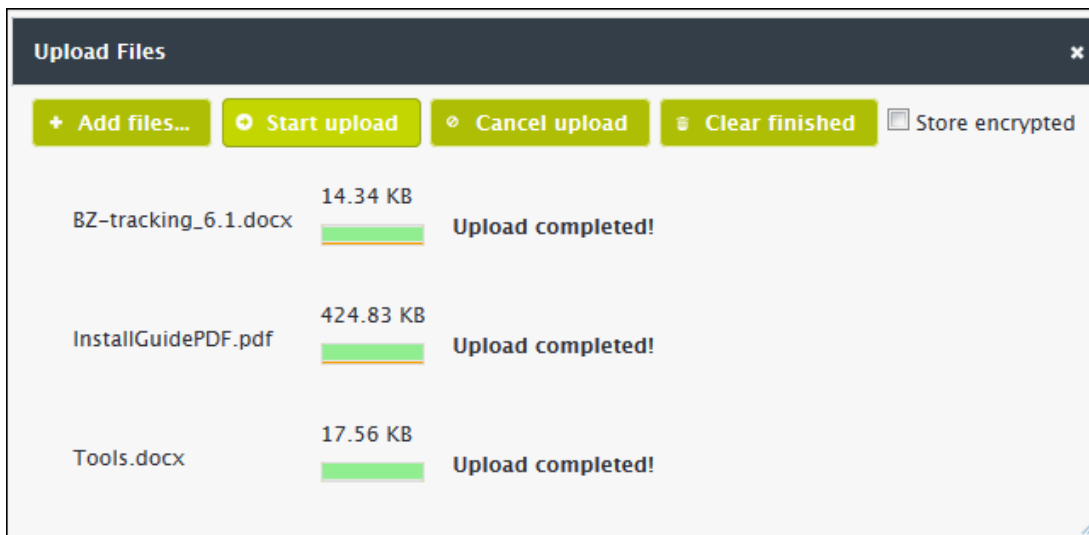
The maximum allowed S3 object name length — that is, the maximum length of the folder path plus file name, combined — is 1024 characters. For additional naming restrictions see **"Object Naming Restrictions"** (page 248). These additional restrictions pertain to unusual characters or character combinations and so are of no concern if you are using simple alphanumeric file names.

3. If you want the Cloudian HyperStore system to encrypt the file(s) in storage, click the "Store encrypted" checkbox. The file(s) will be encrypted using a system-generated encryption key. If an encrypted file is

subsequently downloaded, the system will automatically decrypt the file before transmitting it to the requesting client application.

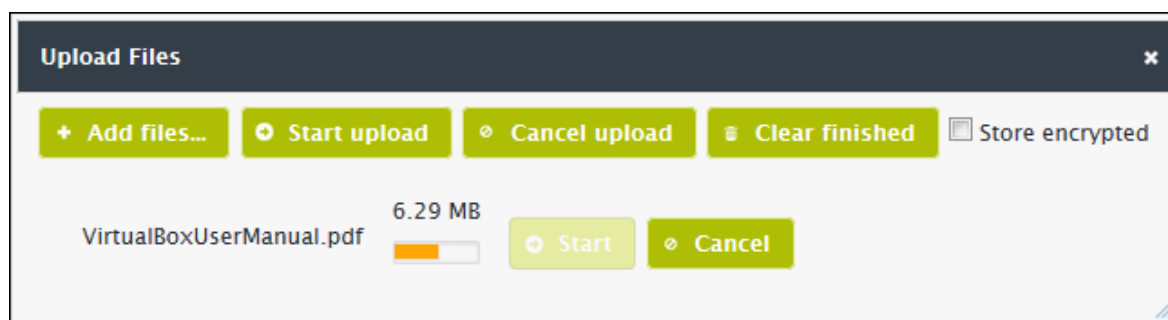
**Note** If you are uploading objects into a bucket that's using a storage policy that requires object encryption, then the objects will be automatically encrypted by the system regardless of whether or not you select the "Store encrypted" checkbox.

- 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.

By default the CMC's **Objects** page will display upload progress for a maximum of 1000 multipart upload objects simultaneously. This is configurable by the **"list.multipart.upload.max"** (page 583) property in [mts-ui.properties](#).

#### 5.4.2.1. Object Naming Restrictions

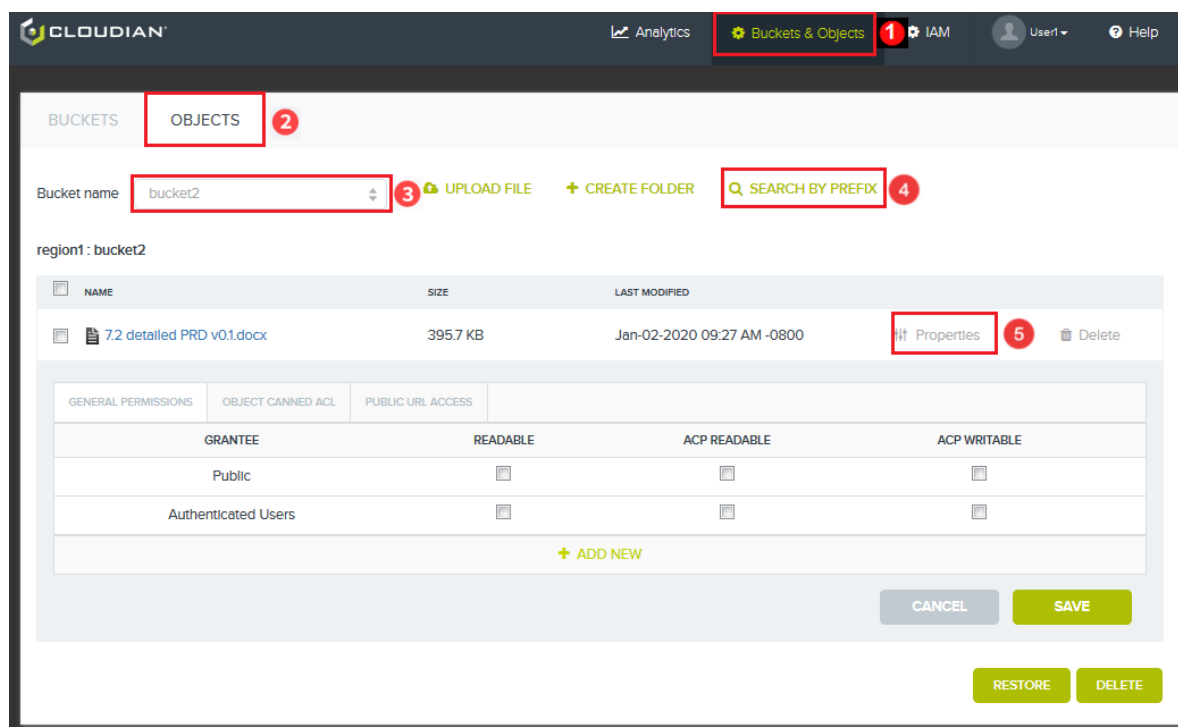
The following control characters cannot be used anywhere in an object name and will result in a 400 Bad Request response: 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A ("n"), 0x0B, 0x0C, 0x0D ("r"), 0x0E, 0x0F, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1A, 0x1B,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

#### 5.4.3. 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 249)
- **"Set "Canned" S3 Permissions on an Object"** (page 252)
- **"Set Public URL Permissions on an Object"** (page 254)
- **"Set Object Lock Attributes on an Object"** (page 255)

**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 **"PutBucketPolicy"** (page 977)), and some S3 client applications may support this, but the CMC currently does not.

#### 5.4.3.1. 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	Delete

GENERAL PERMISSIONS

GRANTEE	READABLE	ACP READABLE	ACP WRITABLE
Public	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Authenticated Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

+ 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 289).
- The CMC does not support creating bucket policies, but the HyperStore S3 Service API does support this, so you can use a third party S3 client application to create a bucket policy for a HyperStore bucket if you wish.
- The CMC supports configuring S3 ACLs for an object. You can do so in the **"Set Object Properties"** (page 248) interface, by using either:
  - The [Object Canned ACL tab](#), to select from a small set of pre-defined S3 ACL packages for an object.
  - The **General Permissions** tab, for more granular, customizable configuring of S3 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 S3 ACL settings

To configure custom S3 ACLs that grant object permissions to other users, follow the instructions below.

**Note** You can use the CMC to configure S3 ACL permissions on your object, but your grantees -- users to whom you grant S3 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 they have been granted 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 S3 ACL permissions by owner. So, your grantees can use S3 applications other than the CMC to exercise S3 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 Cloudian HyperStore service endpoint (URL) and the bucket name and object name.

Grantee	Description
Authenticated Users	All registered users of the Cloudian HyperStore service.
Specific group or user	<p>If you want to grant permissions to a specific Cloudian HyperStore group or user, click <b>Add New</b>. A new grantee row appears, with an empty grantee name box. In the name box do either of the following:</p> <ul style="list-style-type: none"> <li>If the new grantee is a <b>group</b>, 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.</li> <li>If the new grantee is a <b>user</b>, 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 -- <b>it cannot be an IAM user</b>.</li> </ul> <p><b>Note</b> Group and user name matching are case-sensitive, so be sure to use the correct case when specifying the grantee.</p> <p>To configure permissions for multiple group or user grantees, click <b>Add New</b> for each one.</p>

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 221).

After making the permissions selections, click **Save**.

#### 5.4.3.1.1. 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.

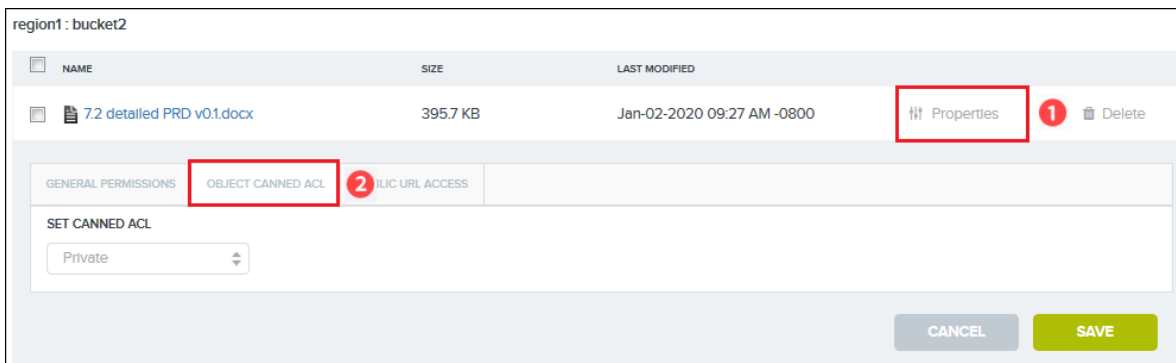
#### 5.4.3.1.2. Editing Custom Object Permissions

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).

#### 5.4.3.2. 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 289).
- The CMC does not support creating bucket policies, but the HyperStore S3 Service API does support this, so you can use a third party S3 client application to create a bucket policy for a HyperStore bucket if you wish.
- The CMC supports configuring S3 ACLs for an object. You can do so in the **"Set Object Properties"** (page 248) interface, by using either:
  - The **Object Canned ACL** tab, to select from a small set of pre-defined S3 ACL packages for an object. This is described in the instructions below.
  - The [General Permissions tab](#), for more granular, customizable configuring of S3 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 S3 ACL settings

To set canned S3 ACLs that grant object permissions to other users, follow the instructions below.

**Note** You can use the CMC to configure S3 ACL permissions on your object, but your grantees -- users to whom you grant S3 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 they have been granted 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 S3 ACL permissions by owner. So, your grantees can use S3 applications other than the CMC to exercise S3 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 Clouidian HyperStore service -- permission to read (open or download) the object.
Authenticated Read	This bucket canned ACL grants all registered Clouidian 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 221).

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.

#### 5.4.3.2.1. 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.

#### 5.4.3.3. 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 which you can provide to whoever you want to have access to the object.

**Note** You cannot set public URL permissions on an object that the system has encrypted with a user-provided encryption key. By contrast, public URL permissions are supported for files that the system has encrypted with a system-generated encryption key, which is the more typical encryption method.

To set public URL permissions on an object, in the CMC's **"Set Object Properties"** (page 248) 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=5cb6eee4356624d883ef8&Expires=1493535600&Signature=9wke%2FoCdGhCNJxsDMACIFVTsxf%3D&x-amz-pt=MDAxZTE1NDExNDkxNTk0NDk2Mzk2](http://s3-region1.mycloudianhyperstore.com/bucket1/6.2_AutoTiering_CustomEndpoint.png?AWSAccessKeyId=5cb6eee4356624d883ef8&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]://<defaultS3Domain>/<bucketName>/<objectName>?AWSAccessKeyId=<accessKeyOfObjectOwner>
&Expires=<expiryTimeInUnix>&Signature=<signatureString>&x-amz-
pt=<uniqueIDforMaxDownloadTracking>
```

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.

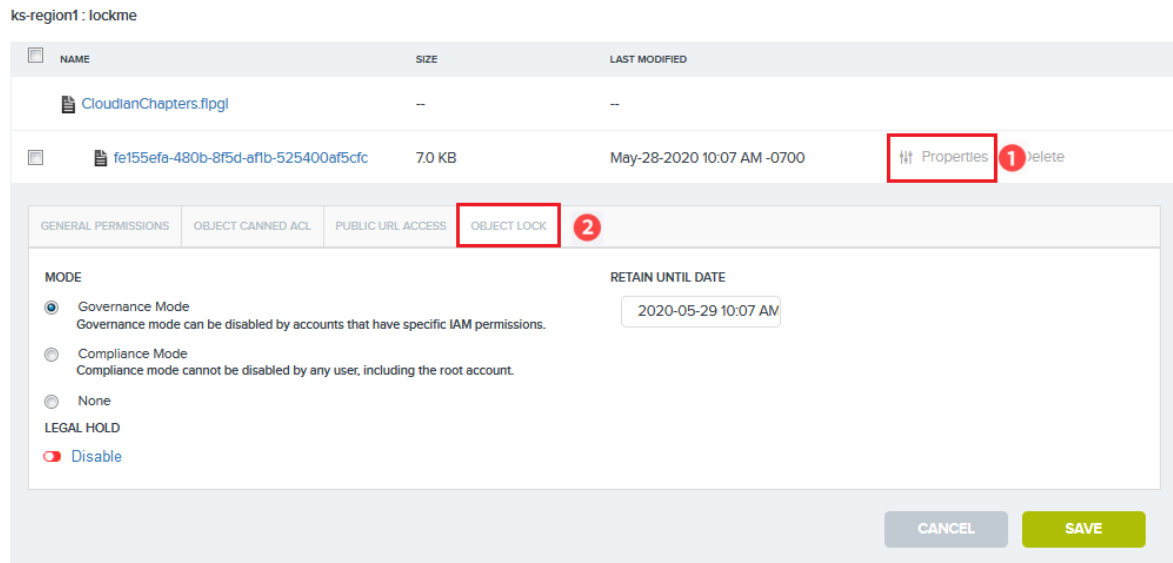
**Note** For the **Mail To** feature to work, your browser must be configured to be able to open your default web-based email client.

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** 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.

#### 5.4.3.4. 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 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 242).
  - **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).
  - **Legal Hold.** Legal Hold prevents the object from being deleted 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** If IAM users that you have created will use third party S3 applications -- applications other than the CMC -- to access this bucket, then the Legal Hold could be removed by IAM users to whom you have granted the necessary special permissions. For detail see **Object Protection Under Governance Retention, Compliance Retention, and Legal Hold**. Note that the CMC does not allow IAM users to log in and access buckets or objects.

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.

#### 5.4.4. List or Search for Objects

If you have many objects in your bucket, in the [Objects](#) interface the objects will be listed alphabetically, 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.

Alternatively, 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/2019/InstallerDemo* -- Retrieves all objects that start with prefix *Video/2019/InstallerDemo* (or in common file system terminology, all files that are under the *Video/2019* "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/2019/InstallerDemo\_2019-05-22.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 245).

**Note** The search matching is **case sensitive**. Be sure to use the correct case when entering your search term.

#### 5.4.4.1. Objects That Indicate an 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: 404" 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.

- 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.

#### 5.4.5. 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 ["Restore an Auto-Tiered Object"](#) (page 258).

##### 5.4.5.1. 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** (if object versions are not already showing in the object list). 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.

#### 5.4.6. Restore an Auto-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 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 HyperStore S3 service.

**Note** The dialog also presents you the option to choose between Bulk retrieval, Standard retrieval, and Expedited retrieval, but **in the current HyperStore release these retrieval settings have no effect** on how the retrieval is implemented.

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 S3 service (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 Auto-Tiered Object Versions"** (page 260).

**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.

#### 5.4.6.1. Icons for Transitioned or Restored Objects

In your object list display, the document icon to the left of object names will indicate if an object is transitioned, transitioning, restored, or restoring. The icons feature right-pointing or left-pointing red or blue arrows.

Icon	Meaning
	Object is <b>in the process of transitioning</b> to a remote tiered storage system.
	Object <b>has been transitioned</b> to a remote tiered storage system.
	A copy of the transitioned object is <b>in the process of being restored</b> to the local HyperStore S3 service. Download of the object is not yet supported.
	A copy of the transitioned object <b>has been restored</b> to the local HyperStore S3 service. Download is supported.

#### 5.4.6.2. Deleting an Auto-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 any other object. You do not need to restore the object first.

**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 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 to another HyperStore region, any overwriting or deleting of objects should be done through the source bucket not the destination bucket.

#### 5.4.6.3. Restoring Auto-Tiered Object Versions

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 S3 service.

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 HyperStore S3 service (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.

### 5.4.7. 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 DELETE Object API method or the S3 DELETE Multiple Objects 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 `"cloudian.delete.queue.poll.interval"` (page 566) property in [mts.-properties.erb](#).

#### 5.4.7.1. 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.

#### 5.4.7.2. Deleting an Auto-Tiered Object

See **"Deleting an Auto-Tiered Object"** (page 260).

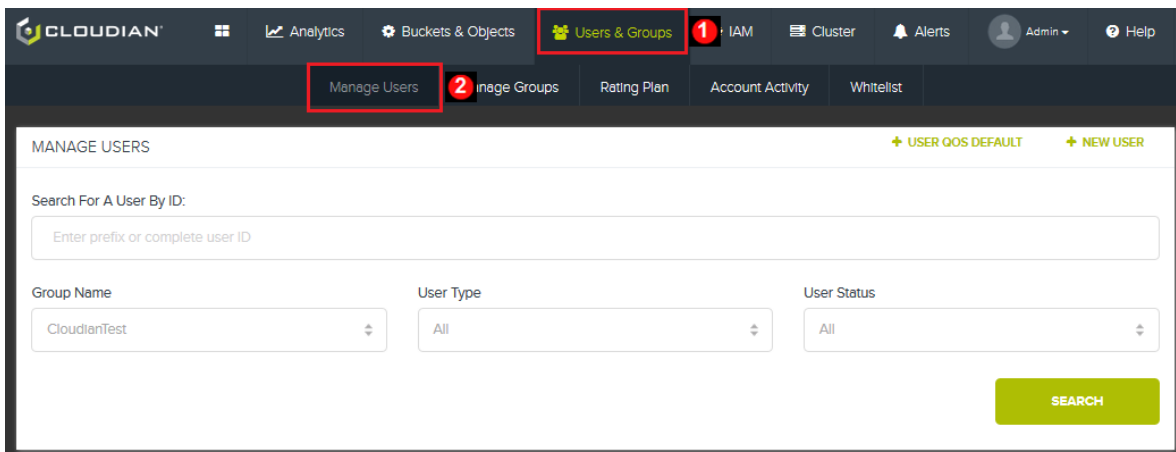
## 5.5. Users & Groups

The **Users & Groups** tab contains the following functions:

- [Manage Users](#)
- [Manage Groups](#)
- [Rating Plan](#)
- [Account Activity](#)
- [Whitelist](#)

### 5.5.1. Manage Users

Path: **Users & Groups** → **Manage Users**



Supported tasks:

- **"Add a User"** (page 263)
- Work with existing users:
  - **"Retrieve a User or List of Users"** (page 265)
  - **"Edit or Suspend a User"** (page 266)
  - **"View or Edit a User's Security Credentials"** (page 268)
  - **"Set Quality of Service (QoS) Controls"** (page 285)
  - **"Manage a User's Stored Objects"** (page 269)
  - **"Delete a User"** (page 270)

### 5.5.1.1. 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. Note however that the user name 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 CMC's [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, and underscores are allowed. No spaces or special characters.
- By default the maximum length is 64 characters. This maximum is configurable by the setting *common.csv*: "**cloudian\_userid\_length**" (page 514).
- 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
- \* Must start with a letter
- \* Cannot end with an underscore

#### *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 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.

#### *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.

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 [common.csv](#), see "**user\_password\_min\_length**" (page 527) and the subsequent settings.

#### *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 configurable by the setting *common.csv*: "**cloudian\_userid\_length**" (page 514).)

\* 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**.

### 5.5.1.2. Retrieve a User or List of Users

In the CMC's [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	<a href="#">Edit</a> <a href="#">Security Credentials</a> <a href="#">Set QoS</a> <a href="#">View User Data</a> <a href="#">Delete</a>
PubsUser1	Pubs	User	Active	<a href="#">Edit</a> <a href="#">Security Credentials</a> <a href="#">Set QoS</a> <a href="#">View User Data</a> <a href="#">Delete</a>
PubsUser2	Pubs	User	Active	<a href="#">Edit</a> <a href="#">Security Credentials</a> <a href="#">Set QoS</a> <a href="#">View User Data</a> <a href="#">Delete</a>

You can then take any of the following actions regarding the retrieved user(s):

- **"Edit or Suspend a User"** (page 266)
- **"View or Edit a User's Security Credentials"** (page 268)
- **"Set Quality of Service (QoS) Controls"** (page 285) for a user
- **"Manage a User's Stored Objects"** (page 269)
- **"Delete a User"** (page 270)

### 5.5.1.3. Edit or Suspend a User

1. Use the CMC's [Manage Users](#) page to **"Retrieve a User or List of Users"** (page 265).
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.

The screenshot shows the 'Edit User' panel for user 'PubsUser1'. The panel includes the following elements:

- User Type:** A dropdown menu currently showing 'User'.
- Canonical ID:** A text field displaying the ID 'a1b8a09e4ee421ffc89b00c50272f22f'.
- Rating Plan:** A dropdown menu currently showing 'Default-RP'.
- More:** A blue link with a downward arrow to expand more options.
- Active User:** A checkbox that is currently checked.
- Buttons:** 'CANCEL' (grey) and 'SAVE' (green) buttons at the bottom right.

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.

- Active User checkbox checked — User is active. User has access to the storage service and the CMC.
- Active User checkbox unchecked — User is suspended. 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.

**Note** You cannot change a regular user's or group administrator's User Type to System Admin. Also, you cannot change a system administrator's User Type to anything other than System Admin.

#### *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.

#### *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.csv*: "**cloudian\_userid\_length**" (page 514).) 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.

When LDAP is enabled for a user, the CMC authenticates the user against their password stored in the LDAP server. The password that's stored in the CMC for such a user is simply a system-generated random string that is not actually used for authentication.

4. Click **Save**.

## 5.5.1.4. View or Edit a User's Security Credentials

1. Use the CMC's [Manage Users](#) page to **"Retrieve a User or List of Users"** (page 265).
2. In the "Actions" column for the user that you want to edit, click **Security 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 *	View Secret Key Inactivate Delete

\* Active Access Key

**CREATE NEW KEY**

Close

**Note** If the user is an LDAP-based user, in the CMC "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**.

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 [common.csv](#), see **"user\_password\_min\_length"** (page 527) and the subsequent 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.

#### 5.5.1.5. 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, log into the Puppet master node, open the configuration file [common.csv](#), and set **"cmc\_view\_user\_data"** (page 541) to *true* (to allow this capability to system administrators and also group administrators [in regard to their own group members]) or to *SystemAdmin* (to allow this capability only to system administrators). Then [use the installer to push your changes out to the cluster and to restart the CMC](#). You can then access users' data via the CMC as follows:

1. Use the [Manage Users](#) page to **"Retrieve a User or List of Users"** (page 265).
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 218)
- **"Set Bucket Properties"** (page 221)
- **"Delete a Bucket"** (page 244)
- **"Create or Delete a "Folder""** (page 245)
- **"Upload an Object"** (page 246)
- **"Set Object Properties"** (page 248)
- **"List or Search for Objects"** (page 257)
- **"Download an Object"** (page 258)
- **"Delete an Object"** (page 261)
- **"Restore an Auto-Tiered Object"** (page 258)

**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.

### 5.5.1.6. Delete a User

You can delete a HyperStore service user from the system.

**IMPORTANT !** If you delete a user, **the user's stored buckets and objects will be deleted from the system.** If you want to temporarily deny service access to a user without deleting their stored buckets and objects, don't delete the user. Instead, suspend the user by using the user Edit function.

To delete a user:

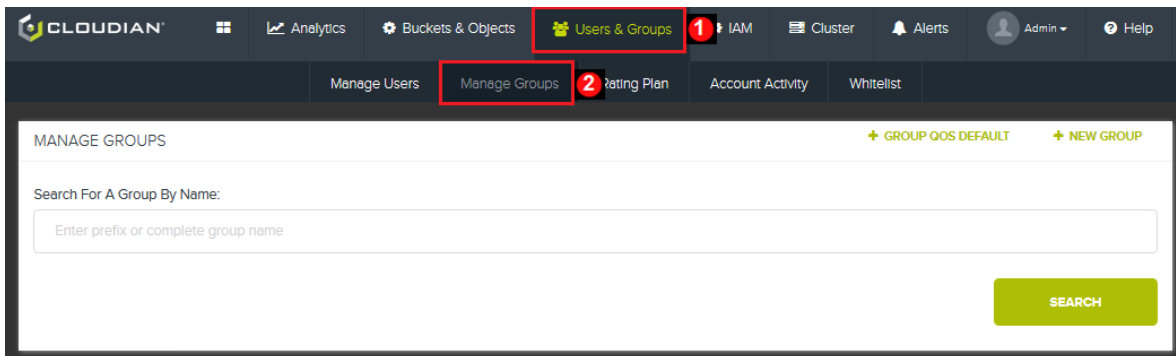
1. Use the CMC's [Manage Users](#) page to **"Retrieve a User or List of Users"** (page 265).
2. In the "Actions" column for the user that you want to delete, click **Delete**.

**Note** You cannot delete the default system administrator account. This is not allowed.

3. When prompted by the system, confirm that you want to delete the user.

## 5.5.2. Manage Groups

Path: **Users & Groups** → **Manage Groups**



Supported tasks:

- **"Add a Group"** (page 271)
- Work with existing groups:
  - **"Retrieve a Group or a List of Groups"** (page 275)
  - **"Edit a Group"** (page 276)
  - **"Set Quality of Service (QoS) Controls"** (page 285)
  - **"Delete a Group"** (page 277)

### 5.5.2.1. 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 281) (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 able 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 272) below.
4. Click **Save**.

#### 5.5.2.1.1. 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 131).

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: \*  Group Description:

Rating Plan:

☐ 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:

#### 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 SSL/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.

#### 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 275).

- **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 Puppet Master 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:

```
hstcl config set hsh.ldap.bindDN=<bind Distinguished Name>
hstcl config set hsh.ldap.bindPassword=<bind password>
hstcl config apply hsh
```

#### 5.5.2.2. 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.
2. 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	<a href="#">Group QOS</a> <a href="#">User QOS</a> <a href="#">Group Default</a> <a href="#">Edit</a> <a href="#">Delete</a>

### 5.5.2.3. Edit a Group

[CMC Interface]

You can change attributes of an existing group, such as the group's rating plan assignment, service status, or LDAP integration.

1. Use the CMC's [Manage Groups](#) page to **"Retrieve a Group or a List of Groups"** (page 275).

**Note** If you want to retrieve the **System Admin group** in order to edit that group's attributes, the group name is **0**.

2. 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 <a href="#">Edit</a> <a href="#">Delete</a>

☒ Active Group

Group Description:

Rating Plan:

Default-RP

☐ 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.

3. 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 272) (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 272).

#### *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 able 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.

- Click **Save**.

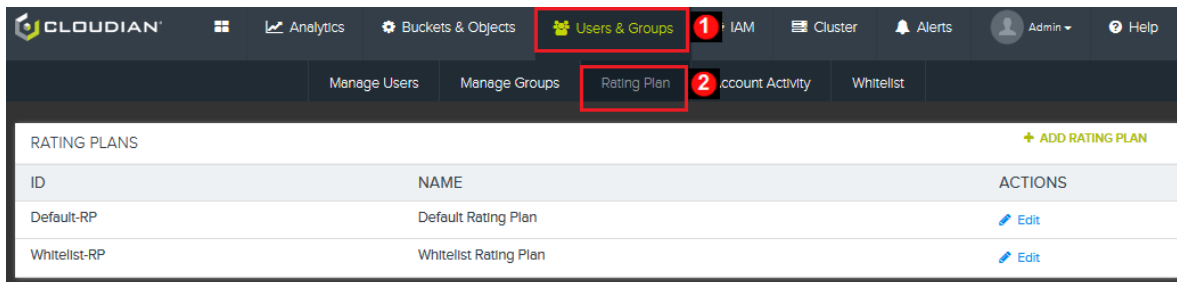
### 5.5.2.4. Delete a Group

**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.

- Use the CMC's [Manage Groups](#) page to **"Retrieve a Group or a List of Groups"** (page 275).
- In the "Actions" column for the group that you want to delete, click **Delete**.
- When prompted by the system, confirm that you want to delete the group.

### 5.5.3. Rating Plan

Path: **Users & Groups** → **Rating Plan**



Supported tasks:

- **"Add a Rating Plan"** (page 278)
- **"Edit a Rating Plan"** (page 280)
- **"Delete a Rating Plan"** (page 282)

**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 **"Usage Reporting and Billing Feature Overview"** (page 138).

#### 5.5.3.1. 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 [whitelist feature](#).

To create a new rating plan:

1. In the **Rating Plans** page click **Add Rating Plan**. This opens the **Add Rating Plan** panel.

ADD RATING PLAN

ID  Name  Currency

Rates

- ▶ Storage Size Rates
- ▶ Data-Transfer-IN rates
- ▶ Data-Transfer-OUT rates
- ▶ HTTP(S) GET/HEAD Rates
- ▶ HTTP(S) PUT/POST Rates
- ▶ HTTP(S) DELETE Rates

- Assign the plan a unique ID, and optionally a Name.
- 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).
- Specify the billing rates that will constitute the rating plan. You can specify billing rates per:
  - Average GBs of data in storage during the calendar month for which the bill is calculated ("Storage-Size Rates")
  - GBs of data uploaded to the system. ("Data-Transfer-IN Rates")
  - GBs 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"** (page 344) 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 146)

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 GB Per Month. In this example, the rate is \$2 per GB-month. That

same per GB-month rate applies whether a user stores, for instance, 10 GB-month (\$20 charge) or 100 GB-month (\$200 charge).

STORAGE SIZE (GB)	RATE PER GB PER MONTH
	<input type="text" value="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 GB of data uploaded during the billing period are priced at \$2 per GB; the next five GB are priced at \$2.50 per GB; and any uploads beyond that level (above the 10 GBs encompassed by the first two tiers) are priced at \$3 per GB.

	STORAGE SIZE (GB)	RATE PER GB PER MONTH	
Up to	<input type="text" value="5"/>	<input type="text" value="2"/>	<a href="#">Delete</a>
Next	<input type="text" value="5"/>	<input type="text" value="2.5"/>	<a href="#">Delete</a>
Above	<input type="text" value="10"/>	<input type="text" value="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.

### 5.5.3.2. 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 [whitelist feature](#).

### 5.5.3.3. Default Rating Plan

The Cloudian 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 GB: \$0.14 per GB-month
  - Second tier, from 1+ GB to 6 GB: \$0.12 per GB-month
  - Third tier, above 6 GB: \$0.10 per GB-month
- Data Transfer IN
  - First tier, up to 1 GB: \$0.20 per GB
  - Second tier, above 1 GB: \$0.10 per GB

- Data Transfer OUT
  - First tier, up to 1 GB: \$0.20 per GB
  - Second tier, above 1 GB: \$0.10 per GB
- 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

#### 5.5.3.4. Delete a Rating Plan

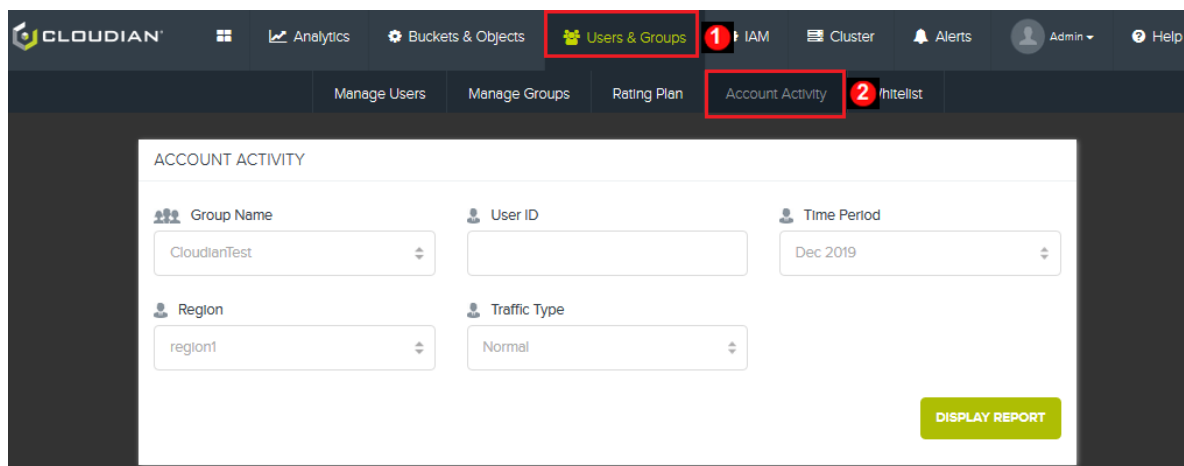
In the CMC's [Rating Plan](#) page click **Delete** to the right of the name of the plan that you want to delete. After you confirm that you want to take this action, the rating plan will be deleted from the system.

**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 default rating plan, or the "Whitelist-RP" plan (which supports the HyperStore [whitelist feature](#)).

#### 5.5.4. Account Activity

Path: **Users & Groups** → **Account Activity**



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 *mts.properties.erb*: "**reports.rolluphour.ttl**" (page 567). The default retention period is 65 days. Once this rollup data is deleted it can no longer be used to generate users' bills.

To generate a billable activity report for a user:

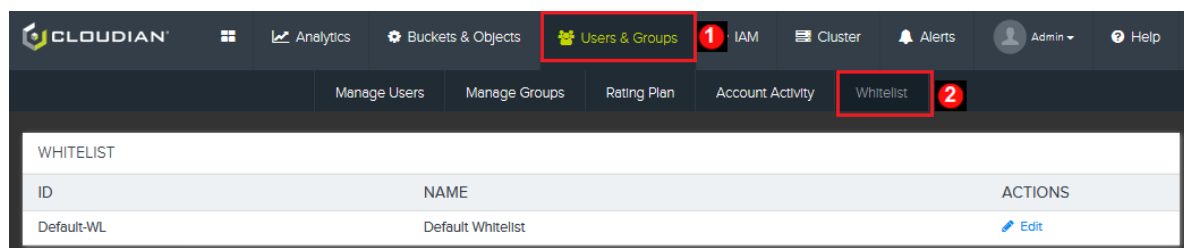
1. Choose a "Group Name" from the drop-down list and enter a "User ID" for the user.
2. Choose a "Time Period" from the drop-down list. The billing periods are calendar months. You can only create a bill for a past, completed month; you cannot create a bill for the current, in-progress month. The default 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 "Whitelist". "Whitelist" refers to request traffic that originates from white-listed source IP addresses (traffic subject to special pricing), and is only an option if white-listing is enabled in the system. "Normal" refers to all other traffic.
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 GB-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 GB-months) and the resulting charges.

ACCOUNT ACTIVITY				<a href="#">Print</a>
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 GB-Month : 0.14 (USD)	108 GB-Month	10.94	
	Next 5 GB-Month : 0.12 (USD)			
	Above : 0.10 (USD)			
	Total (USD)		10.94	

### 5.5.5. Whitelist

Path: **Users & Groups** → **Whitelist**



Supported tasks:

- Add or remove IP addresses from the whitelist

The Cloudian HyperStore billing whitelist feature enables service providers to specify a list of IP addresses or subnets that are allowed to have free S3 traffic with the HyperStore system. For S3 requests originating from IP addresses on the whitelist, a special rating plan is used that applies zero charge to all the traffic-related pricing metrics:

- Price per GB data transferred in
- Price per GB 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 whitelist 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 whitelisted 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 whitelisted IP addresses to be specially priced but not free).

**Note** In the current release you can only have one whitelist, with only one rating plan applied to it.

**Note** If you are using load balancers in front of the HyperStore S3 Service, the whitelist 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 "[s3\\_proxy\\_protocol\\_enabled](#)" (page 530) in [common.csv](#). 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 whitelist feature. You must use PROXY Protocol if you have load balancers in front of the S3 Service and want to use the whitelist feature .

#### *Note for multi-region systems*

In a multi-region HyperStore deployment, the whitelist is applied the same across all regions. There is no support for region-specific whitelists.

To add or remove IP addresses from the whitelist:

1. In the **Whitelist** page, to the right of the whitelist name, click **Edit**. This opens the **Edit Whitelist** panel.

2. If you want you can change the name of the whitelist ("Default Whitelist"), 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 whitelist (by adding or deleting IP addresses or subnets) will take effect starting with the next hourly roll-up of HyperStore usage data.

### 5.5.6. Set Quality of Service (QoS) Controls

Use the CMC's **Manage Users** page and **Manage Groups** page to set Quality of Service (QoS) limits for users.

**Note** By default the HyperStore system's enforcement of QoS restrictions is **disabled**. If you want to use the QoS feature, then before setting specific QoS limits for users and groups you must go to the CMC's **Configuration Settings** page and enable QoS enforcement.

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 <a href="#">Manage Users</a> page, click <b>User QoS Default</b> .	User QoS Limits: Defaults	User QoS Default. Defaults to "unlimited" for all QoS

QoS Task	Path to Configuration Panel	Configuration Panel Name	Panel's Default Values
			settings.
Set a default user QoS profile for members of a group	In the <a href="#">Manage Groups</a> page, retrieve the group. Then for the group click <b>User QoS Group Default</b> .	User QoS Limits: Group Defaults	Defaults to system default user QoS profile.
Set a user-specific QoS profile	In the <a href="#">Manage Users</a> page, retrieve the user. Then click <b>Set QoS</b> 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 <a href="#">Manage Groups</a> page, click <b>Group QoS Default</b> .	Group QoS Limits: Defaults	Defaults to "unlimited" for all QoS settings.
Set a group-specific group QoS profile	In the <a href="#">Manage Groups</a> page, retrieve the group. Then click <b>Group QoS</b> 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 (KB)		Unlimited <input checked="" type="checkbox"/>
Storage Quota Count		Unlimited <input checked="" type="checkbox"/>
Request Rate (Requests/minute)	Unlimited <input checked="" type="checkbox"/>	Unlimited <input checked="" type="checkbox"/>
Data Bytes IN (KB/minute)	Unlimited <input checked="" type="checkbox"/>	Unlimited <input checked="" type="checkbox"/>
Data Bytes OUT (KB/minute)	Unlimited <input checked="" type="checkbox"/>	Unlimited <input checked="" type="checkbox"/>

DELETE OVERRIDES
CANCEL
SAVE

- Make your desired edits to the QoS limits. Note that 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 (KB) — High Limit*

Storage quota limit, in number of KBs

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.
- *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.

#### *Storage Quota Count — High Limit*

Storage quota 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.
- *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.

#### *Request Rate — Warning Limit*

Request rate warning 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, an INFO level message is written to the S3 Service's application log. 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. (Note that the system does not inform the user that the warning threshold has been exceeded -- it only writes the aforementioned log message.)
- *For group QoS* — The implementation is the same as for user QoS, except that it applies to requests from all users in the group. 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, 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. 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. 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 (KB/minute) — Warning Limit*

Inbound data rate warning limit, in KBs 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 kilobytes of data.

*Data Bytes IN (KB/minute) — High Limit*

Inbound data rate high limit, in KBs 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 kilobytes of data. Note that if a block is triggered by the Data Bytes IN (KB) High Limit being reached, the block applies to all HTTP request types (not just PUTs.)

*Data Bytes OUT (KB/minute) — Warning Limit*

Outbound data rate warning limit, in KBs 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 kilobytes of data.

*Data Bytes OUT (KB/minute) — High Limit*

Outbound data rate high limit, in KBs 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 kilobytes of data. Note that if a block is triggered by the Data Bytes OUT (KB) High Limit being reached, the block applies to all HTTP request types (not just GETs.)

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 HyperStore System

In a multi-region HyperStore 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 HyperStore deployment with regions named "West" and "East", you might set your QoS limits in the "West" region so that each user is allowed to store 20GB of data. The 20GB 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 20GB of storage in the "East" region (for a total of 40GB per user across the entire multi-region system); or a different value might be configured for the "East" region such as 10GB or 30GB.

## 5.6. IAM

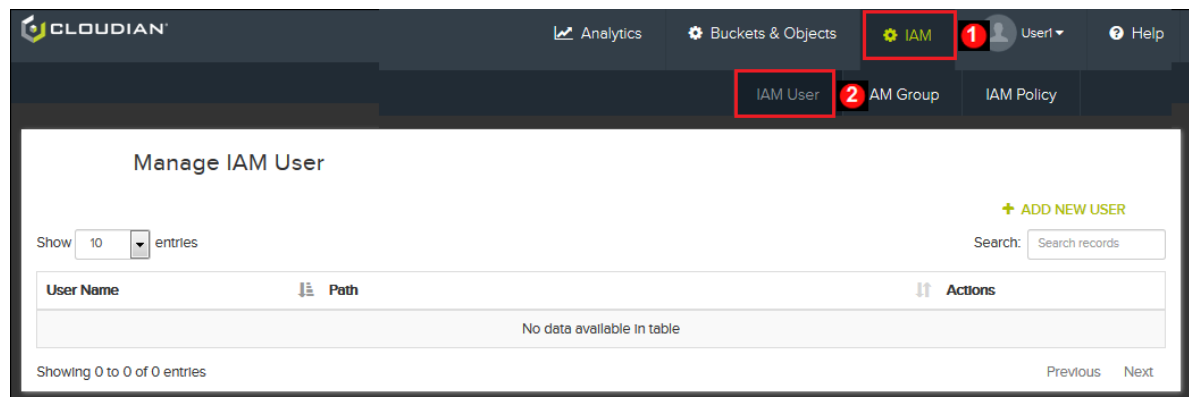
The **IAM** tab contains the following functions:

- [Manage IAM User](#)
- [Manage IAM Group](#)
- [Manage IAM Policy](#)

**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](#)" (page 991).

### 5.6.1. Manage IAM User

Path: **IAM** → **IAM User**



Supported tasks:

- **"Add an IAM User"** (page 290)
- **"Select an IAM User to Work With"** (page 291)
- **"Manage an IAM User's S3 Access Keys"** (page 291)
- **"Manage an IAM User's Group Membership"** (page 293)

- **"Manage an IAM User's Permissions"** (page 294)
- **"Delete an IAM User"** (page 296)

As a HyperStore account holder, you can create one or more IAM (Identity and Access Management) users under your 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 HyperStore S3 Service activity by IAM users under your account **will be counted toward your account usage**.

#### 5.6.1.1. 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**.

### 5.6.1.2. Select an IAM User to Work With

In the **Manage IAM User** page, by default all your IAM users will be listed in alphabetical order, with 10 users displayed at a time. You can navigate through the alphabetized list, 10 users at a time, by using the "Next" and "Previous" links in the lower right.

Alternatively, you can use the "Search" field to retrieve a single IAM user or a filtered list of users. To retrieve just one specific user, enter the user name. To retrieve a user list filtered by text string, enter the text string -- such as a user name prefix or a suffix. The text string filter will be applied to user names and also to user name "paths" (if you've been using paths with your IAM user names).

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'. 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' (selected), 'IAM POLICIES', and 'IAM GROUPS'. Below the tabs, there's a '+ CREATE NEW KEY' link. A search bar is present with the text 'Search: by access key id'. Below the search bar is a table with columns: 'Access Key ID', 'Status', 'Secret Access Key', 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.

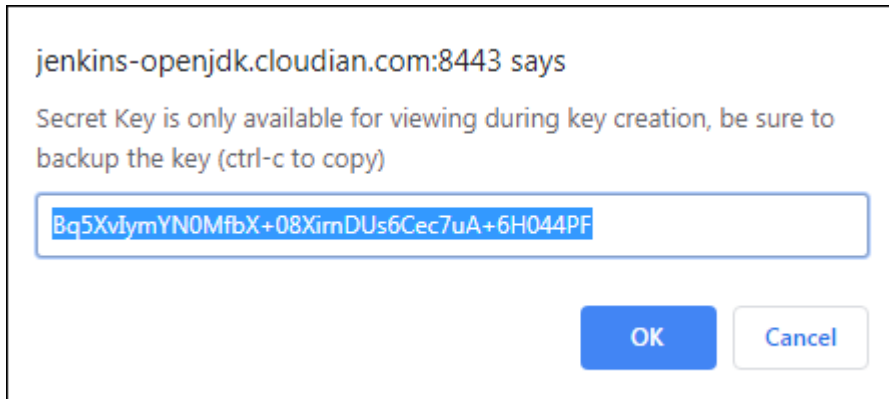
From this user detail page you can **"Manage an IAM User's S3 Access Keys"** (page 291) or **"Manage an IAM User's Group Membership"** (page 293) or **"Manage an IAM User's Permissions"** (page 294).

**Note** You can also edit the user's user name or path, by clicking **Edit User**.

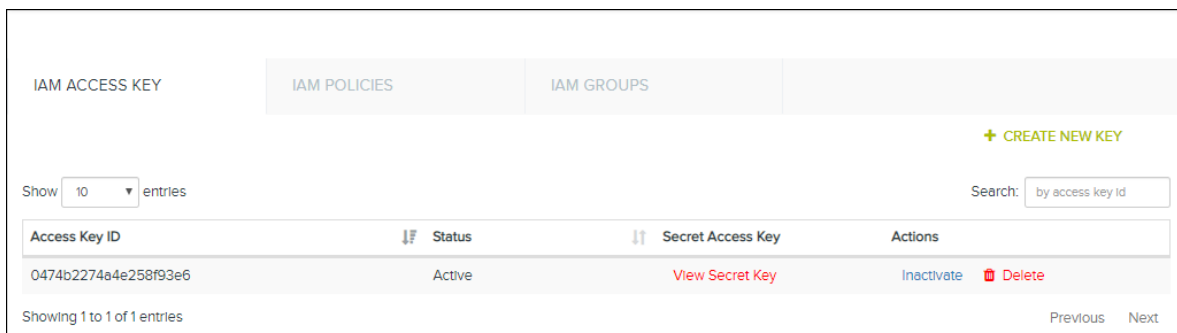
### 5.6.1.3. Manage an IAM User's S3 Access Keys

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). This limit is controlled by the "**credentials.iamuser.max**" (page 571) setting in *mts.properties.erb*.

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.

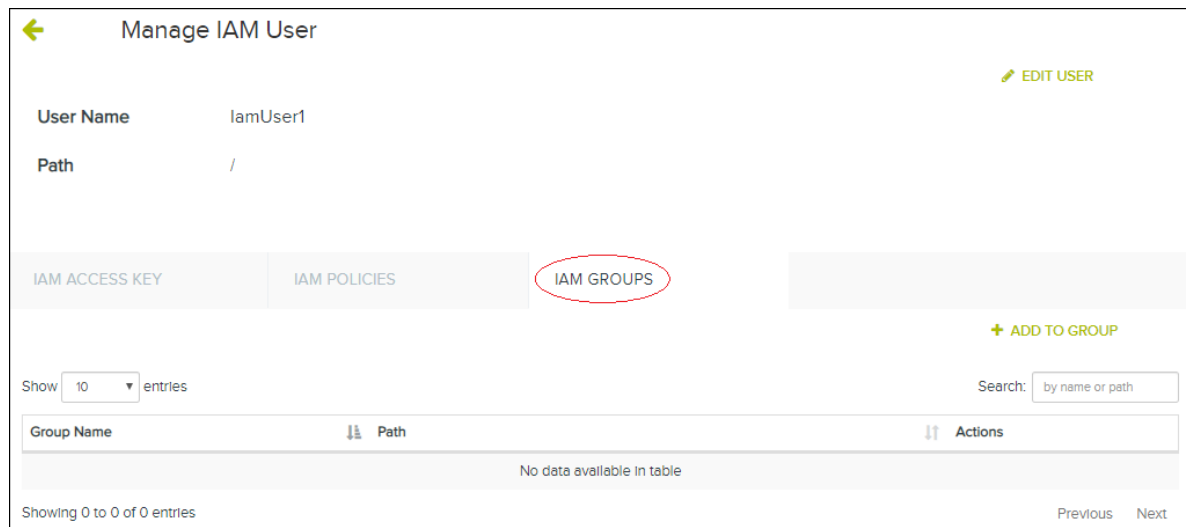
**Note** If an IAM user has no active access key he or she will not be able to access the Hyper-Store S3 Service. Deactivating all of an IAM user's keys can be a means for you to temporarily suspend the IAM user if you wish.

- 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**.

#### 5.6.1.4. Manage an IAM User's Group Membership

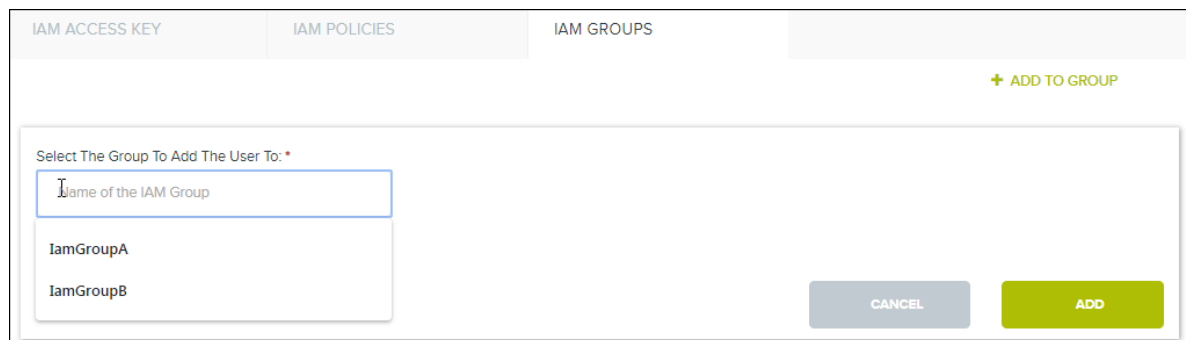
An IAM user can belong to one or more IAM groups that you have created. When an IAM user belongs to a group the user inherits the permissions associated with that group. (For more information on permissions see **"Manage an IAM User's Permissions"** (page 294)). You can manage a user's membership in 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 a green 'EDIT USER' link. Below the title, the 'User Name' is 'IamUser1' and the 'Path' is '/'. There are four tabs: 'IAM ACCESS KEY', 'IAM POLICIES', 'IAM GROUPS' (which is circled in red), and an unlabeled tab. Below the tabs, there's a '+ ADD TO GROUP' link. A 'Show' dropdown is set to '10' entries, and a 'Search' field is labeled 'by name or path'. Below this 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.

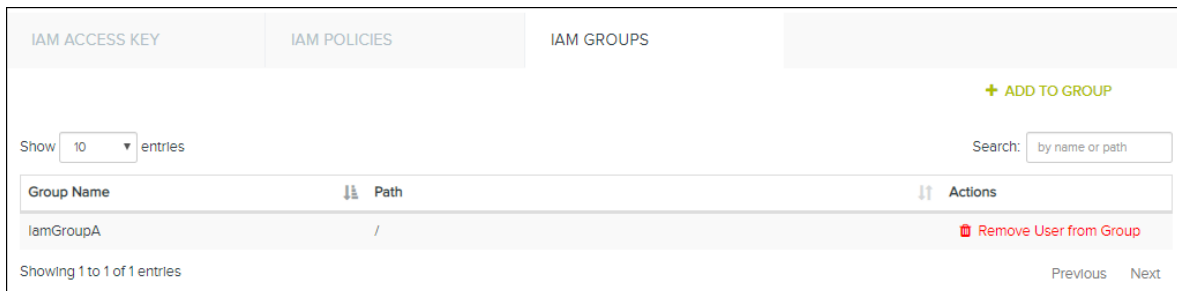
To add the user to an IAM group that you have created under your HyperStore account, click **Add to Group**. Then click in the "Name of the IAM Group" field, and select the group from the drop-down list that displays. The list shows all the IAM groups that the user does not already belong to.



The screenshot shows a modal dialog box titled 'Select The Group To Add The User To: \*'. It has a text input field labeled 'Name of the IAM Group'. Below the input field, there's a list of two options: 'IamGroupA' and 'IamGroupB'. At the bottom right of the dialog, there are two buttons: 'CANCEL' and 'ADD'.

If you have many IAM groups under your account and you want to filter the drop-down list, type a text string in the "Name of the IAM Group" 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 **Add**. The group will then appear in the list of groups that the user belongs to.



**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**.

### 5.6.1.5. 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 attached to a group that the user belongs to or attached directly to the user.

The most common way to manage IAM user permissions is to attach 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 attaching policies to IAM groups, see **"Manage IAM Group"** (page 297).

The system also supports attaching one or more policies directly to an individual IAM user, as described below. If multiple policies are associated 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.

The screenshot shows the 'Manage IAM User' console for user 'IamUser1'. The 'IAM POLICIES' tab is selected. A modal titled 'ADD IAM POLICY' is open, showing options for 'Managed Policy' (selected) and 'Inline Policy'. The 'Managed Policy Name' field is empty, and the 'Policy Document' field is also empty. The modal includes 'CANCEL' and 'ADD' buttons.

2. In the Add IAM Policy panel you can either:

- Attach an existing managed policy to the user, by selecting "Managed Policy", then clicking in the "Managed Policy Name" field and selecting the policy from the drop-down list that displays, and then clicking **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 304). 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.

The screenshot shows the 'Manage IAM User' console for user 'IamUser1'. The 'IAM POLICIES' tab is selected. Below the tabs, there is a search bar and a table listing the attached policies.

Policy Type	Policy Name	Policy Document	Actions
Managed Policy	S3_Read_Only	<a href="#">View Document</a>	<a href="#">Detach from User</a>

Showing 1 to 1 of 1 entries

### 5.6.1.5.1. 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 associated with 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.

**Note** The system does not support editing managed policies. Editing is only supported for inline policies.

### 5.6.1.5.2. 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 associated with 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.

### 5.6.1.6. Delete an IAM User

This procedure is for deleting an IAM user. Note first that:

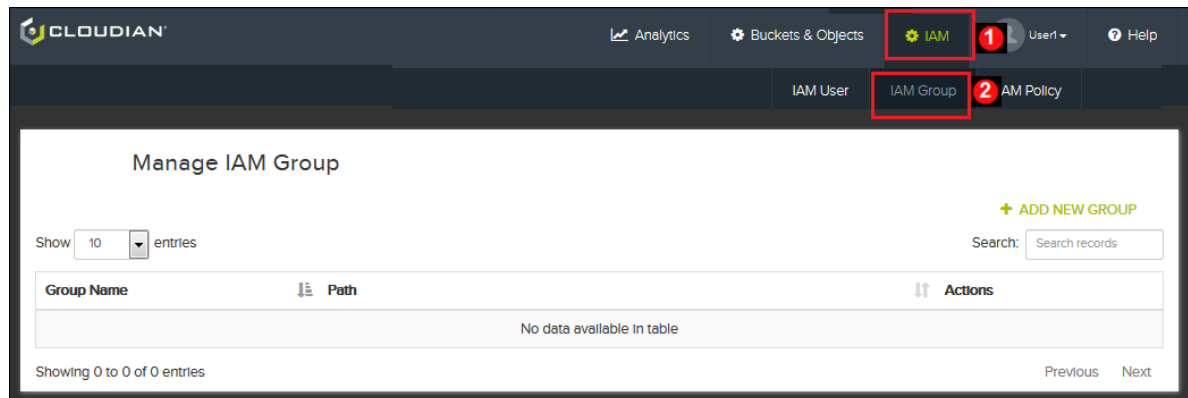
- Before deleting an IAM user you must [delete the user's S3 credentials](#) and [remove the user from any IAM groups](#) that she is in.
- Deleting an IAM user will not delete the user's stored S3 buckets and objects from the system. The S3 data is considered to belong to the parent HyperStore user 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.

## 5.6.2. Manage IAM Group

Path: **IAM** → **IAM Group**



Supported tasks:

- **"Add an IAM Group"** (page 297)
- **"Select an IAM Group to Work With"** (page 298)
- **"Manage an IAM Group's User Membership"** (page 299)
- **"Manage an IAM Group's Permissions"** (page 300)
- **"Delete an IAM Group"** (page 303)

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 a group's membership will inherit the S3 and IAM permissions that you have associated with the group.

### 5.6.2.1. 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 form titled "Manage IAM Group". In the top right corner, there is a green link that says "+ ADD NEW GROUP". Below the title, there are two input fields. The first is labeled "Group Name" with a red asterisk next to it, and the second is labeled "Path". At the bottom right of the form, 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 identity 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**.

#### 5.6.2.2. Select an IAM Group to Work With

In the **Manage IAM Group** page, by default all your IAM groups will be listed in alphabetical order, with 10 groups displayed at a time. You can navigate through the alphabetized list, 10 groups at a time, by using the "Next" and "Previous" links in the lower right.

Alternatively, you can use the "Search" field to retrieve a single IAM group or a filtered list of groups. To retrieve just one specific group, enter the group name. To retrieve a group list filtered by text string, enter the text string - such as a group name prefix or a suffix. The text string filter will be applied to group names and also to group name "paths" (if you've been using paths with your IAM group names).

Once the desired group is displayed in the list, **click the group name**. This opens the **Manage IAM Group detail page** for the group.

Manage IAM Group

Group Name IamGroupA

Path /

IAM POLICIES IAM USERS

+ ADD IAM POLICY

Show 10 entries Search: by name

Policy Type	Policy Name	Policy Document	Actions
No data available in table			

Showing 0 to 0 of 0 entries Previous Next

From this page you can **"Manage an IAM Group's User Membership"** (page 299) or **"Manage an IAM Group's Permissions"** (page 300).

**Note** You can also edit the group's group name or path, by clicking **Edit Group**.

### 5.6.2.3. 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 300)). 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.

Manage IAM Group

Group Name IamGroupA

Path /

IAM POLICIES IAM USERS

+ ADD IAM USER

Show 10 entries Search: by name or path

User Name	Path	Actions
No data available in table		

Showing 0 to 0 of 0 entries Previous Next

2. To add a user to the IAM group, click **Add IAM User**. Then click in the "Name of the IAM User" field, and select the user from the drop-down list that displays. The list shows all the IAM users that have been

created under your HyperStore account that do not currently belong to this IAM group.

If you have many IAM users under your account and you want to filter the drop-down list, type a text string in the "Name of the IAM User" field. This will shorten the drop-down list by limiting it to users whose user names match against the text string.

- 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**.

#### 5.6.2.4. 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 attach one or more IAM "policies", with each policy defining a particular set of permissions. If multiple policies are attached to a group, then the members of the group get the combined permissions associated with the multiple policies. If there are conflicts within the attached 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 attached 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.

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, the 'Group Name' is 'IamGroupB' and the 'Path' is '/'. There are two tabs: 'IAM POLICIES' (selected) and 'IAM USERS'. Below the tabs, there's a '+ ADD IAM POLICY' button. The main form area has two radio buttons: 'Managed Policy' (selected) and 'Inline Policy'. Below these, there's a 'Managed Policy Name' field with a placeholder 'Name of the IAM Policy'. Underneath is a 'Policy Document' field with a placeholder 'Detail of the IAM Policy Document'. At the bottom right, there are 'CANCEL' and 'ADD' buttons.

2. In the Add IAM Policy panel you can either:
  - Attach an existing managed policy to the group, by selecting "Managed Policy", then clicking in the "Managed Policy Name" field and selecting the policy from the drop-down list that displays, and then clicking **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 304). 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.

Manage IAM Group

EDIT GROUP

Group Name: IamGroupB

Path: /

IAM POLICIES | IAM USERS

ADD IAM POLICY

Show 10 entries Search: by name

Policy Type	Policy Name	Policy Document	Actions
Managed Policy	S3_Read_Only	<a href="#">View Document</a>	<a href="#">Detach from Group</a>

Showing 1 to 1 of 1 entries Previous Next

#### 5.6.2.4.1. 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 associated with 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:::bucket"
    }
  ]
}
```

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.

**Note** The system does not support editing managed policies. Editing is only supported for inline policies.

#### 5.6.2.4.2. 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 associated with 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.

#### 5.6.2.5. Delete an IAM Group

Before deleting an IAM group you must:

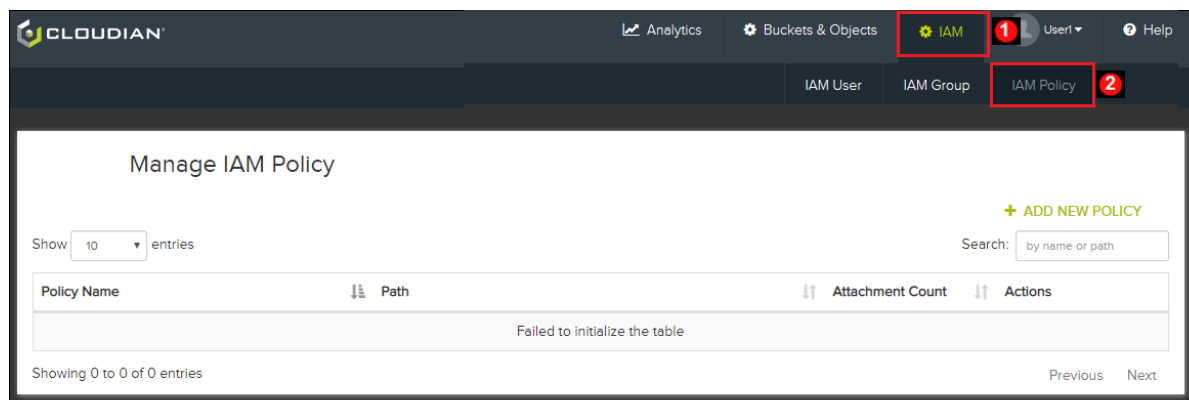
- [Remove any IAM users](#) who are currently in the group. You cannot delete an IAM group that currently has IAM users in it. You do not need to delete the IAM users from the system -- you only need to remove them from the group.
- [Delete any inline policies](#) that are configured for the group.

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.

### 5.6.3. Manage IAM Policy

Path: **IAM** → **IAM Policy**



Supported tasks:

- **"Add an IAM Managed Policy"** (page 304)
- **"Select an IAM Managed Policy to Work With"** (page 306)
- **"Attach or Detach an IAM Managed Policy to Groups and Users"** (page 307)
- **"Delete an IAM Managed Policy"** (page 309)

The CMC's **Manage IAM Policy** page is for creating and working with IAM **managed policies** -- not inline policies. For background:

- An IAM managed policy is an independent, reusable policy that you can attach to multiple IAM groups and/or to multiple individual IAM users. In most circumstances managed policies are the best way to grant permissions to your IAM groups and users.
- An IAM inline policy is a policy that you create for one specific IAM group or one specific IAM user. The policy exists solely as a part of that group's profile or that user's profile, and cannot be applied to any other groups or users. 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 or user and not for any other groups or users. To create an inline policy you use the [Manage IAM User](#) page or the [Manage IAM Group](#) page -- not the **Manage IAM Policy** page.

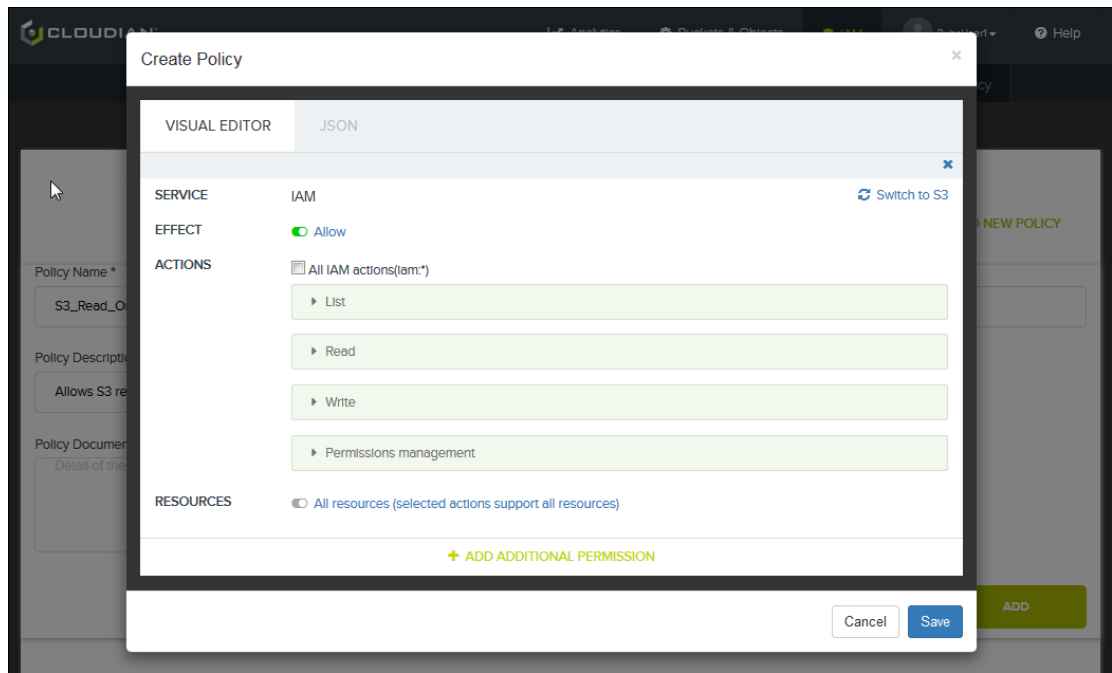
### 5.6.3.1. Add an IAM Managed Policy

**Note** When you create a new IAM managed policy, the policy by default is not attached to any of your IAM groups or users. After creating a policy you can attach it to your IAM groups or users.

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.

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.



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 **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- 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 resources scope. (Use the Resources toggle that lets you apply the statement to all applicable resources [the default] or only to a specified resource or resources).

**Note** The current HyperStore release does not support editing an IAM managed policy after you finish creating it, so as you create this policy be sure that you configure the permission definitions exactly as you want them.

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.



### 5.6.3.2. Select an IAM Managed Policy to Work With

In the **Manage IAM Policy** page, by default all your IAM managed policies will be listed in alphabetical order, with 10 policies displayed at a time. You can navigate through the alphabetized list, 10 policies at a time, by using the "Next" and "Previous" links in the lower right.

Alternatively, you can use the "Search" field to retrieve a single IAM managed policy or a filtered list of managed policies. To retrieve just one specific policy, enter the policy name. To retrieve a policy list filtered by text string, enter the text string -- such as a policy name prefix or a suffix. The text string filter will be applied to policy names and also to policy name "paths" (if you've been using paths with your IAM managed policy names).

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.

The screenshot shows the 'Manage IAM Policy' page in the AWS IAM console. At the top, there is a back arrow and the title 'Manage IAM Policy'. Below this, the policy details are listed: 'Policy Name' is 'S3\_Read\_Only', 'Policy Arn' is 'arn:aws:iam::6c8947d1e96b1ed9211bd624bef01277:policy/S3\_Read\_Only', 'Policy Description' is 'Allow S3 List actions and Read actions but not Write or Permissions actions', and 'Attachment Count' is '0'. Below the details, there are four tabs: 'IAM POLICY DETAIL' (selected), 'IAM GROUPS', 'IAM USERS', and an empty tab. Below the tabs, there is a 'Show' dropdown set to '10' and a 'entries' label. Below this, there is a 'Policy Document' section with a 'View Document' link. At the bottom, it says 'Showing 1 to 1 of 1 entries' and has 'Previous' and 'Next' links.

From this page you can **"Attach or Detach an IAM Managed Policy to Groups and Users"** (page 307).

**Note** You can also view the policy details in JSON format by clicking **View Document**. However, the current version of HyperStore **does not support editing the contents of an existing IAM policy**.

### 5.6.3.3. Attach or Detach an IAM Managed Policy to Groups and Users

You can attach an IAM managed policy to 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. Alternatively you can attach an IAM managed policy to a specific IAM user, in which case just that user will inherit the permissions defined by that IAM managed policy.

You can also attach a managed policy to multiple groups and/or multiple specific users. To do this, you will need to execute the attachment process one group or user at a time (for example, attach the policy to a group and then attach it to another group and so on).

You can attach an IAM managed policy to an IAM group by using either the policy's [Manage IAM Policy detail page](#) or the group's [Manage IAM Group detail page](#). Likewise you can attach an IAM managed policy to an IAM user by using either the policy's Manage IAM Policy detail page or the user's [Manage IAM User detail page](#).

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::6c8947d1e96b1ed9211bd624bef01277:policy/S3\_Read\_Only

**Policy Description** Allow S3 List actions and Read actions but not Write or Permissions actions

**Attachment Count** 0

**IAM POLICY DETAIL** **IAM GROUPS** **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 account that do not currently have this IAM managed policy attached.

**IAM POLICY DETAIL** **IAM GROUPS** **IAM USERS**

[+ ATTACH TO GROUP](#)

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.

**IAM POLICY DETAIL** **IAM GROUPS** **IAM USERS**

[+ ATTACH TO GROUP](#)

Show 10 entries Search: by name

Group Name	Actions
IamGroupA	<a href="#">Detach from Group</a>

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:** Follow the instructions above, except in the policy's Manage IAM Policy detail page, select the **IAM Users** tab rather than the **IAM Groups** tab.

#### 5.6.3.3.1. Checking an IAM Managed Policy's Current Attachments

At any time you can see which groups and specific users 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.

#### 5.6.3.4. Delete an IAM Managed Policy

**Before deleting an IAM managed policy** you must [detach the policy](#) from any of your IAM groups or users to which it is currently attached.

To delete an IAM managed policy:

1. In the **Manage IAM Policy** page, if the policy that you want to delete is not among those listed when you first access the page, retrieve the policy by typing the policy name in the "Search" field.
2. In the "Actions" column for the policy that you want to delete, click **Delete**.
3. When prompted by the system, confirm that you want to delete the policy.

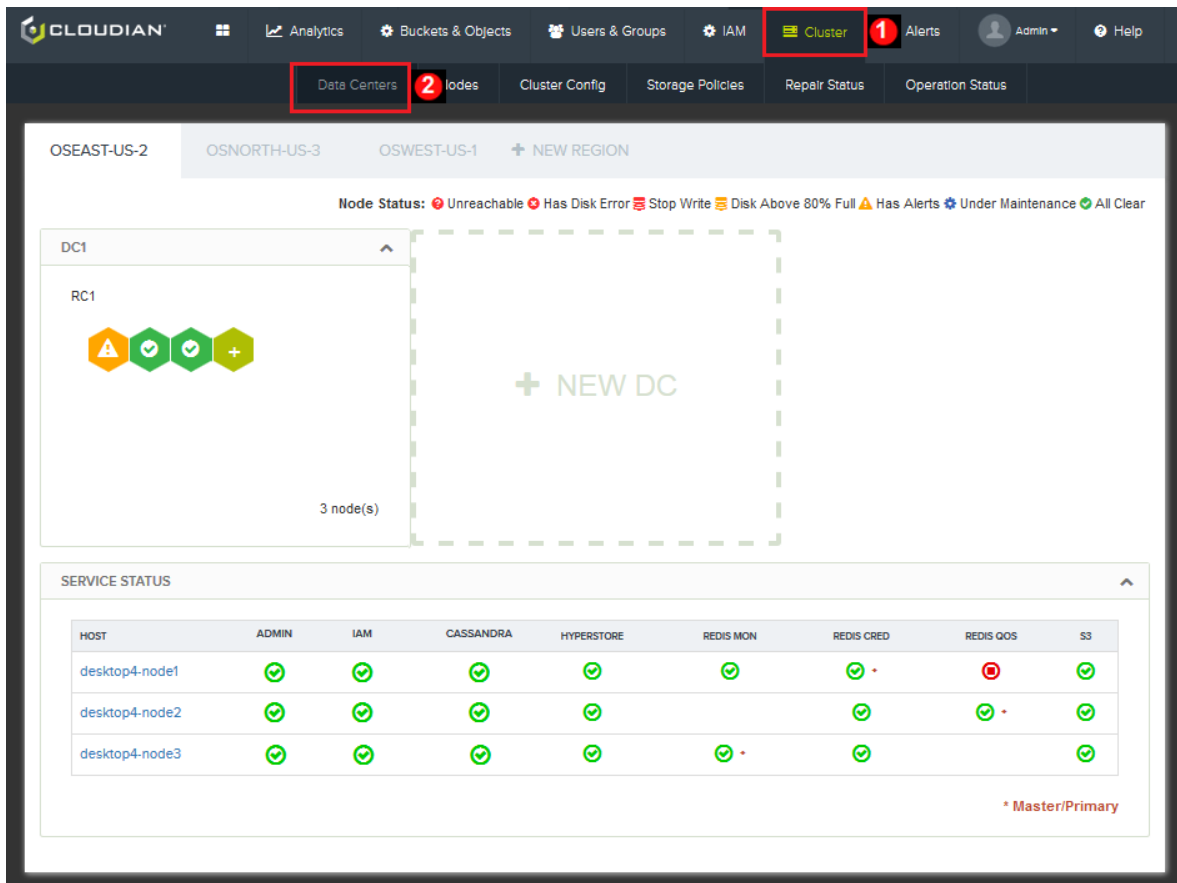
## 5.7. Cluster

The **Cluster** tab contains the following functions:

- [Data Centers](#)
- [Node Status](#)
- [Node Activity](#)
- [Node Advanced](#)
- [Cluster Information](#)
- [Configuration Settings](#)
- [Storage Policies](#)
- [Repair Status](#)
- [Operation Status](#)

### 5.7.1. Data Centers

Path: **Cluster** → **Data Centers**



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 420).
- 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 430).
- Adding a Region — For the full procedure including important steps to take before you add a region, see **"Adding a Region"** (page 437).

#### 5.7.1.1. 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 158)

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 198).



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.



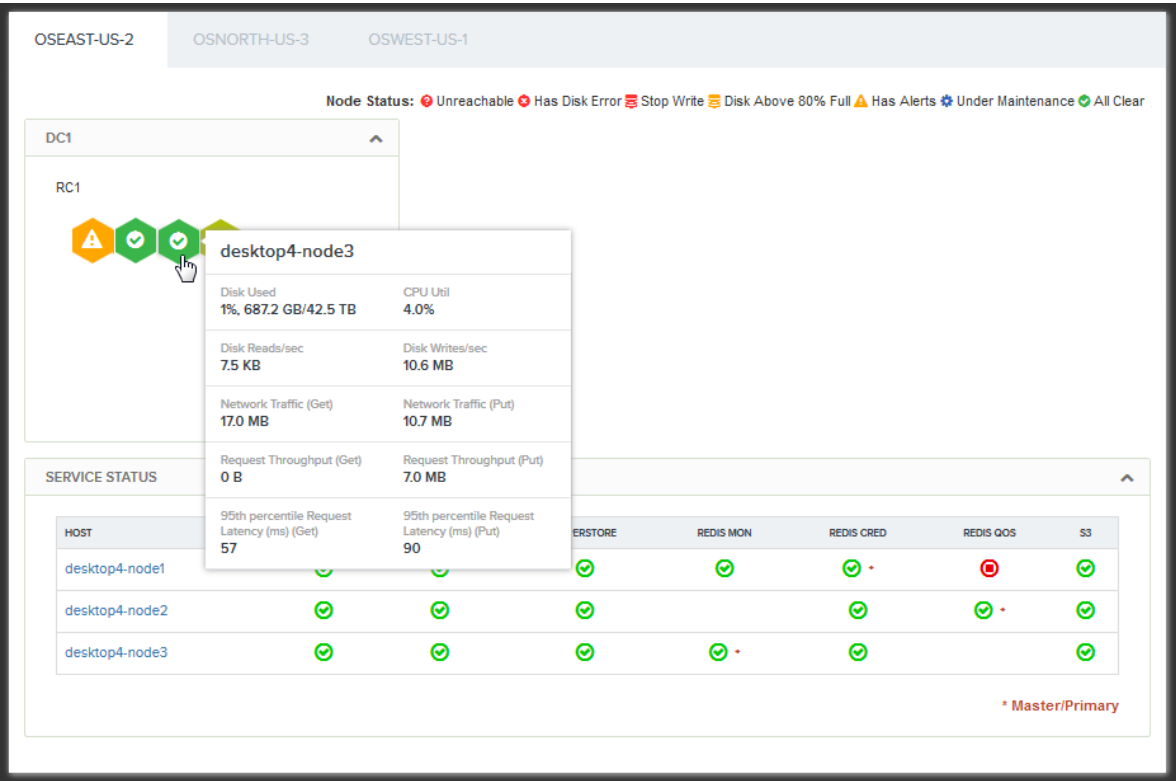
Blue with gear indicates that the **node is under maintenance**. This means either that you or another administrator put the node into maintenance mode (see **"Start Maintenance Mode"** (page 329)) in which case the node is not currently supporting S3 writes or reads.



Green with check mark indicates that the **node has no unacknowledged alerts** nor any of the conditions listed above.

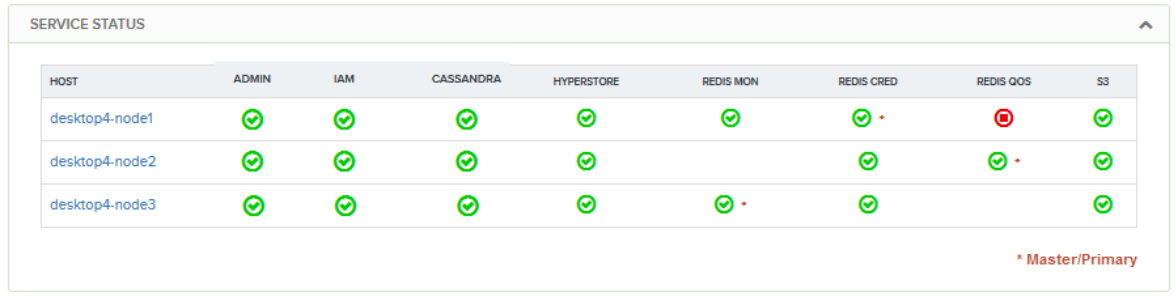
**Note** If multiple of the above conditions apply to one node, that node's icon will reflect just the highest priority condition, with the conditions prioritized in this order: "Under Maintenance" > "Unreachable" > "Has Disk Error" > "Disks Above 80% Full" > "Has Alerts".

To view a node's hostname and summary node statistics, hold your cursor over a cube.






The 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 314) 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.



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.

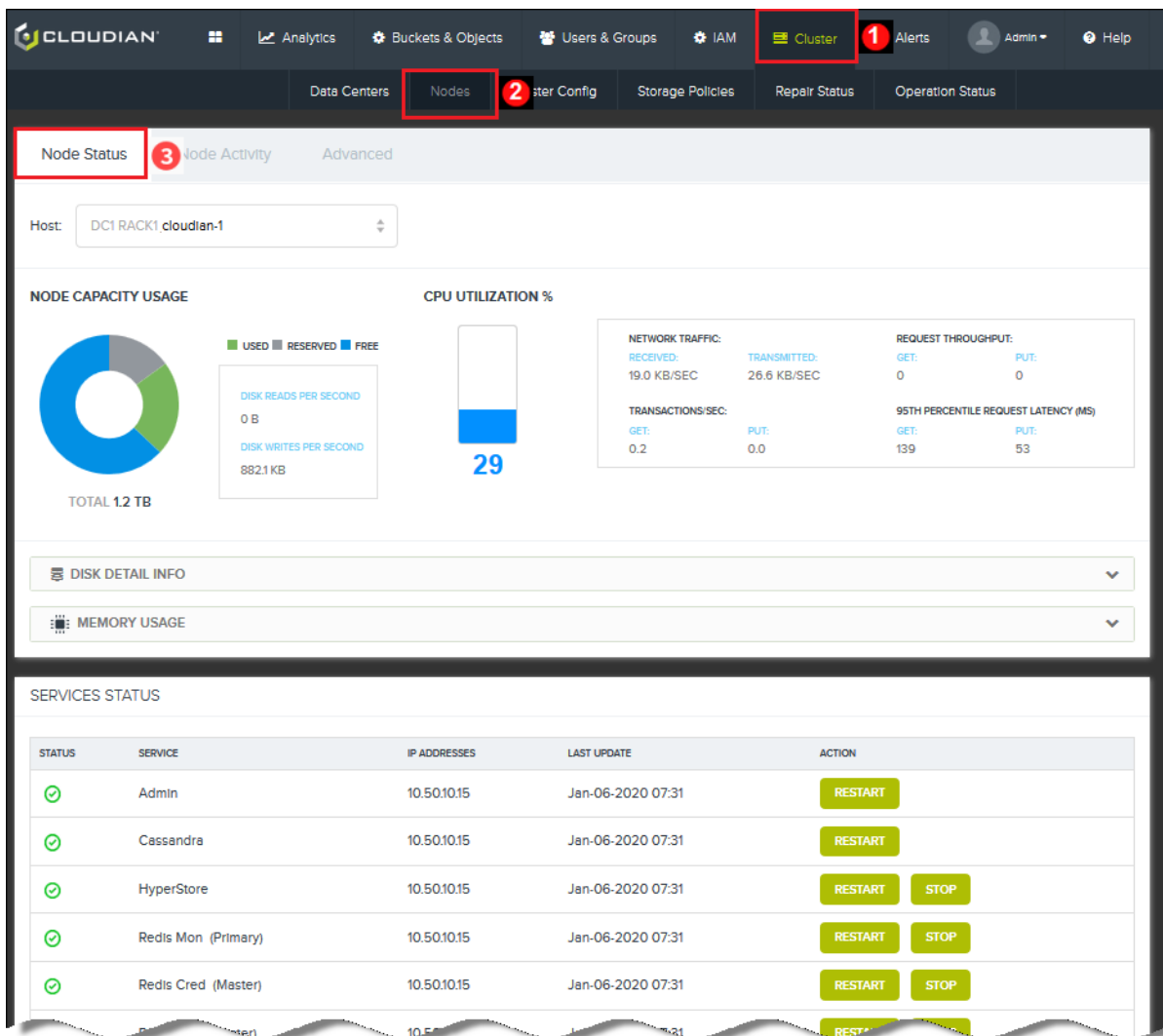
Status is shown for the following services:

- [Admin Service](#)
- [IAM Service](#) (if enabled)
- [Cassandra](#)
- [HyperStore Service](#)
- [Redis Monitor](#)
- [Redis Credentials](#)
- [Redis QoS](#)
- [S3 Service](#)

For Redis Monitor, Redis Credentials, and Redis QoS, the service status icon includes a red asterisk ( \* ) if it is the master or primary instance of the service. For Redis Credentials or QoS, 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).

## 5.7.2. Node Status

Path: **Cluster** → **Nodes** → **Node Status**



Supported tasks:

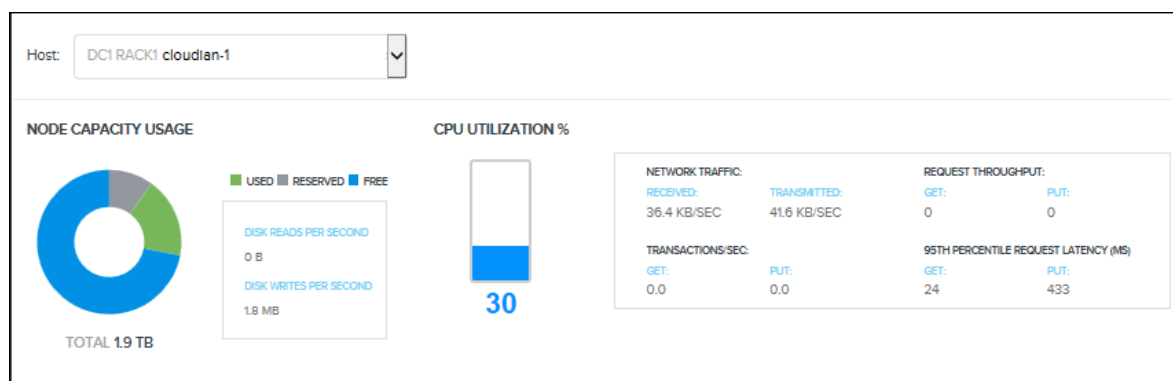
- **"View a Node's Summary Status"** (page 314)
- **"View a Node's Disk Detail"** (page 317)
- **"Locate a Disk on a HyperStore Appliance"** (page 319)
- **"View a Node's Memory Usage"** (page 320)
- **"View a Node's Services Status"** (page 321)
- **"Start, Stop, or Restart Services On a Node"** (page 321)
- **"View and Acknowledge Node Alerts"** (page 322)

**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.

### 5.7.2.1. 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, Reserved, 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 capacity 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 **Reserved** segment indicates the portion of the node's total disk capacity that is reserved and cannot be used for data storage. The Reserved portion consists of the Linux "reserved blocks percentage" plus the HyperStore stop-write buffer.
  - By default in CentOS/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 157).

The Reserved segment always displays in **gray**.

- The **Free** segment indicates the portion of the node's total disk capacity that is neither used nor reserved, and is therefore available for storing new data. The Free segment always displays in **blue**.

**IMPORTANT !** See **"Capacity Monitoring and Expansion"** (page 71) 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 KB or MB or GB 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 KB or MB or GB 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 (received and transmitted) 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 KB or MB or GB as appropriate.

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 MB 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)*

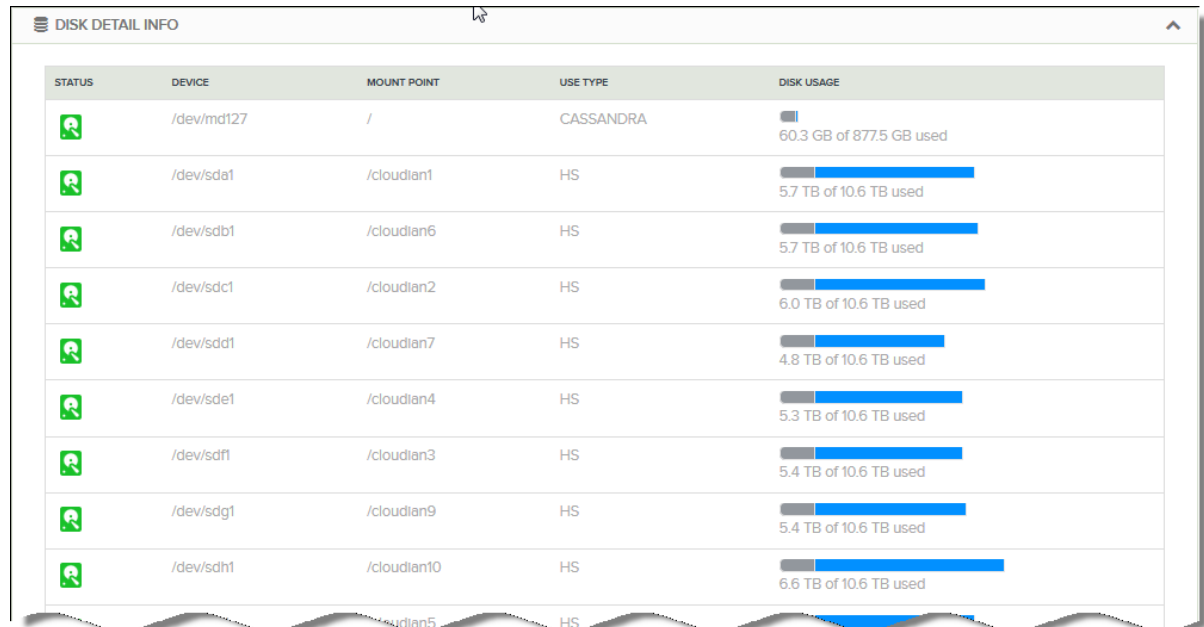
For just this node, the 95th percentile latencies for S3 PUT and S3 GET transactions in milliseconds.

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.

### 5.7.2.2. 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.



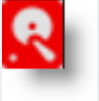
STATUS	DEVICE	MOUNT POINT	USE TYPE	DISK USAGE
	/dev/md127	/	CASSANDRA	60.3 GB of 877.5 GB used
	/dev/sda1	/cloudian1	HS	5.7 TB of 10.6 TB used
	/dev/sdb1	/cloudian6	HS	5.7 TB of 10.6 TB used
	/dev/sdc1	/cloudian2	HS	6.0 TB of 10.6 TB used
	/dev/sdd1	/cloudian7	HS	4.8 TB of 10.6 TB used
	/dev/sde1	/cloudian4	HS	5.3 TB of 10.6 TB used
	/dev/sdf1	/cloudian3	HS	5.4 TB of 10.6 TB used
	/dev/sdg1	/cloudian9	HS	5.4 TB of 10.6 TB used
	/dev/sdh1	/cloudian10	HS	6.6 TB of 10.6 TB used
	/dev/sdi1	/cloudian5	HS	5.7 TB of 10.6 TB used

The panel displays the information described below.

#### Status

This field displays an icon that indicates the disk's summary status.

Icon	Meaning
	<p><b>OK</b> — The green disk icon indicates that there are no errors for the disk.</p>
	<p><b>Error</b> — The orange disk icon indicates that the disk is in an error state.</p> <p>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 configurable error rate threshold that would trigger the automatic disabling of the disk (<i>hyperstore-server.properties.erb: "disk.-fail.error.count.threshold"</i> (page 551)).</p> <p>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.</p> <p>If a disk is in an Error state, a <b>Clear Error History</b> button appears toward the bottom of the <b>Detail Disk Info</b> panel. You can click this button to clear the disk's Error status and return the</p>

Icon	Meaning
	<p>disk status to "OK". If the error occurs again, the disk status display will revert back to showing an Error status for the disk.</p> <p><b>Note</b> 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.</p>
	<p>The red disk icon means one of two things depending on whether the disk is a HyperStore data disk or an OS disk:</p> <ul style="list-style-type: none"> <li> <b>Disabled</b> (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: <ul style="list-style-type: none"> <li>You or another administrator disabled the disk using the HyperStore <i>disableDisk</i> function. For background information see <b>"Disabling a HyperStore Data Disk"</b> (page 479).</li> <li>The HyperStore system detected indications of disk failure and automatically disabled the disk, in accordance with the <a href="#">HyperStore Disk Failure Action</a> configuration setting.</li> </ul> <p><b>Note</b> For recovering from a disabled disk, see <b>"Enabling a HyperStore Data Disk"</b> (page 480) or -- if the disk is bad -- <b>"Replacing a HyperStore Data Disk"</b> (page 482).</p> </li> <li> <b>Disk Failure Under RAID-1</b> (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 <b>only for HyperStore Appliance machines</b>, not for software-only deployments of HyperStore. <p><b>Note</b> If this occurs, the disk will also be marked by a red exclamation mark in the <a href="#">appliance disk map</a>.</p> </li> </ul> <p>As part of the <a href="#">Smart Support</a> feature, if a data disk fails (becomes disabled), <b>information about the failed disk is automatically sent to Cloudian Support</b> 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.</p>

*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). Disks with mount points specified by the configuration file setting `common.csv: hyperstore_data_directory` are HS disks. In a typical configuration there will be multiple HS disks on each node.
- "Cassandra" — The disk is storing Cassandra data (the directory specified by the configuration file setting `common.csv: cassandra_data_directory`) and/or the Cassandra commit log (the directory specified by the configuration file setting `common.csv: cassandra_commit_log_directory`).
- "Redis" — The disk is storing Redis data (the directory specified by the configuration file setting `common.csv: redis_lib_directory`).
- "Log" — The disk is storing application logs (the directories specified by the `common.csv` settings `cloud-ian_log_directory`, `cassandra_log_directory`, and `redis_log_directory`).
- "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 used (in **blue**) and what portion is reserved (in **gray**). For more information about "reserved" capacity see **"Node Capacity Usage"** (page 315)

The blue portion of the graphic turns to:

- **Orange** if the reserved 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 reserved 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 GB of Y TB used", the reserved capacity is counted as "used".

#### 5.7.2.3. Locate a Disk on a HyperStore Appliance

The **Appliance** section of the CMC's [Node Status](#) page is available only if the node is a HyperStore appliance. In the event of a disk problem (as indicated in the [Disk Detail](#) section) you can open the **Appliance** section 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 turn the ID light of the drive on the appliance to **amber** color (rather than the default of **blue** which indicates a healthy drive). First look through the drive table to find the device name that you're looking for, then click the button for that drive.

**Note** The following are exceptions to the drive light turning amber:

- On a HyperStore 40xx appliance, the drive light for an OS drive will blink blue for three minutes, rather than turning amber.
- On a Lenovo Storage DX8200C appliance, the drive light for any drive (OS drive or data disk drive) will blink blue for three minutes, rather than turning amber.

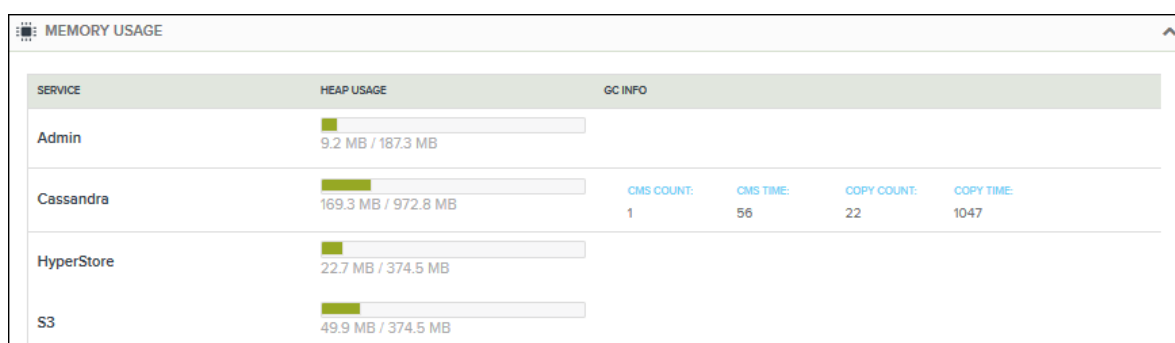
**Note** If a red exclamation mark appears beside a device name in the drive table, this means that the device is offline (not running). A script monitors the /sys directory on the appliance to detect devices that have gone offline.

### 5.7.2.3.1. 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 and the appliance's chassis light will blink blue for three minutes.

### 5.7.2.4. 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.

### 5.7.2.5. 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	192.168.1.10	Mar-09-2018 13:16	<a href="#">RESTART</a>
	Cassandra	192.168.1.10	Mar-09-2018 13:16	<a href="#">RESTART</a>
	HyperStore	192.168.1.10	Mar-09-2018 13:16	<a href="#">START</a>
	Redis Mon (Primary)	192.168.1.10	Mar-09-2018 13:16	<a href="#">RESTART</a> <a href="#">STOP</a>
	Redis Cred (Master)	192.168.1.10	Mar-09-2018 13:16	<a href="#">RESTART</a> <a href="#">STOP</a>
	Redis Qos (Master)	192.168.1.10	Mar-09-2018 13:16	<a href="#">RESTART</a> <a href="#">STOP</a>
	S3	192.168.1.10	Mar-09-2018 13:16	<a href="#">RESTART</a>

[RESTART ALL](#)

For each service, one of the following service status icons is displayed:



The service is **up and running**.



The service is **down**.

Status is shown for the following services:

- [Admin Service](#)
- [Cassandra](#)
- [HyperStore Service](#)
- [Redis Credentials](#) (if installed on the node)
- [Redis QoS](#) (if installed on the node)
- [Redis Monitor](#) (if installed on the node)
- [S3 Service](#)

**Note** The node's status is automatically checked and updated each minute, as indicated by the "Last Update" column.

### 5.7.2.6. 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.

SERVICES STATUS				
STATUS	SERVICE	IP ADDRESSES	LAST UPDATE	ACTION
✓	Admin	10.10.10.10	Mar-09-2018 13:16	RESTART
✓	Cassandra	10.10.10.10	Mar-09-2018 13:16	RESTART
⊙	HyperStore	10.10.10.10	Mar-09-2018 13:16	START
✓	Redis Mon (Primary)	10.10.10.10	Mar-09-2018 13:16	RESTART STOP
✓	Redis Cred (Master)	10.10.10.10	Mar-09-2018 13:16	RESTART STOP
✓	Redis Qos (Master)	10.10.10.10	Mar-09-2018 13:16	RESTART STOP
✓	S3	10.10.10.10	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 (the screen shot above is from a single-node system). 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 — depending on your configured consistency requirements for service metadata and the status of your other Cassandra nodes.

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](#). However, the CMC may not function properly while these services are down.

#### 5.7.2.7. 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					<a href="#">+ SHOW ACKNOWLEDGED</a>
<input type="checkbox"/>	SEVERITY	ALERT TYPE	ALERT TEXT	LAST UPDATE	COUNT
<input type="checkbox"/>	High	HyperStore	[Service Down or Unreachable]	Mar-09-2018 13:21	17
					<a href="#">ACKNOWLEDGE</a>

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 385).

**Note** Acknowledged alerts are **automatically deleted from the system** after a time period configured by the *mts.properties.erb*: **"events.acknowledged.ttl"** (page 568) 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 Clouidian Support each day.

### 5.7.3. 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 (as configured by *common.csv*: "**hyperstore\_data\_directory**" (page 517)) or the Cassandra data directory (as configured by *common.csv*: "**cassandra\_data\_directory**" (page 535)) 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.

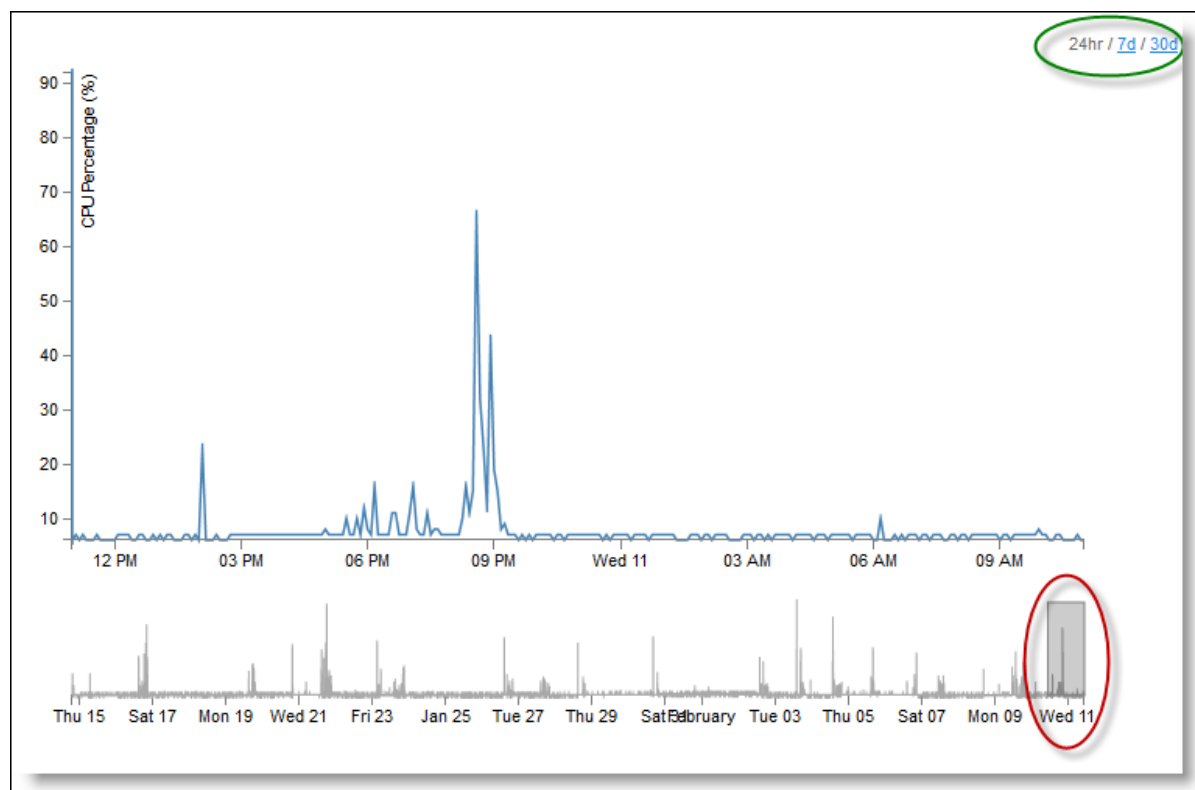
These stats are measured each five minutes.

#### *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).



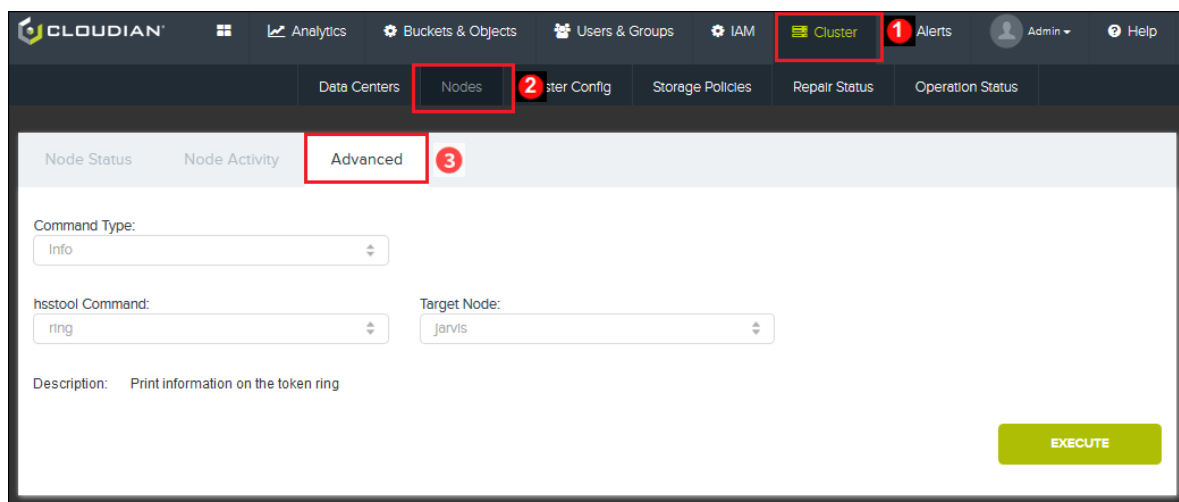
**Note** For statistics that measure quantities of data, the metric used on a graph's Y-axis auto-scales to units that are most appropriate for the particular quantities being conveyed. Specifically, for a given statistic the Y-axis may be expressed in terms of bytes, KBs, MBs, or GBs, depending on the 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.

### 5.7.4. Node Advanced

Path: **Cluster** → **Nodes** → **Advanced**



Supported tasks:

- **"Info Commands"** (page 326)
- **"Maintenance Commands"** (page 327)
- **"Redis Monitor Commands"** (page 327)
- **"Disk Management Commands"** (page 327)
- **"Collect Diagnostics"** (page 328)
- **"Start Maintenance Mode"** (page 329)
- **"Uninstall Node"** (page 330)

#### 5.7.4.1. Info Commands

In the CMC's [Node Advanced](#) page, with Command Type "Info" selected, you can execute any of the following *hsstool* commands which retrieve information about your system:

- [ring](#) — View vNode info for whole cluster (see **"hsstool ring"** (page 712))
- [info](#) — View vNode and data load info for a physical node (see **"hsstool info"** (page 658))
- [status](#) — View summary status for whole cluster (see **"hsstool status"** (page 714))

- [opstatus](#) — View status of repair or cleanup operations (see "**hsstool opstatus**" (page 666))
- [proactiverepairq](#) — View status of proactive repair queues (see "**hsstool proactiverepairq**" (page 674))
- [repairqueue](#) — View auto-repair schedule (see "**hsstool repairqueue**" (page 707))
- [ls](#) — View vNode and data load info per mount point (see "**hsstool ls**" (page 660))
- [whereis](#) — View storage location information for an S3 object (see "**hsstool whereis**" (page 721))
- [metadata](#) — View metadata for an S3 object (see "**hsstool metadata**" (page 662))
- [trmap](#) — View a list of active token range map snapshots (see "**hsstool trmap**" (page 717))

#### 5.7.4.2. Maintenance Commands

In the CMC's [Node Advanced](#) page, with Command Type "Maintenance" selected, you can execute any of the following *hsstool* commands for acting on the data and metadata in your system:

- [cleanup](#) — Clean a node of replicated data that doesn't belong to it (see "**hsstool cleanup**" (page 644))
- [cleanupec](#) — Clean a node of erasure coded data that doesn't belong to it (see "**hsstool cleanupec**" (page 651))
- [autorepair](#) — Enable or disable auto-repair (see "**hsstool repairqueue**" (page 707))
- [proactiverepair](#) -- Enable or disable or stop or start proactive repair (see "**hsstool proactiverepairq**" (page 674))
- [repair](#) — Repair the replicated data on a node (see "**hsstool repair**" (page 686))
- [repairec](#) — Repair the erasure coded data on a node (see "**hsstool repairec**" (page 697))
- [repaircassandra](#) — Repair the system and object metadata on a node (see "**hsstool repair-cassandra**" (page 695))
- [rebalance](#) — Rebalance some data from the cluster to a newly added node (see "**hsstool rebalance**" (page 679))

#### 5.7.4.3. 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 Redis Credentials master role or the Redis QoS master role (see "**Move the Redis Credentials Master or QoS Master Role**" (page 458))
- [getClusterInfo](#) — Get information about the Redis Credentials cluster or the Redis QoS cluster (see "**get cluster**" (page 727) )

**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 726).

#### 5.7.4.4. 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:

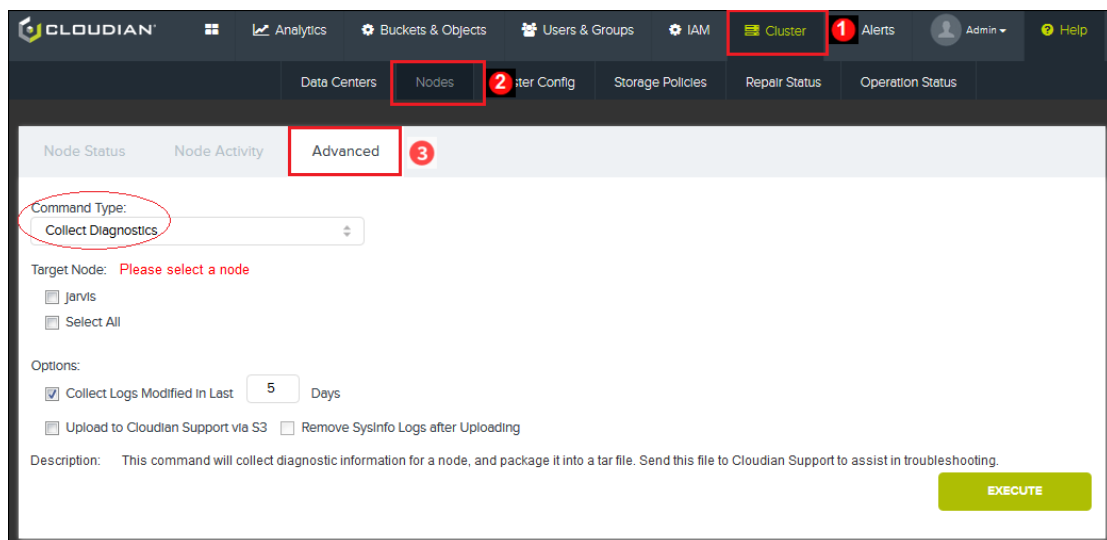
- [disableDisk](#) — Temporarily disable a disk (see **"Disabling a HyperStore Data Disk"** (page 479))
- [enableDisk](#) — Re-enable a disk that is currently disabled (see **"Enabling a HyperStore Data Disk"** (page 480))
- [replaceDisk](#) — Replace a disk (see **"Replacing a HyperStore Data Disk"** (page 482))

#### 5.7.4.5. Collect Diagnostics

When you are experiencing system problems, Clouidian 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 Clouidian 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 192).

### *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 Cloudian 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 removed. This retention period is configurable by the "**cleanup\_sysinfo\_logs\_timelimit**" (page 516) setting in [common.csv](#).

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.

### 5.7.4.6. Start Maintenance Mode

Through the CMC you can place a node into "maintenance mode" so that no S3 write or read requests will be directed to the node and the node will not generate any alerts. If the node is hosting a Redis master, the master role will be temporarily shifted to a different node.

Within a service region, you can have no more than one node in maintenance mode at a time. While the 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. In the CMC's **Node Status** page, most detailed status information will not be available for the node while it is under maintenance.

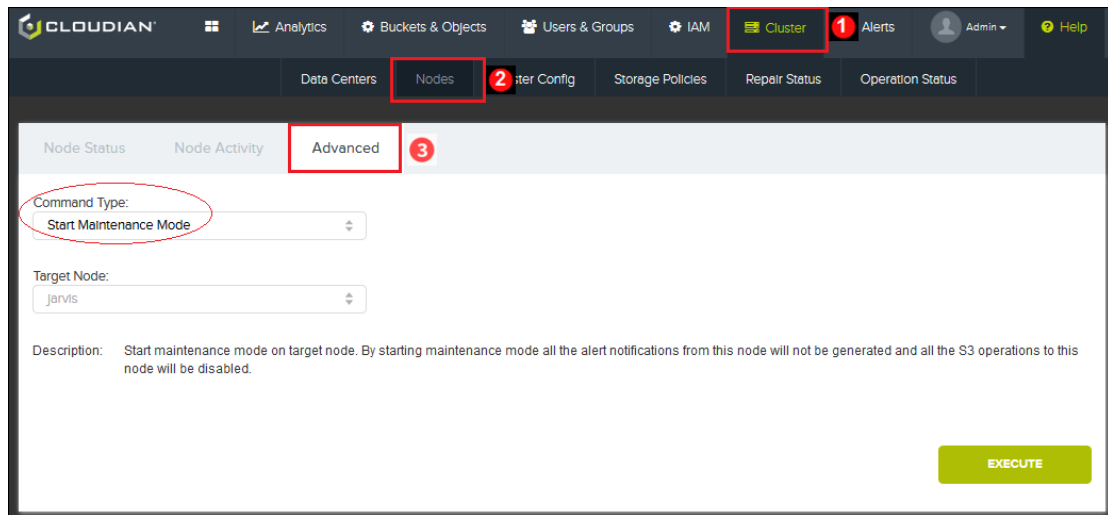
You can subsequently use the CMC to take the node out of maintenance mode and put it back into regular service.

**Note** For you to put a node into maintenance mode, all other nodes in the region must be up, and all services on those other nodes must be up. The system will not allow you to put a target node into maintenance mode if other nodes are down in the system, or if some services are down on other nodes in the system.

When a node is in maintenance mode, the system supports using fallback consistency levels for S3 write or read requests for which the node is an endpoint (if you have configured "**Dynamic Consistency Levels**" (page 39).)

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**.

After confirming that all other nodes and services are up in the region, the system will put the target node into maintenance mode.

**Note** If any S3 requests 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.

#### 5.7.4.6.1. 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**.

The target node is put back into regular service.

#### 5.7.4.7. 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 443).

### 5.7.5. Cluster Information

Path: **Cluster** → **Cluster Config** → **Cluster Information**

The screenshot shows the Cloudian CMC interface. The top navigation bar includes 'Cluster' (highlighted with a red box and a red '1'), 'Alerts (1)', 'Admin', and 'Help'. Below this, the 'Cluster Config' sub-tab is selected (highlighted with a red box and a red '2'). The main content area is divided into two tabs: 'Cluster Information' (selected, highlighted with a red box and a red '3') and 'Configuration Settings'. The 'Cluster Information' tab displays the following sections:

- VERSION INFORMATION**
  - APPLICATION VERSION: 7.2.2 Compiled: 2020-05-21 18:09
- LICENSE INFORMATION**
  - EXPIRE DATE: May-21-2020 13:54 -0700
  - LICENSED MAX NET STORAGE: 100.00 TB
  - LICENSED MAX TIERED STORAGE: Unlimited
  - NET STORAGE USED: 28.00 KB
  - TIERED STORAGE USED: 0
  - OBJECT LOCK LICENSE: Disabled
  - HYPERIQ LICENSE: BASIC
- SERVICE INFORMATION**
  - CMC ADMIN SERVER HOST: s3-admin.mycloudianhyperstore.com:19443
  - CASSANDRA CLUSTER NAME: Cloudianregion1
  - S3 ENDPOINT (HTTP): s3-region1.mycloudianhyperstore.com:80
  - S3 ENDPOINT (HTTPS): s3-region1.mycloudianhyperstore.com:443
  - S3 WEBSITE ENDPOINT: s3-website-region1.mycloudianhyperstore.com
  - REDIS CREDENTIALS MASTER HOST: jarvis
  - REDIS CREDENTIALS SLAVE HOST(S): not installed
  - REDIS QOS MASTER HOST: jarvis
  - REDIS QOS SLAVE HOST(S): not installed

At the bottom of the License Information section, there is a 'Browse...' button (disabled), a text field 'No file selected.', an 'UPDATE LICENSE' button, and a '+ REQUEST LICENSE' link.

Supported tasks:

- View Cluster Information (below)
- **"Renew and Install a License"** (page 337)

The **Cluster Information** page displays summary information about your HyperStore system. The page shows the summary information items below.

#### 5.7.5.0.1. Version Information

##### *Application Version*

Your current HyperStore software version

#### 5.7.5.0.2. 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 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. All incoming S3 requests will be rejected with a "503 Service Unavailable" error response. The response also includes the expiry header described above.
- You can still log into the CMC to perform system administration functions (including applying an updated license), but you will not be able to access users' stored S3 objects.

For information on renewing your license, see **"Renew and Install a License"** (page 337).

#### *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 1GB object is replicated three times in your system it counts as only 1GB 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 1GB object is replicated three times in your system it counts as 3GB 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 15).

#### *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 15).

#### *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 1MB S3 object is replicated three times in your system it counts as only 1MB 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 1MB S3 object is replicated three times in your system it counts as 3MB 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 Feature Overview"** (page 186)

#### *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 support or non-support of the WORM (Object Lock) feature:

- Disabled -- Your license does not support the Object Lock feature.
- Enabled -- Your license supports the Object Lock feature.

**Note** To use the Object Lock feature, along with having a license that supports this feature you must enable the HyperStore Shell (HSH) and disable the root password on your HyperStore nodes. For more information see **"Enabling the HSH and Managing HSH Users"** (page 90) and **"WORM (Object Lock)"** (page 121).

#### *HyperIQ License*

Cloudian HyperIQ is a solution for dynamic visualization and analysis of HyperStore monitoring 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 CMC's **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 OS and service status monitoring are supported indefinitely.
- Enterprise <expiration date> -- HyperIQ dashboards for 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.

#### 5.7.5.0.3. 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 457).

##### *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" in the HyperStore Installation Guide.

**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](#) cluster in this service region. In a multi-region HyperStore system, each region has an independent Cassandra cluster. The system automatically derives a Cassandra cluster 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 (port 80), and optionally HTTPS (port 443). S3 client applications connect to these endpoints to submit S3 API calls.

For each region's S3 endpoint, all of the HyperStore nodes within that region should service requests to the endpoint. In a production environment, typically you would have the S3 service endpoint resolve to the IP addresses of one or more load balancers within the region, and the load balancers would in turn distribute traffic across all the HyperStore nodes in the region. For more information see "DNA Set-Up" in the HyperStore Installation Guide

**Note** HTTPS for the S3 Service is not implemented by default. To set this up, see **"HTTPS Support (TLS/SSL)"** (page 114).

#### *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.

For each region's S3 website endpoint, all of the HyperStore nodes within that region should service requests to the endpoint. In a production environment, typically you would have the S3 website endpoint resolve to the IP addresses of one or more load balancers within the region, and the load balancers would in turn distribute traffic across all the HyperStore nodes in the region. For more information see "DNA Set-Up" in the HyperStore Installation Guide

#### *Redis Credentials Master Host*

Host on which the [Redis Credentials master node](#) is located. In a multi-region system, there is just one Redis Credentials master for the whole system.

If the one Redis Credentials 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 [Redis Credentials slave nodes](#) are located. This is typically two nodes per data center.

**Note** If you upgraded from a HyperStore version older than 6.0, then by default you will have just one Redis Credentials slave node per data center.

#### *Redis QoS Master Host*

Host on which this service region's [Redis QoS master node](#) is located. In a multi-region system, each region has its own Redis QoS master.

#### *Redis QoS Slave Host(s)*

Hosts on which the [Redis QoS 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 this service region, the external NTP servers to which the HyperStore cluster'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 598).

#### *Internal NTP Host*

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 598).

#### *Puppet Master Host*

Host on which the active [Puppet Master](#) is located. This is just one host per whole system, even for a multi-region system.

#### *Puppet Master Backup Host*

Host that is configured to be the backup host for the [Puppet Master](#). This is just one host per whole system, even for a multi-region system. If the node on which the active Puppet Master is located fails, you can **manually** fail over the Puppet Master role to the backup host. Automatic fail-over of this role is not supported. For instructions see **"Move the Puppet Master Primary or Backup Role"** (page 468).

If the one Puppet Master 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 Puppet Master*

On the Puppet Master 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 [pushing configuration file edits out to the cluster](#) or [restarting services](#). By default the installation staging directory for HyperStore version 7.2.3 is `/opt/cloudian-staging/7.2.3`.

#### *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>`.

### 5.7.5.1. 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.

#### 5.7.5.1.1. Request a License Renewal

Click **Request License** to send an email to [cloudian-license@cloudian.com](mailto: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](mailto:cloudian-license@cloudian.com).

#### 5.7.5.1.2. 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**.

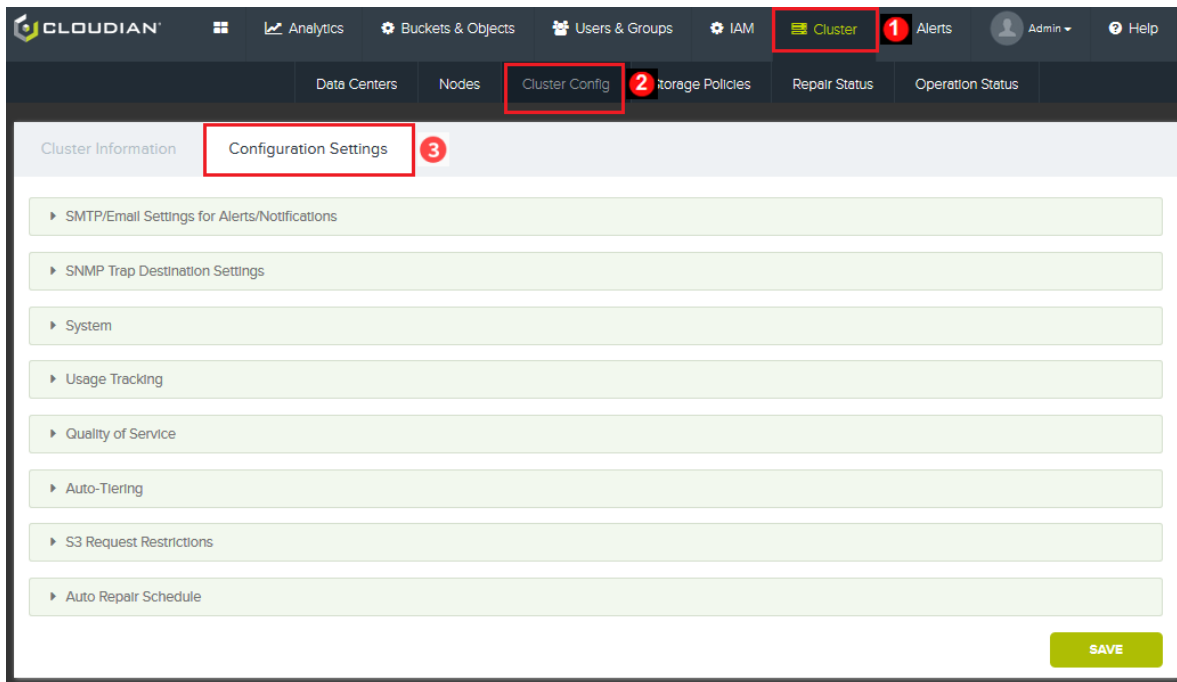
The CMC pushes the license file to the Puppet master node, which in turn propagates the file to all your HyperStore nodes. The Puppet master then uses JMX to trigger your HyperStore nodes to dynamically reload the license file. You do not need to restart any services.

After the process completes, if you refresh the **Cluster Information** page you should see updated information in the **License Information** section.

**Note** The system saves your old license file on the Puppet master node as `<old-license-file-name>.<-timestamp>` (in directory `/etc/puppet/modules/baselayout/files`).

### 5.7.6. 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.

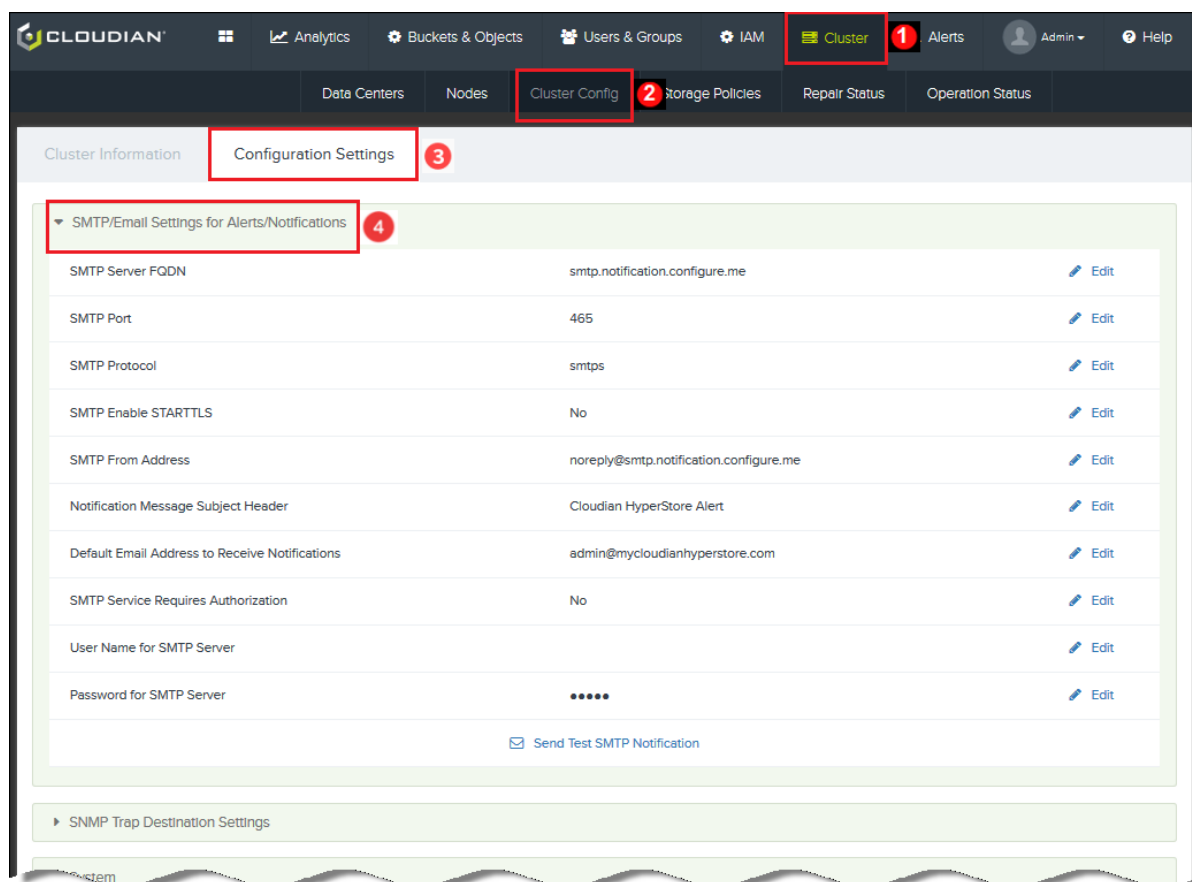
You can edit any of the following types of settings. **Be sure to click Save at the bottom of the page** to save your changes.

- **"SMTP/Email Settings for Alerts/Notifications"** (page 338)
- **"SNMP Trap Destination Settings"** (page 341)
- **"System Settings"** (page 342)
- **"Usage Tracking Settings"** (page 343)
- **"Quality of Service Settings"** (page 345)
- **"Auto-Tiering Settings"** (page 346)
- **"S3 Request Restriction Settings"** (page 349)
- **"Auto-Repair Schedule Settings"** (page 350)

**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.

#### 5.7.6.1. SMTP/Email Settings for Alerts/Notifications

The settings in this section of the CMC's **Configuration Settings** page pertain to the SMTP service 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 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 cluster. 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 service 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*.

The HyperStore system uses the default SMTP ports (25 for *smtp* or 465 for *smtps*). If you want to use non-default SMTP ports, consult with Cloudian Support.

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 Notification 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.

Default = `admin@mycloudianhyperstore.com`

#### *SMTP Service Requires Authorization*

Whether the SMTP service that the HyperStore system will use to send alert notification emails 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

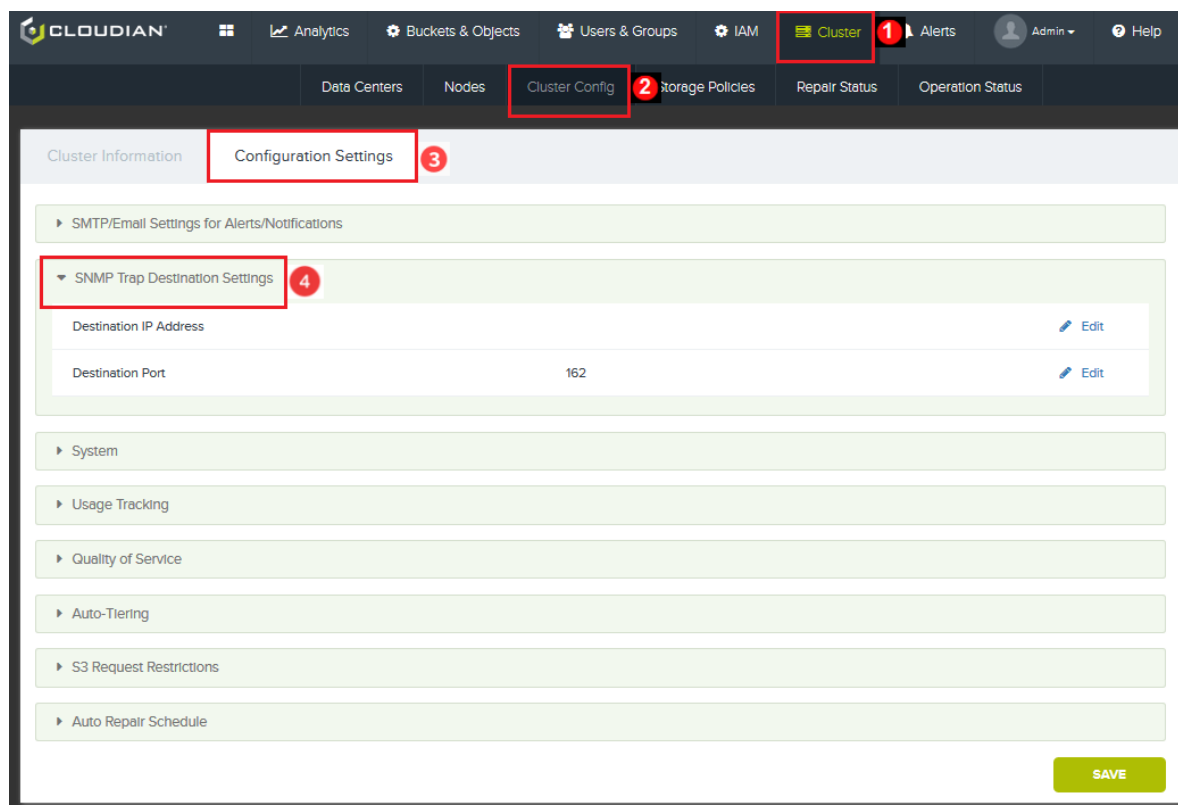
#### *Password for SMTP Server*

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

### 5.7.6.2. 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 system alerts sent as SNMP traps.



After making any edits, click **Save** at the bottom of the page to dynamically apply the configuration change to the cluster.

**Note** For information about enabling SNMP trap sending for alerts, see **"Alert Rules"** (page 390).

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 IP Address*

IP address of the SNMP manager to which the CMC will send system alerts (for alert types for which you've enabled SNMP trap sending). Do not enter multiple IP addresses.

Default = None

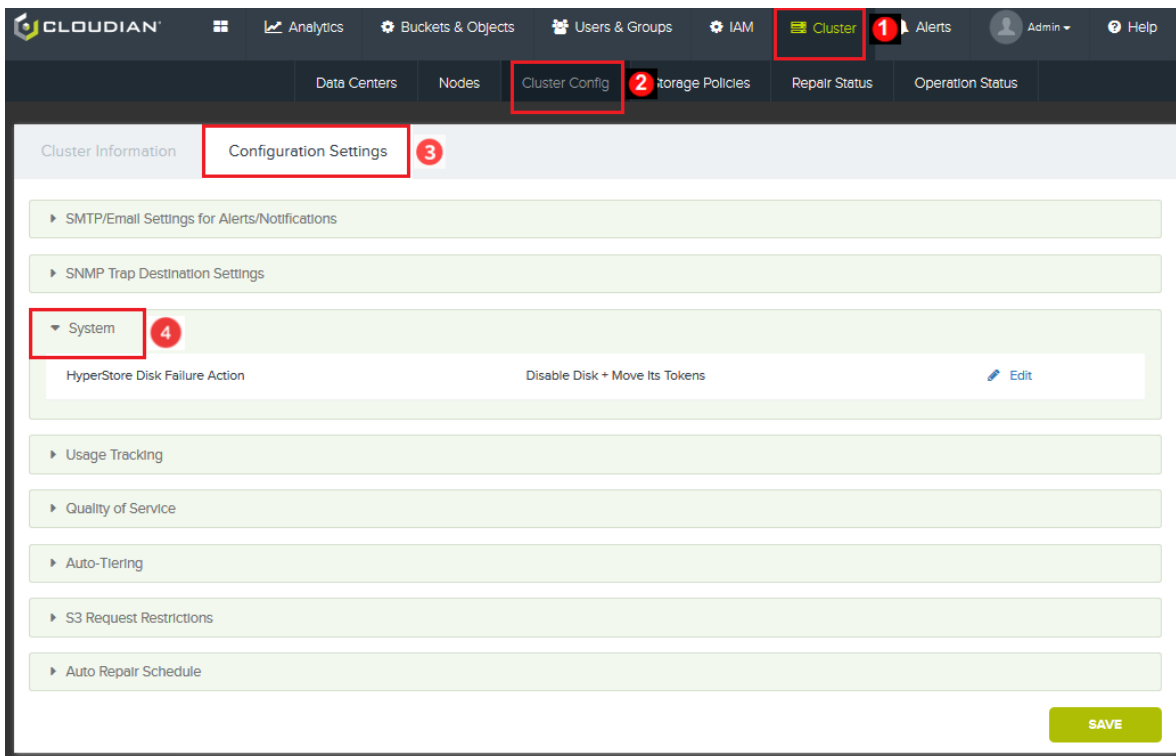
#### *Destination Port*

Listening port used by the SNMP manager.

Default = 162

### 5.7.6.3. System Settings

The settings in this section of the CMC's **Configuration Settings** page configure storage system operations.



After making any edits, click **Save** at the bottom of the page to dynamically apply the configuration change to the cluster.

#### *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 480) operation or the **"Replacing a HyperStore Data Disk"** (page 482) 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 160).

- **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. This threshold is configurable by these settings in *hyperstore-server.properties.erb*:

- **"disk.fail.error.count.threshold"** (page 551)
- **"disk.fail.error.time.threshold"** (page 552)

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.

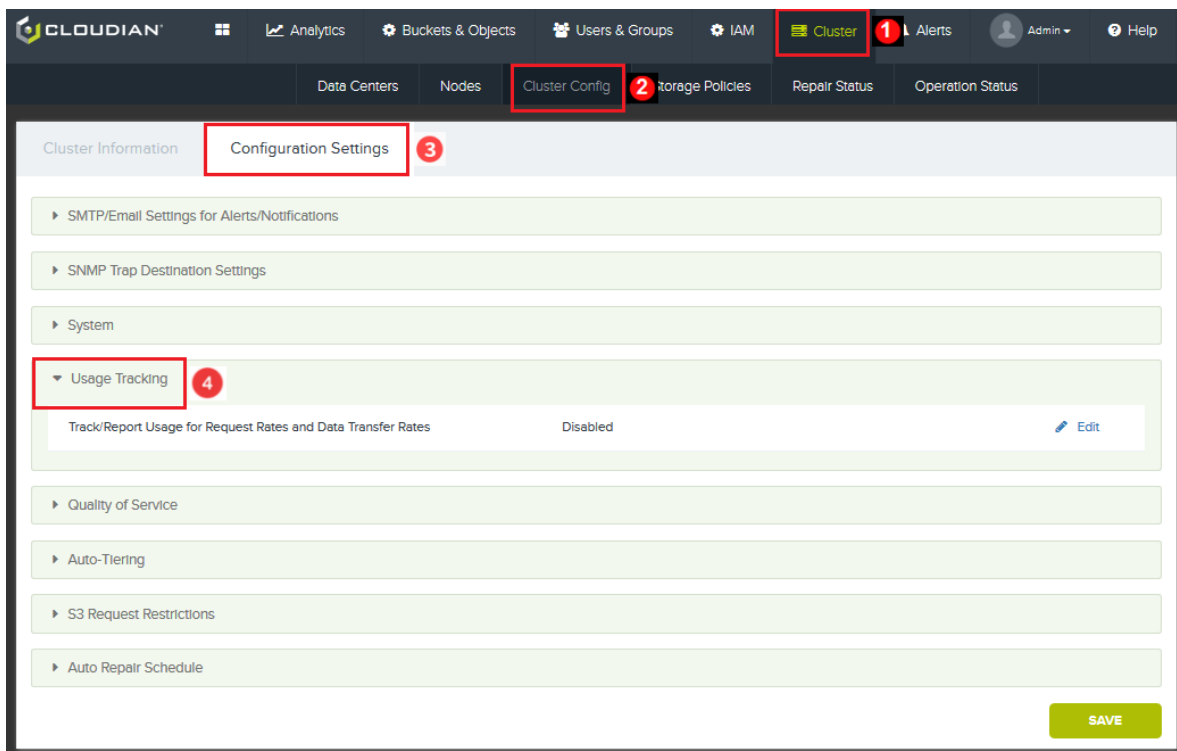
Default = Disable Disk + Move Its Tokens

**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 Cloudian Support if you are using any of these technologies.

#### 5.7.6.4. 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.



After making any edits, click **Save** at the bottom of the page to dynamically apply the configuration change to the cluster.

**Note** For an overview of the HyperStore usage reporting feature, see "**Usage Reporting and Billing Feature Overview**" (page 138).

#### *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.

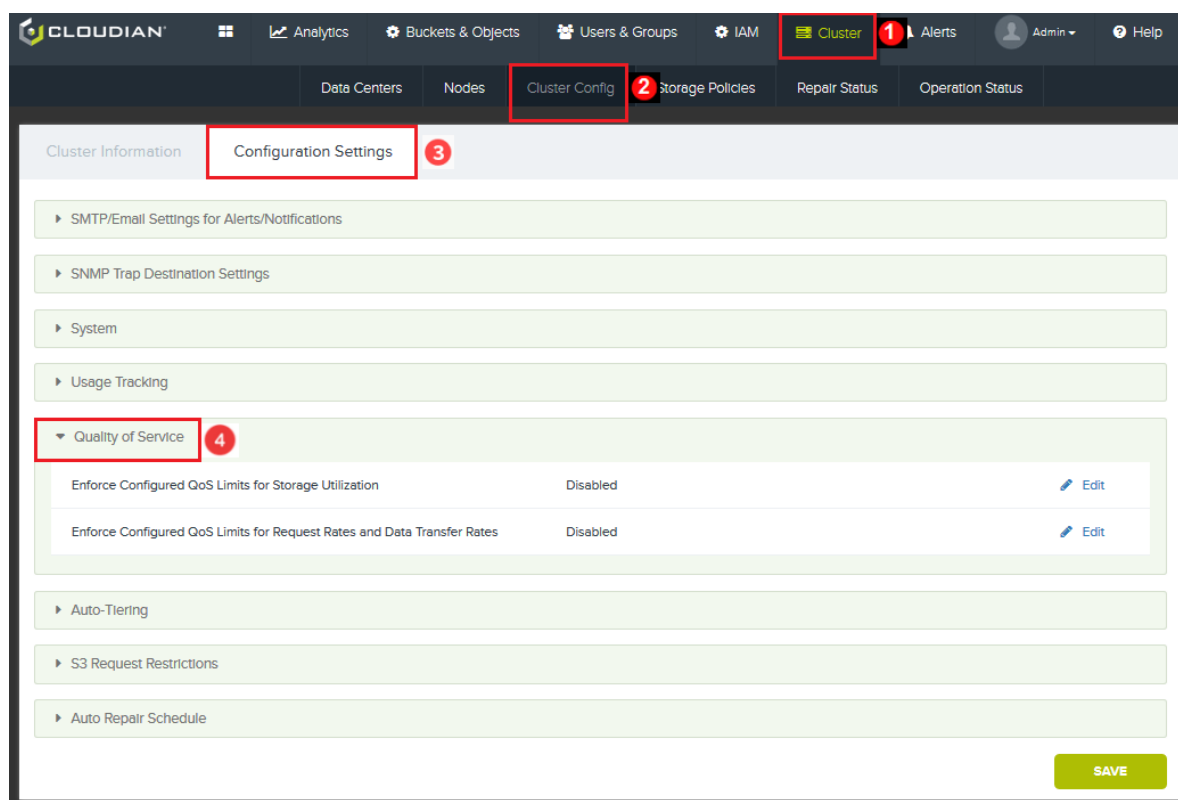
Default = Disabled

**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 Cassandra 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. See "**reports.raw.ttl**" (page 567) in *mts.properties.erb*. 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.

### 5.7.6.5. 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.



After making any edits, click **Save** at the bottom of the page to dynamically apply the configuration change to the cluster.

**Note** For an overview of the HyperStore QoS feature, see "**Quality of Service (QoS) Feature Overview**" (page 135).

#### *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](#).

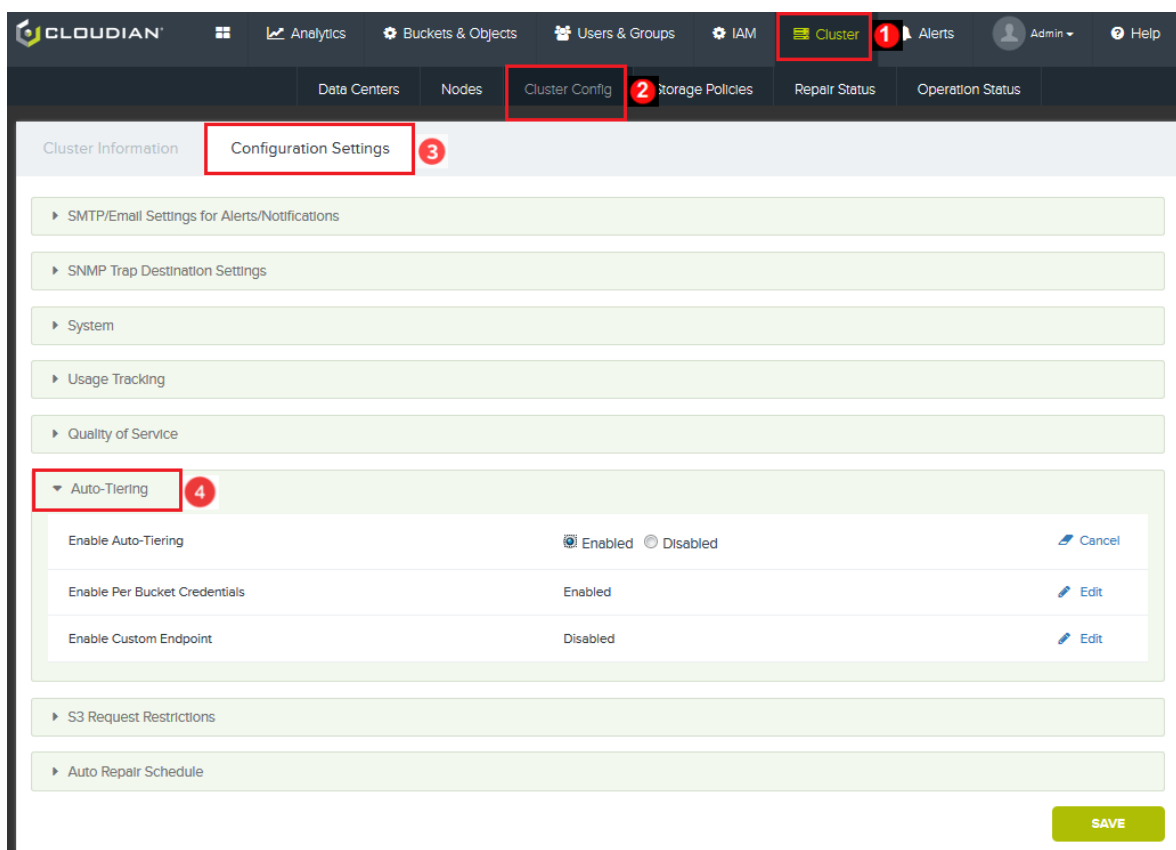
If this setting is **disabled**, then the system will not enforce HTTP request rate, bytes-in rate, and bytes-out rate limits, even if you have set such limits for users and groups.

Enabling this setting is supported only if the *Enforce Configured QoS Limits for Storage Utilization* setting is also enabled.

Default = Disabled

#### 5.7.6.6. Auto-Tiering Settings

The settings in this section of the CMC's **Configuration Settings** page configure the HyperStore system's auto-tiering feature.



After making any edits, click **Save** at the bottom of the page to dynamically apply the configuration change to the cluster.

**Note** For an overview of the HyperStore auto-tiering feature, see **"Auto-Tiering Feature Overview"** (page 176). 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 **"Setting Up Auto-Tiering"** (page 180).

#### 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"** (page 969), 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 you have upgraded from a HyperStore version older than 7.1.4 -- it is not in the default list for new installs of HyperStore 7.1.4 or later).

**Note** You can change this list of destinations and/or their endpoints if you wish. For details see "**Configure Tiering Destinations**" (page 181).

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", this configuration will **override** the list of destinations configured in the "**cmc\_bucket\_tiering\_default\_destination\_list**" (page 541) setting in [common.csv](#). 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.

### 5.7.6.7. 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.

The screenshot shows the Cloudbian CMC interface. The top navigation bar includes links for Analytics, Buckets & Objects, Users & Groups, IAM, Cluster (highlighted with a red box and a red circle with '1'), Alerts (highlighted with a red box and a red circle with '1'), Admin, and Help. Below this, a secondary navigation bar shows Data Centers, Nodes, Cluster Config (highlighted with a red box and a red circle with '2'), Storage Policies, Repair Status, and Operation Status. The main content area is titled 'Cluster Information' and 'Configuration Settings' (highlighted with a red box and a red circle with '3'). Under 'Configuration Settings', there are several expandable sections: SMTP/Email Settings for Alerts/Notifications, SNMP Trap Destination Settings, System, Usage Tracking, Quality of Service, Auto-Tiering, and S3 Request Restrictions (highlighted with a red box and a red circle with '4'). The 'S3 Request Restrictions' section is expanded, showing a table with three rows: 'Put Object Maximum Size (Bytes)' with value '5368709120', 'Multipart Upload Maximum Parts' with value '10000', and 'Maximum Buckets Per User' with value '100'. Each row has an 'Edit' link. At the bottom right of the configuration area is a 'SAVE' button.

Setting	Value	Action
Put Object Maximum Size (Bytes)	5368709120	<a href="#">Edit</a>
Multipart Upload Maximum Parts	10000	<a href="#">Edit</a>
Maximum Buckets Per User	100	<a href="#">Edit</a>

After making any edits, click **Save** at the bottom of the page to dynamically apply the configuration change to the cluster.

#### *Put Object Maximum Size (Bytes)*

The maximize single-part object size that S3 clients will be allowed to upload to the system through a PUT Object request or POST Object request, in number of bytes. If a client submits a PUT Object 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 Upload Part request.

For best upload performance it's recommended for S3 clients to use the S3 [Initiate Multipart Upload](#) operation for objects larger than 100MB (rather than PUT Object or POST Object).

Default = 5368709120 (5 GiB)

**Note** In HyperStore version 7.1.x and older, the default was 5000000000 (5 GB). If you upgraded from HyperStore 7.1.x, your system retains the default of 5000000000 unless you change the setting through the CMC's **Configuration Settings** page.

#### *Multipart Upload Maximum Parts*

Maximum number of parts to allow per [Initiate Multipart Upload](#) request.

Each individual part can be no larger than the configured "Put Object Maximum Size (Bytes)" limit.

Default = 10000

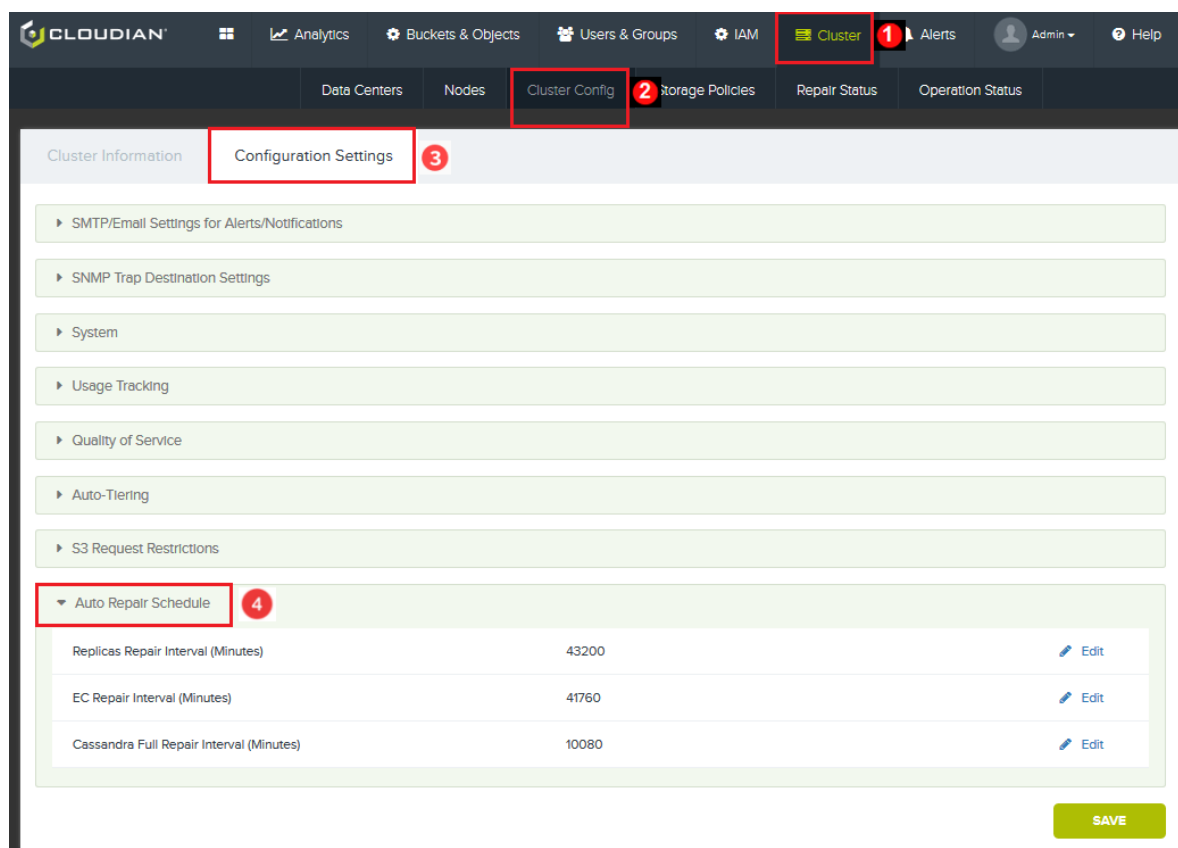
#### *Maximum Buckets Per User*

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

### 5.7.6.8. 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.



After making any edits, click **Save** at the bottom of the page to dynamically apply the configuration change to the cluster.

**Note** For more information about the auto-repair feature see "**Automated Data Repair Feature Overview**" (page 150)

#### *Replicas Repair Interval (Minutes)*

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.

The system maintains a queue of nodes scheduled for auto-repair of replicated object data, and the queue is processed in such a way that within a service region *hsstool repair* will run on only one target node at a time. Repair of the node that is next in queue will not start until the repair operation completes on the node on which *hsstool repair* is currently running. Consequently if you have a lot of nodes and/or a high volume of data in your system, the actual time between repairs of a given node may be longer than the scheduled interval.

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 "**auto\_repair\_computedigest\_run\_number**" (page 518) setting in *common.csv*. By default "computedigest" is not used in auto-repair runs.

**Note** The system allows repair operations of different types -- such as an *hsstool repair* operation and an *hsstool repairec* operation -- to run concurrently within the same service region.

#### *EC Repair Interval (Minutes)*

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.

The system maintains a queue of nodes scheduled for auto-repair of erasure coded object data, and the queue is processed in such a way that within a service region *hsstool repairec* will run on only one target node at a time. Repair of the node that is next in queue will not start until the repair operation completes on the node on which *hsstool repairec* is currently running. Consequently if you have a lot of nodes and/or a high volume of data in your system, the actual time between repairs of a given node may be longer than the scheduled 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.

In a multi- data center HyperStore service region, the erasure coded data auto-repair queue is ordered in such a way that the target nodes alternate among the data centers -- for example after a repair completes on a target node in DC1, then the next target node will be from DC2, and then after that completes the next target node will be from DC3, and then after that completes the next target node will be from DC1 again, and so on.

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)

**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 "**auto\_repair\_computedigest\_run\_number**" (page 518) setting in *common.csv*. By default "computedigest" is not used in auto-repair runs.

**Note** The system allows repair operations of different types -- such as an *hsstool repair* operation and an *hsstool repairec* operation -- to run concurrently within the same service region.

#### *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 (system metadata and object metadata stored in Cassandra) . 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).

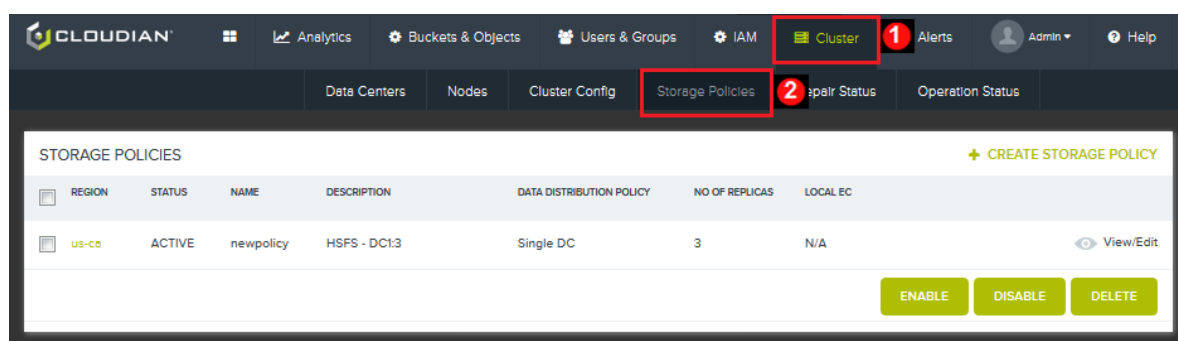
The system maintains a queue of nodes scheduled for auto-repair of Cassandra data, and the queue is processed in such a way that within a service region Cassandra repair will run on only one target node at a time. Repair of the node that is next in queue will not start until the repair operation completes on the node on which a Cassandra repair is currently running.

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)

### 5.7.7. Storage Policies

Path: **Cluster** → **Storage Policies**



Supported tasks:

- "Add a Storage Policy" (page 353)
- "Edit a Storage Policy" (page 377)
- "Designate a Default Storage Policy" (page 377)
- "Disable a Storage Policy" (page 378)
- "Delete a Storage Policy" (page 379)

**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 84).

#### 5.7.7.1. 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 pre-configure one or more storage policies. Users when they create a new storage bucket can then 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 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.

**Note** 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.

In 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.

**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

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
sfo-del-region1	DC1	1 of 3	
		2 of 3	disable
		3 of 3	

CONSISTENCY SETTING

CONSISTENCY LEVEL	READ	WRITE
ALL	<input type="checkbox"/>	<input type="checkbox"/>
QUORUM	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
ONE	<input type="checkbox"/>	

GROUP VISIBILITY

Please select a Group

ADD

Compression Type

NONE

Server-side Encryption

NONE

SAVE

CANCEL

2. Complete the sections of the **Create New Policy** interface.

- [Policy Name and Description](#)
- [Data Distribution Scheme and Data Center Assignment](#)
- [Consistency Setting](#)
- [Group Visibility](#)
- [Compression](#)
- [Server-Side Encryption](#)

**Note** If you have configured Elasticsearch integration for your HyperStore system, then when you create (or edit) a storage policy you will have an option to "Enable metadata search". For information about this feature see "**Elasticsearch Integration for Object Metadata**" (page 171).

- Click **Save** to create the new 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.

#### 5.7.7.1.1. Create New 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.

### 5.7.7.1.2. Create New Storage Policy -- Data Distribution Scheme and Data Center Assignment

In the "Data Distribution Scheme" and "Data Center Assignment" sections of the [Create New Policy](#) 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 84).

### 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	
		2 of 3	disable
		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.

**IMPORTANT !** Whatever  $k+m$  configuration you choose, the system will maintain  $(2m)-1$  replicas of each object's metadata (in Cassandra). For more detail see "**Object Metadata Replication**" (page 80).

**Consult with Cloudian Support to ensure that the disks or SSDs on which you are storing Cassandra data have sufficient capacity** to support your desired  $k+m$  configuration with its associated metadata storage requirements as well as the temporary storage overhead needed during Cassandra repair operations.

**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

O

↓

REPLICAS

O

O

O

DC-1

DC-2

...

DC-n

Replicated EC

OBJECT

O

↓

REPLICAS

O

O

O

↓

EC FRAGMENTS

X

X

X

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
<div>region1</div>	DC1	<input type="checkbox"/>
	DC2	<input type="checkbox"/>
	DC3	<input type="checkbox"/>

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:

362

- 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).

**IMPORTANT !** In each data center included in the policy, the system will maintain  $(2m)-1$  replicas of each object's metadata (in Cassandra). For more detail see **"Object Metadata Replication"** (page 80).

**Consult with Cloudian Support to ensure that the disks or SSDs on which you are storing Cassandra data have sufficient capacity** to support your desired  $k+m$  configuration with its associated metadata storage requirements as well as the temporary storage overhead needed during Cassandra repair operations.

**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

OBJECT

DC-1

DC-2

DC-n

REPLICAS

Replicated EC

OBJECT

DC-1

DC-2

DC-n

REPLICAS

EC FRAGMENTS

EC Across Datacenters

OBJECT

DC-1

DC-2

DC-n

EC FRAGMENTS

ERASURE CODING K+M VALUE

7+5

DATACENTER ASSIGNMENT

Please select the correct amount of DCs.

REGION	DATACENTER	SELECTED
<div>region1</div>	DC1	<input type="checkbox"/>
	DC2	<input type="checkbox"/>
	DC3	<input type="checkbox"/>

With EC Across Data Centers, 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 Dis-tributed
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

364

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).

**IMPORTANT !** Distributed across all the data centers included in the policy, the system will maintain a **total** of  $(2m)-1$  replicas of each object's metadata (in Cassandra). For more detail see **"Object Metadata Replication"** (page 80).

**Consult with Cloudian Support to ensure that the disks or SSDs on which you are storing Cassandra data have sufficient capacity** to support your desired  $k+m$  configuration with its associated metadata storage requirements as well as the temporary storage overhead needed during Cassandra repair operations.

**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.

#### 5.7.7.1.3. Create New Storage Policy -- Consistency Setting

In the "Consistency Setting" section of the [Create New Policy](#) 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 84).

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
<a href="#">ALL</a>	<a href="#">ALL</a>
<a href="#">QUORUM</a> (default)	<a href="#">QUORUM</a> (default)
<a href="#">ONE</a>	--

### 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
<a href="#">ALL</a>	<a href="#">ALL</a>
<a href="#">QUORUM</a> (default)	<a href="#">QUORUM</a> (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
<a href="#">ALL</a>	<a href="#">ALL</a>
--	<a href="#">EACH QUORUM</a>
<a href="#">QUORUM</a> (default)	<a href="#">QUORUM</a> (default)
<a href="#">LOCAL QUORUM</a>	<a href="#">LOCAL QUORUM</a>
<a href="#">ONE</a>	--

### Replicated EC

With a "Replicated EC" scheme you can choose from the following consistency levels.

Read	Write
<a href="#">ALL</a>	<a href="#">ALL</a>

Read	Write
--	<a href="#">EACH QUORUM</a>
<a href="#">LOCAL QUORUM</a>	<a href="#">LOCAL QUORUM</a>
<a href="#">ANY QUORUM</a> (default)	<a href="#">ANY QUORUM</a> (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. With the other settings the system will read from all data centers when trying to get "k" unique fragments.

## EC Across Data Centers

With an "EC Across Data Centers" scheme you can choose from the following consistency levels.

Read (metadata only)	Write
<a href="#">ALL</a>	<a href="#">ALL</a>
<a href="#">QUORUM</a> (default)	<a href="#">QUORUM</a> (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 39), 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.

### 5.7.7.1.4. 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 endpoint 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.

- **"Consistency Level "ALL""** (page 369)
- **"Consistency Level "QUORUM""** (page 374)
- **"Consistency Level "EACH QUORUM""** (page 371)
- **"Consistency Level "LOCAL QUORUM""** (page 372)
- **"Consistency Level "ANY QUORUM""** (page 370)
- **"Consistency Level "ONE""** (page 373)

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 84).

**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 39).

### 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 Cassandra) 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 Cassandra, and all the object's data replicas' file digests in RocksDB -- and then it retrieves one object data replica and returns it to the client.

### Note About Bucket Content List Reads

In the documentation of the supported consistency levels such as "ALL", "QUORUM", and so on (see the cross references above), when read consistency requirements are discussed the focus is on reads of individual objects -- that is, the consistency requirements for successfully implementing S3 *GET Object* requests. It's worth noting however that your configured read consistency requirements also apply to bucket content list reads -- that is, implementing S3 *GET Bucket (List Objects)* requests.

Metadata for objects is stored in two different types of record in Cassandra: 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 *GET Object* request requires reading the object's object-level metadata record and a *GET Bucket (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 buckets content lists. So for example if you use a QUORUM read consistency requirement, then in order to successfully execute a *GET Bucket (List Objects)* request the system must be able to read a QUORUM of the bucket-level object metadata records for the bucket.

For more on the meaning of QUORUM and the other supported consistency levels, see the cross references above.

#### 5.7.7.1.5. 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
<b>Read ALL</b>	For an S3 GET to succeed, the system must succeed in reading all <a href="#">object data replicas</a> 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.
<b>Write ALL</b>	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.

### 5.7.7.1.6. 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
<b>Read ANY QUORUM</b>	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 <a href="#">dynamic consistency levels</a> ). 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.
<b>Write ANY QUORUM</b>	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

	Replicated EC
	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.

#### 5.7.7.1.7. 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
<b>Write EACH QUORUM</b>	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.

**5.7.7.1.8. 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
<b>Read LOCAL QUORUM</b>	For an S3 GET to succeed, the system must succeed in reading a majority of the local DC's <a href="#">object data replicas</a> 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.
<b>Write LOCAL QUORUM</b>	For an S3 PUT to succeed, the system must succeed in writing a majority of the local DC's object data replicas and a majority of the local DC's object metadata replicas.	For an S3 PUT to succeed, the system must succeed in writing "k"+1 object data fragments within the local DC and a 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.

### **5.7.7.1.9. 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.

### 5.7.7.1.10. 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
<b>Read QUORUM</b>	For an S3 GET to succeed, the system must succeed in reading a majority of <a href="#">object data replicas</a> 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.
<b>Write QUORUM</b>	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.

### 5.7.7.1.11. Create New Storage Policy -- Group Visibility

In the "Group Visibility" section of the [Create New Policy](#) 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.

#### 5.7.7.1.12. Create New Storage Policy -- Compression

In the "Compression" field of the [Create New Policy](#) interface, select the type of compression (if any) to use for S3 objects stored in the HyperStore File System. This applies to replicated S3 object data and erasure coded S3 object data stored in the HSFS. It does not apply to data stored in Cassandra.

Supported options are:

- [Snappy](#)
- [Zlib](#)
- [LZ4](#)
- None (no compression)

When enabled, compression is applied to incoming S3 objects by the S3 Service, before those objects are transmitted to the HyperStore Service and placed into storage. 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 S3 objects as they come into the system, not retroactively to objects that are already stored in the system.

Each object's compression type (if any) is stored in the object's metadata in Cassandra. 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 **"CopyObject"** (page 939) 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).

#### 5.7.7.1.13. Create New Storage Policy -- Server-Side Encryption

In the "Server-Side Encryption" field of the [Create New Policy](#) 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:

- SSE -- Regular server-side encryption using encryption keys managed by the HyperStore system

- None -- No server-side encryption

Your chosen default method of server-side encryption for the storage policy will apply only to objects for which no server-side encryption method is specified either in the object upload request or in the bucket configuration.

**Object-level and bucket-level server-side encryption settings supersede the storage policy level setting.**

For more information about HyperStore's support for server-side encryption -- including the interaction of object level, bucket level, and storage policy level encryption settings -- see **"Server-Side Encryption"** (page 105).

**Note** 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. The change does not apply to objects that are already in storage.

### 5.7.7.2. 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 data consistency requirements, group visibility, compression, and server-side encryption. For guidance on working with these policy characteristics, see **"Add a Storage Policy"** (page 353).

Note that 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.

**Note** When you click **View/Edit** for a storage policy you can also view the storage policy's data distribution scheme (such as replication factor or EC "k"+"m" values), its data center assignment, and its system-generated policy ID. However these policy attributes are not editable.

### 5.7.7.3. 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								<a href="#">+ CREATE STORAGE POLICY</a>
<input type="checkbox"/>	REGION	STATUS	NAME	DESCRIPTION	DATA DISTRIBUTION POLICY	NO OF REPLICAS	LOCAL EC	
<input type="checkbox"/>	Us-norcal	ACTIVE	HSFS-1	HSFS - DC1:2 - DC2:1	Multi DC	3	N/A	<a href="#">View/Edit</a>
<input type="checkbox"/>	Us-norcal	ACTIVE	EC-1	EC: K2-M1	Multi DC	2	2 + 1	<a href="#">Set As Default</a> <a href="#">View/Edit</a>
<input type="checkbox"/>	Us-norcal	ACTIVE	Test_Policy		Single DC	3	N/A	<a href="#">Set As Default</a> <a href="#">View/Edit</a>
<input type="checkbox"/>	Us-norcal	ACTIVE	Test_Policy_2		Single DC	1	2 + 1	<a href="#">Set As Default</a> <a href="#">View/Edit</a>
								<a href="#">DISABLE</a> <a href="#">DELETE</a>

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.

To designate a 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.

**Note** When a pre-5.2 version of HyperStore is upgraded to 5.2 or newer, the upgrade process automatically creates a storage policy named "DEFAULT\_<region-name>". This policy is configured with the data distribution scheme (replication factor or EC "k"+"m" values) and user data read/write consistency requirements that the pre-5.2 system had been using. This "DEFAULT\_<region-name>" will be your default storage policy until you designate some other policy as the default.

#### 5.7.7.4. 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.

### 5.7.7.5. 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 378). 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.

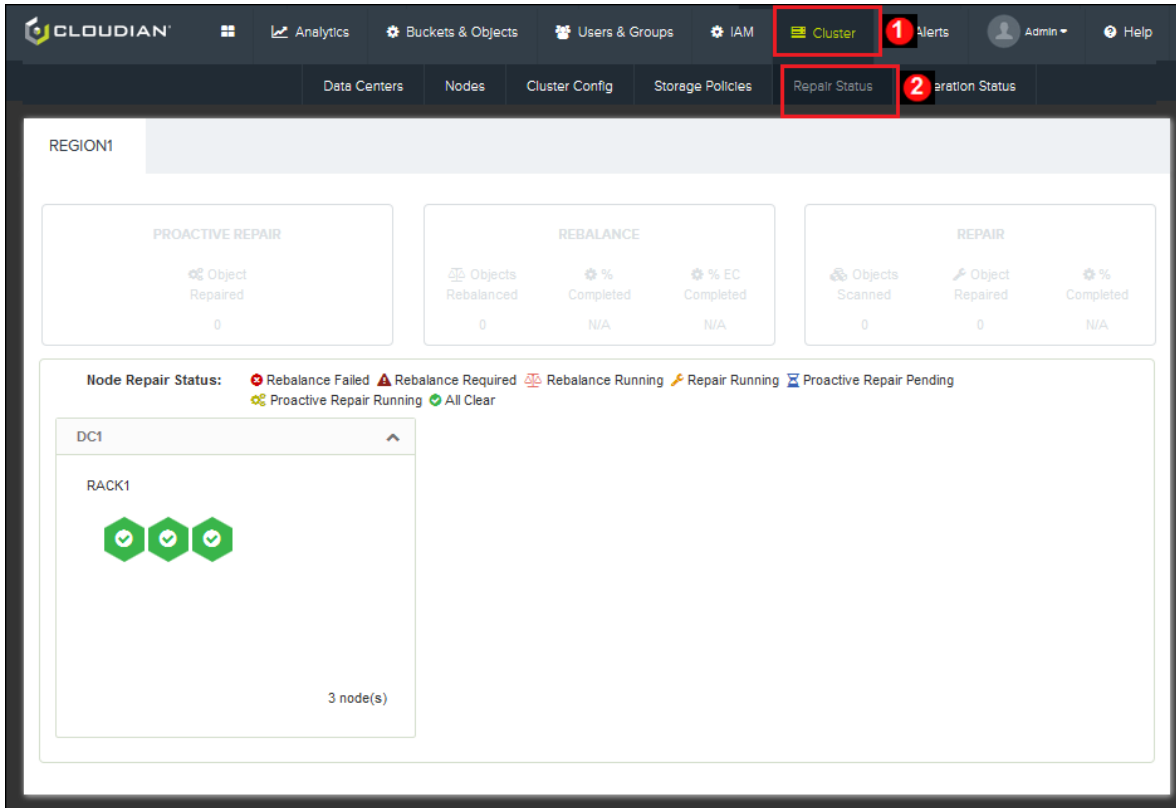
If you are not certain which buckets are currently using a given storage policy, you can use the Admin API method [GET /bppolicy/bucketsperpolicy](#) to retrieve this information.

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 [hsstool cleanup](#) on each node in your cluster, using the `-a` option and the `-policy` option (`hsstool 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 `hsstool 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.

## 5.7.8. Repair Status

Path: **Cluster** → **Repair Status**



Supported tasks:

- View repair status for cluster
- View repair status for a node

### 5.7.8.1. 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 420))
- A *decommission* operation (which the system would automatically initiate as part of the process of **"Removing a Node"** (page 443), 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 repairec](#) 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 *repairec* 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 *repairec* 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 *repairec* operations -- if only a *repairec* operation is running (and not a *repair* operation), then the % Completed field displays "N/A" for not applicable.

### 5.7.8.2. 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 420).



**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 420).



**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.



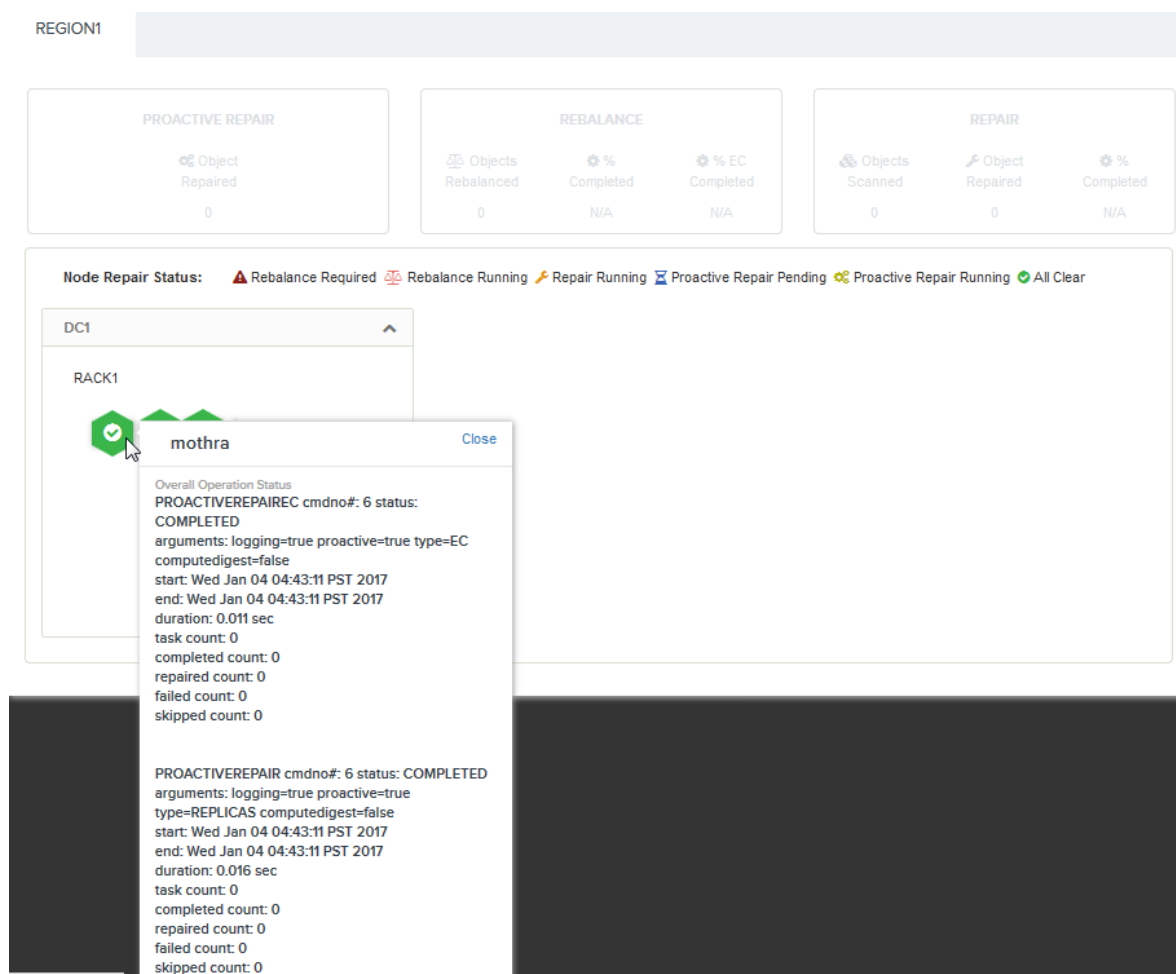
*All Clear* — No repair of any type is in progress on the node, and the node is not currently in need of rebalance or proactive repair.



*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 666).

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 154).

### 5.7.9. Operation Status

Path: **Cluster** → **Operation Status**

OPERATION LIST

Show  entries

Search:

OPERATION NAME	TARGET	STATUS	PROGRESS	START TIME	LAST UPDATE	
addNode	region1:dc1:failhost	<span style="color: red;">❌ failed</span>	<span style="background-color: red; color: white;">failed</span>	Feb-23-2018 07:27	Feb-23-2018 07:28	<a href="#">View</a> <a href="#">Delete</a>
cleanupec	region1:dc1:store1	<span style="color: green;">✅ completed</span>	<span style="background-color: green; color: white;">100%</span>	Feb-23-2018 07:25	Feb-23-2018 07:25	<a href="#">View</a> <a href="#">Delete</a>
repair	region1:dc1:store1	<span style="color: green;">✅ completed</span>	<span style="background-color: green; color: white;">100%</span>	Feb-23-2018 07:25	Feb-23-2018 07:25	<a href="#">View</a> <a href="#">Delete</a>
cleanup	region1:dc1:store1	<span style="color: green;">✅ completed</span>	<span style="background-color: green; color: white;">100%</span>	Feb-23-2018 07:24	Feb-23-2018 07:24	<a href="#">View</a> <a href="#">Delete</a>

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 Status**

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.

Close

In the case of *hsstool* operations these are the same operation details as are provided by [hsstool opstatus](#). 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.

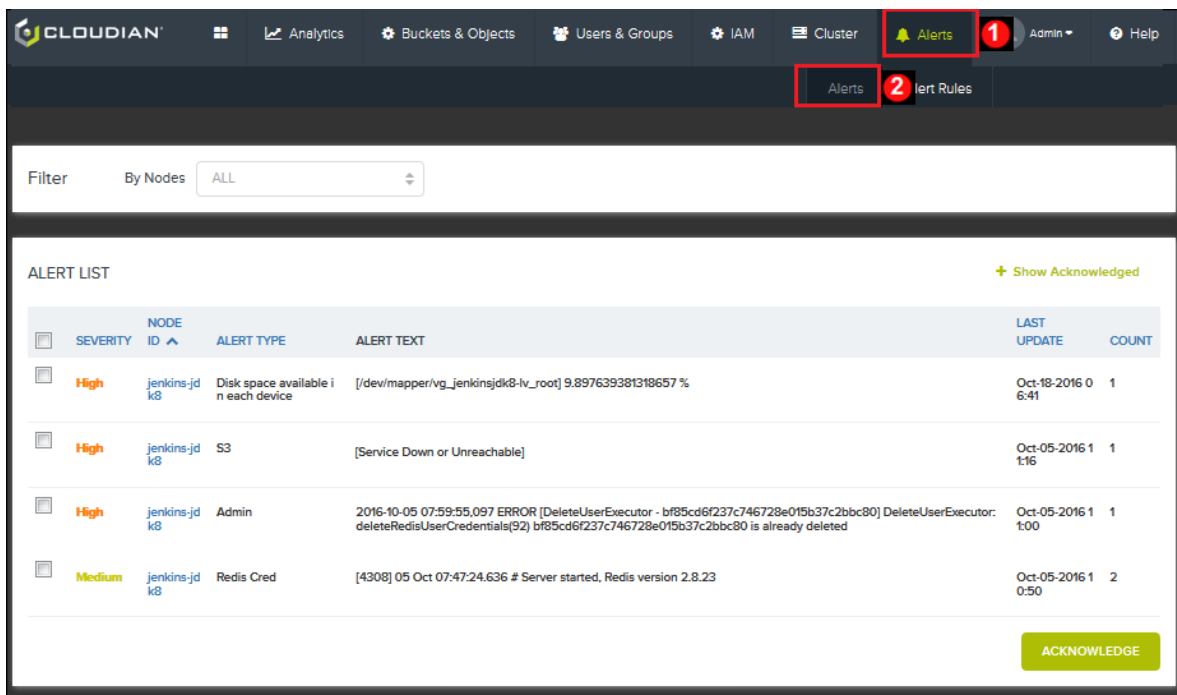
## 5.8. Alerts

The **Alerts** tab contains the following functions:

- **"Alerts"** (page 385)
- [Alert Rules](#)

### 5.8.1. Alerts

Path: **Alerts** → **Alerts**



Supported tasks:

- **"Review Alerts"** (page 386)
- **"Acknowledge Alerts"** (page 389)

**Note** Optionally you can suppress alerting for specific log messages as identified by message code. HyperStore provides a configuration setting for suppressing specific log-based alerts. In the configuration file [common.csv](#), see the setting **"alert\_suppression\_list"** (page 543).

### 5.8.1.1. 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 313) 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, HyperStore, Redis Credentials, Redis QoS, 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-dnsmasq}.

- **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 -- Admin, Cassandra, HyperStore, Redis Credentials, Redis QoS, or S3.

**Note** Along with alerts pertaining to providing S3 service to clients, the "S3" alerts category also includes alerts pertaining to auto-tiering and cross-region replication.

- **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 313) 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 390).

#### 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]"

**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:

```
/etc/init.d/<service-name> status
```

The <service-name> string can be one of {cloudian-s3, cloudian-cassandra, cloudian-hyperstore, cloudian-redis-credentials, cloudian-redis-qos, cloudian-redismon, cloudian-cmc, cloudian-agent, cloudian-dnsmasq}.

- 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 605).
  - Most log entries of level ERROR or higher include an alphanumeric **message code** that uniquely identifies the log message (either "HSxxxxxx" or "DCxxxxxx" or "RMxxxxxx"). For documentation of an individual message code, click the message code text (in blue font) in the **Alerts** interface. This opens the log message code Help. The documentation for the message code that you clicked will be expanded, but you may need to scroll down the Help page to see it. (Alternatively, simply holding your cursor over the message code in the **Alerts** interface will display in-place "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.

**Note** Optionally you can suppress alerting for specific log messages as identified by message code. HyperStore provides a configuration setting for suppressing specific log-based alerts. In the configuration file [common.csv](#), see the setting **"alert\_suppression\_list"** (page 543).

#### *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 *hsstool opstatus* or check the CMC's **"Repair Status"** (page 380) 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.

### 5.8.1.2. 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 `mts.properties.erb:"events.acknowledged.ttl"` (page 568) 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.

## 5.8.2. Alert Rules

Path: **Alerts** → **Alert Rules**

Supported tasks:

- **"Review Supported Rule Types and Pre-Configured Rules"** (page 391)
- **"Add Alert Rules"** (page 395)
- **"Edit Alert Rules"** (page 397)
- **"Disable Alert Rules"** (page 397)
- **"Delete Alert Rules"** (page 398)
- **"Suppress Alerting for Specific Log Messages"** (page 398)

### 5.8.2.1. 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. 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.

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).

## 5.8.2.1.1. Network Status Rules

Rule Type	Description	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.	No
Number of PUT transactions per second	Same as above, except for S3 PUTs.	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.	No
Throughput for PUT operations	Same as above, except for S3 PUTs.	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.	No
Latency for PUT operations	Same as above, except for S3 PUTs.	No
Network throughput (incoming)	For each node, each minute the system calculates the 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.  <b>Note</b> Network throughput statistics are based on <b>all</b> 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.	No
Network throughput (outgoing)	Same as above, except for outgoing throughput.	No

## 5.8.2.1.2. General Status Rules

Rule Type	Description	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.	Yes -- a High severity alert if disk space

Rule Type	Description	Pre-Configured?
	<p><b>Note</b> 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 "<b>Capacity Managed</b>" (page 198).</p>	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.	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.	Yes -- a Critical severity alert
Node unreachable	With this alert rule, an alert is triggered if a node cannot be reached by the HyperStore monitoring system.	Yes -- a Critical severity alert
Load average (5 minutes)	<p>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.</p> <p><b>Note</b> 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.</p>	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.	Yes -- a Medium severity alert if CPU utilization is greater than 90%
Repair completion status	<p>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.</p> <p><b>Note</b> The completion of routine "proactive" repairs does not trigger these alerts. For more information on repair</p>	Yes -- a Low severity alert

Rule Type	Description	Pre-Configured?
	types see <b>"Automated Data Repair Feature Overview"</b> (page 150).	

### 5.8.2.1.3. Service Status Rules

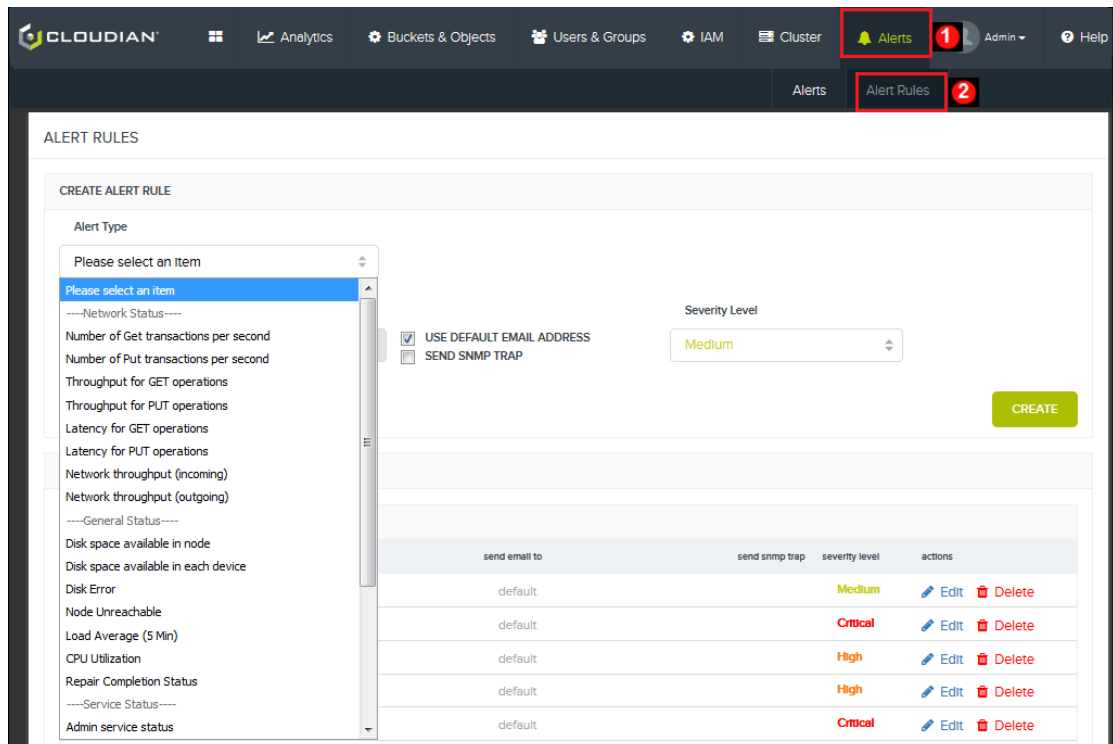
Rule Type	Description	Pre-Configured?
Admin Service status	For the <a href="#">Admin Service</a> , separate alert rules can be set for any of these statuses: <ul style="list-style-type: none"> <li>• Service is down</li> <li>• Service goes back up after being down</li> <li>• Service logs an error</li> <li>• Service logs a warning</li> </ul>	Yes -- a High severity alert if the service is down, and a High severity alert if the service logs an error
Cassandra Service status	For the <a href="#">Cassandra Service</a> , separate alert rules can be set for any of these statuses: <ul style="list-style-type: none"> <li>• Service is down</li> <li>• Service goes back up after being down</li> <li>• Service logs an error</li> <li>• Service logs a warning</li> </ul>	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 <a href="#">HyperStore Service</a> , separate alert rules can be set for any of these statuses: <ul style="list-style-type: none"> <li>• Service is down</li> <li>• Service goes back up after being down</li> <li>• Service logs an error</li> <li>• Service logs a warning</li> </ul>	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 <a href="#">Redis QoS Service</a> , separate alert rules can be set for any of these statuses: <ul style="list-style-type: none"> <li>• Service is down</li> <li>• Service goes back up after being down</li> <li>• Service logs a warning</li> </ul>	Yes -- a High severity alert if the service is down, and a Medium severity alert if the service logs a warning
Redis Credentials Service	For the <a href="#">Redis Credentials Service</a> , separate alert rules can be	Yes -- a

Rule Type	Description	Pre-Configured?
status	set for any of these statuses: <ul style="list-style-type: none"> <li>• Service is down</li> <li>• Service goes back up after being down</li> <li>• Service logs a warning</li> </ul>	High severity alert if the service is down, and a Medium severity alert if the service logs a warning
Redis Monitor Service status	For the <a href="#">Redis Monitor Service</a> , separate alert rules can be set for any of these statuses: <ul style="list-style-type: none"> <li>• Service is down</li> <li>• Service goes back up after being down</li> </ul>	Yes -- a High severity alert if the service is down
S3 Service status	For the <a href="#">S3 Service</a> , separate alert rules can be set for any of these statuses: <ul style="list-style-type: none"> <li>• Service is down</li> <li>• Service goes back up after being down</li> <li>• Service logs an error</li> <li>• Service logs a warning</li> </ul> <div> <b>Note</b> 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.         </div>	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 <a href="#">Cron and Monitoring services</a> , separate alert rules can be set for either of these statuses: <ul style="list-style-type: none"> <li>• Service logs an error</li> <li>• Service logs a warning</li> </ul>	Yes -- a High severity alert if the service logs an error
Phone Home Service status	For the <a href="#">Phone Home (Smart Support) service</a> , separate alert rules can be set for either of these statuses: <ul style="list-style-type: none"> <li>• Service logs an error</li> <li>• Service logs a warning</li> </ul>	Yes -- a High severity alert if the service logs an error

### 5.8.2.2. Add Alert Rules

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.



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 391).
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 338) 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.

**Note** Trap destination settings are configured in the **"SNMP Trap Destination Settings"** (page 341) section of the CMC's **Configuration Settings** page. 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. Traps are sent by the [HyperStore System Monitoring / Cron Job host](#).

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.

### 5.8.2.3. Edit Alert Rules

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 email 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.

### 5.8.2.4. Disable Alert Rules

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.

### 5.8.2.5. Delete Alert Rules

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.

### 5.8.2.6. Suppress Alerting for Specific Log Messages

Optionally **you can suppress alerting for specific log messages** as identified by message code. HyperStore provides a configuration setting for suppressing specific log-based alerts. In the configuration file [common.csv](#), see the setting "**alert\_suppression\_list**" (page 543).

#### See Also:

- "**How HyperStore Implements Alerts**" (page 398)

## 5.8.3. 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 341) 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 390)). 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.

### 5.8.3.1. 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.

## 5.9. 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](#)

### 5.9.1. 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: <input type="text" value="System Admin"/>	EMAIL ADDRESS: <input type="text"/>	PHONE NUMBER: <input type="text"/>
ADDRESS LINE 1: <input type="text"/>	ADDRESS LINE 2: <input type="text"/>	
CITY: <input type="text"/>	STATE, PROVINCE, REGION: <input type="text"/>	ZIP OR POSTAL CODE: <input type="text"/>
COUNTRY: <input type="text"/>	WEB SITE URL: <input type="text"/>	

**SAVE**

Supported task:

- Change your contact information

#### 5.9.1.0.1. 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"** (page 340) setting on the **Configuration Settings** page.

### 5.9.2. Security Credentials

Path: Drop-down menu under your user name (at top right) → **Security Credentials**

**SIGN-IN CREDENTIALS**

USER ID: admin

CURRENT PASSWORD:

NEW PASSWORD:

CONFIRM PASSWORD:

**CHANGE PASSWORD**

**S3 ACCESS CREDENTIALS**

CREATED	ACCESS KEY ID	ACTIONS
Dec-26-2019 11:18 -0800	6c2d83059ea622bdde7b *	View Secret Key Inactivate Delete

\* Active Access Key

**CREATE NEW KEY**

Supported task:

- Change your Console password
- Manage your S3 access keys

### 5.9.2.1. Change Your Console Password

**Note** If the system is configured to use LDAP authentication for your user group, the CMC's **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 [common.csv](#), see **"user\_password\_min\_length"** (page 527) and the subsequent settings.

3. Click **Change Password**.

### 5.9.2.2. 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 **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).)

- 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.

## 5.10. Customizing the CMC

### 5.10.1. 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, or whether the function is by default hidden from all user types. The fourth column indicates the configuration setting that controls access to the functionality — all settings are in *mts-ui.properties.erb* on your Puppet master node unless otherwise noted.

**Note** After making changes to a configuration file, use the installer to push the changes out to your cluster and restart the CMC service. For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

Function Area	Functionality	Default Availability	Controlling Setting
Manage Users	Basic user management interface	System admins and group admins	<b>"admin.manage_users.enabled"</b> (page 584)
	Create users	System admins	<b>"admin.manage_users.create.enabled"</b> (page

Function Area	Functionality	Default Availability	Controlling Setting
		and group admins	585)
	Edit users	System admins and group admins	" <b>admin.manage_users.edit.enabled</b> " (page 585)
	Delete users	System admins and group admins	" <b>admin.manage_users.delete.enabled</b> " (page 585)
	View and manage users' security credentials	System admins and group admins	" <b>admin.manage_users.edit.user_credentials.enabled</b> " (page 586)
	View and manage users' stored data	System admins and group admins	" <b>admin.manage_users.viewuserdata.enabled</b> " (page 586)
	Edit users' Quality of Service settings	System admins and group admins	" <b>admin.manage_users.edit.user_qos.enabled</b> " (page 586)
Manage Groups	Basic group management interface	System admins and group admins	" <b>admin.manage_groups.enabled</b> " (page 587)
	Create groups	System admins	" <b>admin.manage_groups.create.enabled</b> " (page 587)
	Edit groups	System admins and group admins	" <b>admin.manage_groups.edit.enabled</b> " (page 587)
	Delete groups	System admins	" <b>admin.manage_groups.delete.enabled</b> " (page 588)
	Set default user QoS for a group	System admins and group admins	" <b>admin.manage_groups.user_qos_groups_default.enabled</b> " (page 588)
Billing whitelist	Set source IP addresses allowed free traffic	Hidden	" <b>admin_whitelist_enabled</b> " (page 537) in <i>common.csv</i>
User Account Self-Management	Edit own profile	System admins, group admins, and regular users	" <b>account.profile.writeable.enabled</b> " (page 588)
	Basic security credentials interface	System admins, group admins, and regular users	" <b>account.credentials.enabled</b> " (page 589)
	Manage own S3 credentials	Group admins and regular users	" <b>account.credentials.access.enabled</b> " (page 589)
	Change own CMC password	System admins, group admins, and regular users	" <b>account.credentials.signin.enabled</b> " (page 589)
Usage Reporting	Basic usage reporting interface	System admins, group admins, and regular users	" <b>usage.enabled</b> " (page 590)
	Reporting on HTTP request rates and byte transfer rates	Hidden	" <b>Track/Report Usage for Request Rates and Data Transfer Rates</b> " (page 344) in the CMC Configuration Settings page

Function Area	Functionality	Default Availability	Controlling Setting
Auto-Tiering	Allow buckets to be configured for auto-tiering	Hidden	<b>"Enable Auto-Tiering"</b> (page 347) in the CMC <b>Configuration Settings</b> page

### 5.10.2. 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")

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 Puppet master node).

To rebrand the CMC UI, follow these steps:

1. Log into the Puppet master 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 Puppet master 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* script, run the script simply as *rebrand\_cmc.sh <command>* without specifying a path to the script.

2. On the Puppet master node create or choose a working directory from which you will manage the process of rebranding the CMC UI. Then change into the working directory.
3. Run the following script command to back up the current CMC files that are relevant to the UI's branding.

```
/opt/cloudian/tools/rebrand_cmc.sh --backup
```

This action creates a *web\_backup\_<timestamp>* sub-directory under your working directory on the Puppet master node. In the backup directory are files copied from the CMC's Tomcat web server configuration, that affect the CMC's look and feel.

4. Make your desired changes to the CMC's branding

*Replace logos*

- a. Copy your organization's logo image files -- as specified in the introduction to this procedure -- from your local machine to the working directory on the Puppet master node (by using `scp`, for example).
- b. Change into working directory on the Puppet master node, if you are not already there. Then run this script command:

```
/opt/cloudian/tools/rebrand_cmc.sh --images
```

This copies the image files from the working directory to the proper location within the Puppet configuration module for the CMC, so that you can subsequently push the files out to the whole cluster (as described in Step 6).

#### *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. On the Puppet master node, copy the `resources.properties` file (for English language) and any of the `resources_<language-code>.properties` files (for other languages that the CMC supports, if applicable to your user population) from the backup sub-directory that you created in Step 3 into the working directory.
- b. Use a text editor to edit the copy of `resources.properties` in the working directory, as follows:
  - i. In the file find the following line:

```
header.title=Cloudian®; 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 that you've copied into the working directory (if any).
- d. After editing the file(s) and saving your changes, while still in the working directory run this script command:

```
/opt/cloudian/tools/rebrand_cmc.sh --resources
```

This copies the resource file(s) from the working directory to the proper location within the Puppet configuration module for the CMC, so that you can subsequently push the file(s) out to the whole cluster (as described in Step 6).

#### *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. On the Puppet master node, copy the `master.css` file from the backup sub-directory that you created in Step 3 into the working directory.
- b. Use a text editor to edit the copy of `master.css` file in the working directory, to replace all occurrences of the existing color codes with the desired new values.
- c. After editing the file and saving your changes, while still in the working directory run this script command:

```
/opt/cloudian/tools/rebrand_cmc.sh --css
```

This copies the CSS file from the working directory to the proper location within the Puppet configuration module for the CMC, so that you can subsequently push the file out to the whole cluster (as described in Step 6).

*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, do the following:

- a. On the Puppet master node, use a text editor to open the configuration file `/etc/cloudian-7.2.3-puppet/manifests/extdata/common.csv` and edit this setting:

```
cmc_application_name,Cloudian
```

Replace *Cloudian* with a different application name string of your choosing. Use only alphanumeric characters, with no spaces, dashes, or underscores.

- b. Save the updated *common.csv* file.
5. To confirm that the rebranding script has successfully copied your customized image, resource, and/or CSS files to the Puppet configuration module, run this script command:

```
/opt/cloudian/tools/rebrand_cmc.sh --list
```

This should list all the image, resource, and/or CSS files that you worked with in Step 4. (It will not list the *common.csv* file.)

6. Use the installer to push your customized file to the cluster and to restart the CMC to apply the changes. If you need instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

**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.

**Note** The *rebrand\_cmc.sh* script also supports adding a custom banner to the top of the CMC login page. For instructions see **"Configuring a Login Page Banner"** (page 407).

### 5.10.2.1. 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 Cloudian logo. For each Help system the logo image file is named *CloudianLogoFull\_1.png* and its size is 235 X 54 pixels.

To replace the Cloudian 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/apache-tomcat-7.0.85/webapps/Cloudian/help/HyperStoreHelp/Skins/Default/Stylesheets/Images/CloudianLogoFull_1.png`

- /opt/cloudian-packages/apache-tomcat-7.0.85/webapps/Cloudian/help/HyperStoreHelpGroupAdmin/Skins/Default/Stylesheets/Images/CloudianLogoFull\_1.png
- /opt/cloudian-packages/apache-tomcat-7.0.85/webapps/Cloudian/help/HyperStoreHelpEndUser/Skins/Default/Stylesheets/Images/CloudianLogoFull\_1.png

**Note** Changing this image file is not supported by the `rebrand_cmc.sh` script, and this image file is not under Puppet control. To replace this image file you must do it manually on each of your HyperStore nodes.

### 5.10.3. Configuring a Login Page Banner

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 a login page banner; for instructions for configuring an acknowledgment gate see **"Configuring a Login Page Acknowledgment Gate"** (page 408).

Here is an example in which a custom text banner has been added to the login page.

To configure a custom banner for the CMC login page:

1. Log in to the Puppet master 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 Puppet master 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* script, run the script simply as *rebrand\_cmc.sh <command>* without specifying a path to the script.

2. Create a working directory, and change into that directory.
3. Run the following command:

```
/opt/cloudian/tools/rebrand_cmc.sh --backup
```

This creates under your working directory a sub-directory named as *web\_backup\_<timestamp>*.

4. Copy the *custom\_banner.jsp* file from the backup sub-directory into the working directory.
5. In the working directory, use a text editor such as *vi* to make the following edits to the *custom\_banner.jsp* file:

- a. Uncomment the starting *style* tag.

Default in *custom\_banner.jsp* file (with starting *style* tag commented out):

```
<!-- style>
```

After uncommenting:

```
<style>
```

- b. Replace *Title* with your desired banner title text. Do not alter the *h2* tags.

```
<h2>Title</h2>
```

- c. Replace *Message* with your desired banner body text. Do not alter the *p* tags.

```
<p>Message</p>
```

6. After saving your changes and exiting the file, while still in the working directory run the following command:

```
/opt/cloudian/tools/rebrand_cmc.sh --custom_banner
```

This copies your edited *custom\_banner.jsp* file to the appropriate Puppet configuration directory.

7. Use the installer to push your changes out to the cluster and then restart the CMC. If you need instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

To verify that the banner is displaying as desired, go to the CMC in your browser.

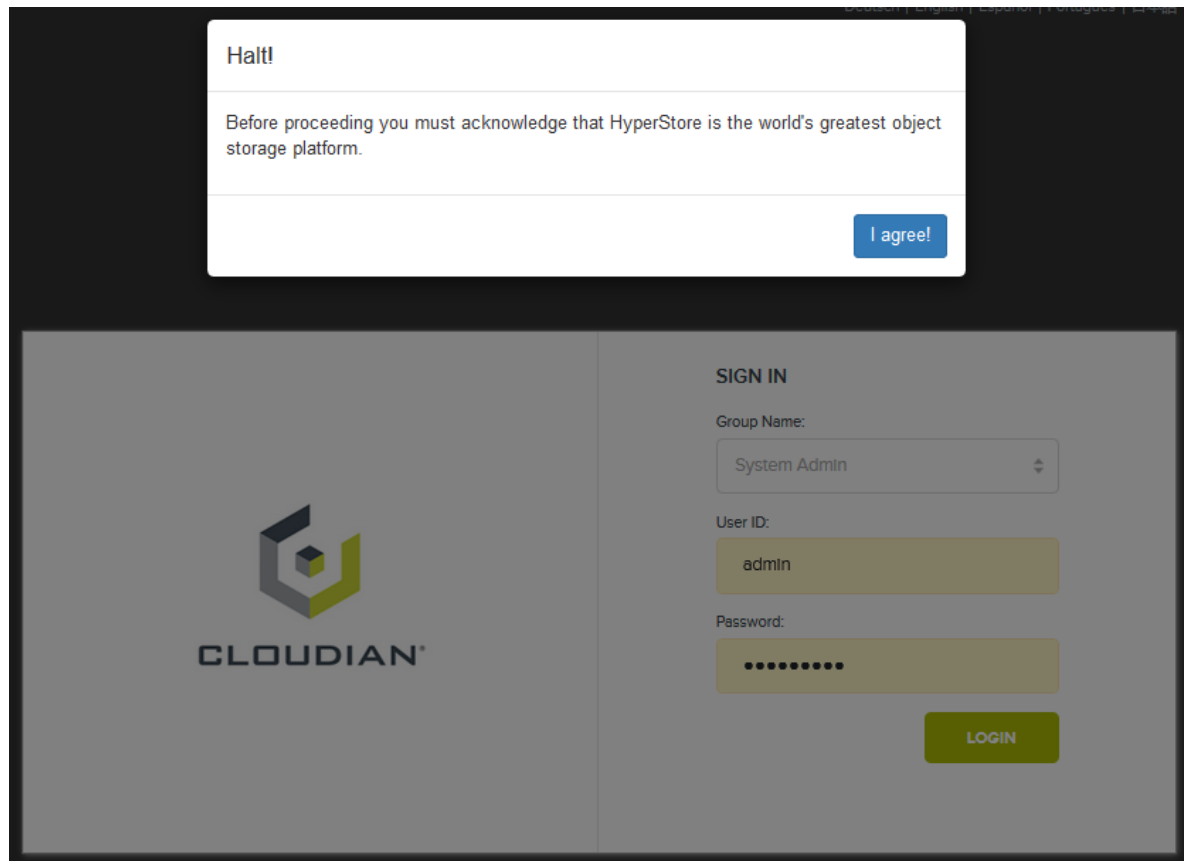
### 5.10.4. Configuring a Login Page Acknowledgment Gate

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 **"Configuring a Login Page Banner"** (page 407).

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:

1. Log into the Puppet master 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 Puppet master node to perform this procedure, so long as you are an HSH [Trusted user](#). To open the *common.csv* file for editing as is required in this procedure you can use the command `hspkg config -e common.csv`. This command opens the file with the *vi* text editor.

2. Open the configuration file [common.csv](#) in a text editor.
3. Edit the following settings as desired:

**Note** You do not need to enclose any of the setting text in quotes.

- *cmc\_login\_banner\_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\_banner\_title* -- Title text of the acknowledgment dialog.

- *cmc\_login\_banner\_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\_banner\_button\_confirm* -- 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.
4. After saving your changes and exiting the file, push your changes to the cluster and then restart the CMC. If you need instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

To verify that the acknowledgment gate is working as desired, go to the CMC in your browser.

### 5.10.5. 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](#) but the implementation of user mapping is left to the portal application integrating with CMC.

#### 5.10.5.1. 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.

**IMPORTANT !** To use the Cloudian HyperStore SSO feature, the following system configuration settings must be set to "true":

-- **"cmc\_web\_secure"** (page 538) in *common.csv* -- This is set to true by default. Leave it true if you want to use SSO.

-- **"sso.enabled"** (page 592) in *mts-ui.properties.erb* -- This is set to false by default. Change it to true if you want to use SSO.

Also in *mts-ui.properties*, if you enable SSO functionality (by setting *sso.enabled* to true), then for security reasons you should set *sso.shared.key* and *sso.cookie.cipher.key* to custom values. **Do not use the default keys.**

### 5.10.5.2. 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/ssosecurelogin.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 configurable request expiration (*mts-ui.properties: sso.tolerance.millis*; expiration defaults to one hour).
- *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.

#### 5.10.5.2.1. 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 *mts-ui.properties: 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 413).

#### 5.10.5.2.2. 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>
```

#### 5.10.5.2.3. 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)

#### 5.10.5.2.4. 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.

#### 5.10.5.2.5. 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.

#### 5.10.5.2.6. 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.

### 5.10.5.3. 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?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.

# Chapter 6. Node and Cluster Operations

## 6.1. Starting and Stopping Services

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Start or Stop Services on All Nodes in the Cluster"** (page 415)
- **"Start or Stop Services on One Node"** (page 417)
- **"Shutting Down or Rebooting a Node"** (page 419)
- **"Automatic Service Start on Node Boot-Up"** (page 419)

You can start, restart, or stop a service across all the nodes in your cluster by using the HyperStore installer. If instead you want to start, restart, or stop a service on just one particular node, you can do so through the CMC or by using a HyperStore service initialization script. Note also that HyperStore services are configured to start automatically when you reboot a node.

### 6.1.1. Start or Stop Services on All Nodes in the Cluster

The interactive tool that you used to install the HyperStore system — *cloudianInstall.sh* — can also be used to manage HyperStore services. The tool enables you to start, restart, or start one or more services **on all nodes at once** (or to put it more precisely, on all nodes in rapid succession).

**Note** The installation tool does not support managing a service on just one particular node.

1. On your **Puppet master node**, change into the [installation staging directory](#) and then launch the HyperStore installer.

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

This displays the top-level menu of options.

## Cloudian HyperStore(R) 7.2 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: █

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

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Puppet 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.

2. Choose "Cluster Management", then from the sub-menu that displays choose "Manage Services". This displays the "Service Management" sub-menu:

### Service Management

---

- 0 ) All services
- 1 ) Redis Credentials
- 2 ) Redis QOS
- 3 ) Cassandra
- 4 ) HyperStore service
- 5 ) S3 service
- 6 ) Redis Monitor
- 7 ) Cloudian Agent
- 8 ) DNSMASQ
- 9 ) Cloudian Management Console (CMC)
- 10) IAM
- 11) SQS
- P ) Puppet service (status only)
- X ) Quit

You can execute the following list of commands:

start, stop, status, restart, version, node-start, node-stop

Select a service to manage: █

**Note** For future reference: As an alternative to launching *cloudianInstall.sh* and navigating to the "Service Management" menu, you can access this menu directly by launching the *cloud-ianService.sh* script in your installation staging directory.

- a. At the prompt, enter a service number from the menu. To manage all services enter option (0).

**Note** The Admin Service is bundled with the S3 Service. Any operation that you apply to the S3 Service (such as stopping or restarting) applies also to the Admin Service. Likewise, the STS Service is bundled together with the IAM Service, so any operation that you apply to the IAM Service also applies to the STS Service.

- b. At the prompt that appears after you make your service selection, enter a service command: start, stop, status, restart, or version. (The "version" option is supported only for the S3 Service.)

The service command you enter will be applied to all nodes on which the service resides. For example, if you choose Cassandra and then enter "start", this will start Cassandra on all nodes on which it is installed. Likewise if you choose S3 and then "status", this will return the status of the S3 Service on each node on which it is installed. And if you choose "All services" and then "stop", this will stop all services on all nodes.

**Note** From the "Service Management" menu all you can do for the Puppet service is check its status. To stop or start the Puppet daemons, from the installer's main menu choose "Advanced Configuration Options". From the advanced sub-menu that displays you can stop or start the Puppet daemons.

### 6.1.2. Start or Stop Services on One Node

You can start, restart, or stop a service on just one particular node by using the CMC's [Node Status](#) page.

As an alternative to using the CMC for this task, you can use the following commands. These commands can be run from any directory on the target node.

*If you are using the HyperStore Shell*

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

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

Example:

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

Example:

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

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>

Service	<servicename>
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 <b>"HyperStore Support for the AWS SQS API"</b> (page 1041) for information on enabling it)
Cassandra Service	<i>cloudian-cassandra</i>
HyperStore Service	<i>cloudian-hyperstore</i>
Redis Credentials Service	<i>cloudian-redis-credentials</i>
Redis QoS Service	<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>
Dnsmasq	<i>cloudian-dnsmasq</i>

#### 6.1.2.1. Stop or Start **All** Services on One Node

To stop all of the services on a single node, stop each individual service (as described in **"Start or Stop Services on One Node"** (page 417)) in this order:

1. CMC
2. Cloudian Monitoring Agent
3. Redis Monitor
4. S3 Service
5. IAM Service
6. SQS Service (if being used)
7. HyperStore Service
8. Redis QoS
9. Redis Credentials
10. Cassandra
11. Dnsmasq

To start all of the services on a single node, start each individual service in this order:

1. Cassandra
2. Redis Credentials
3. Redis QoS
4. HyperStore Service
5. SQS Service (if being used)
6. IAM Service
7. S3 Service

8. Redis Monitor
9. Cloudbian Monitoring Agent
10. CMC
11. Dnsmasq

### 6.1.2.2. Checking the ISO Version on a HyperStore Appliance machine

On a HyperStore Appliance machine, you can check the ISO version (the version of the HyperStore ISO file from which CentOS was installed on the machine) by changing into the `/root/CloudianPackages` directory and then running the following command:

```
# cat ISOVERSION
```

Example:

```
# cat ISOVERSION
Cloudian CentOS 7.4 Rev d56af03 Custom - 09/27/17
```

### 6.1.3. 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.

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

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

To power off:

```
# systemctl poweroff
```

To reboot:

```
# systemctl reboot
```

### 6.1.4. Automatic Service Start on Node Boot-Up

By default all of the HyperStore services on a host are configured to automatically start when the host is booted up (the configuration setting is `common.csv: "service_starts_on_boot"` (page 515), which defaults to "true"). This includes `dnsmasq` if it was included in your HyperStore software installation.

**Note** `ntpd` is configured to automatically start on host boot-up. By design, the sequencing is such that `ntpd` starts up before major HyperStore services do.

Cloudian Inc. recommends that after booting a HyperStore host, you verify that `ntpd` is running. You can do this with the `ntpq -p` command. If `ntpd` is running this command will return a list of connected time servers.

**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.

## 6.2. 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
- Clean from the existing nodes data that they are no longer responsible for

**IMPORTANT !** Be sure about the node or nodes that you are adding to the cluster, before you add them to the cluster:

\* Once you have added a node to the cluster's Cassandra ring **you cannot simply "undo" the process**. If after adding a new node to the Cassandra ring you were to change your mind about keeping the node in the cluster, you would still be required to rebalance data within the cluster (so that data is streamed in to the new node) and then afterwards you would need to decommission the node (so that data is streamed away from the new node). That is the only way that the system will allow you to remove the node from the cluster.

\* Once you add a node to your cluster, **HyperStore does not support adding disks to the node**. Make sure that the node or nodes that you are adding have sufficient disk capacity to meet your needs.

### 6.2.1. 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 443) 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.

### 6.2.2. 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 OS Requirements" in the HyperStore installation Guide
  - Open listening ports: See "HyperStore Listening Ports" in the Reference section of the *HyperStore Installation Guide*
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 Puppet master 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 data disk formatting for each new node.

*More detail*

- a. On your Puppet master node change into the [installation staging directory](#) and then launch the `system_setup.sh` tool.

```
# ./system_setup.sh
```

*If you are using the HyperStore Shell on the Puppet master node*

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Puppet master 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.
  - 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 Puppet master node you must for each new node create a text file named `<hostname>_fstlist.txt` that specifies the new node's data mount points, in this format:

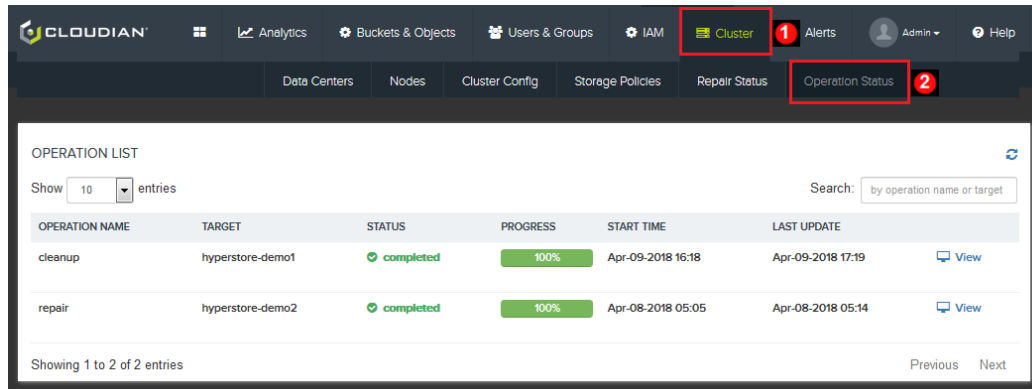
```
<devicename> <mountpoint>
<devicename> <mountpoint>
etc...
```

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*, *repaiREC*, *cleanup*, or *cleanupEC* 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.
- It's inconsequential to allow the system to stop an in-progress cleanup operation on a node, because you will need to run a fresh cleanup operation on each existing node anyway, at the end of the Adding Nodes procedure.

**Note** When you initiate the **Add Node** operation in the CMC, the system will also automatically disable the auto-repair feature and the proactive repair feature -- so that no new repairs kick off while you're expanding your cluster -- and then after you've completed the rebalance operation the system automatically re-enables auto-repair and proactive repair.

- In the **Data Centers** page, make sure that all the existing nodes and services are up and running in the service region.

*More detail*

**Node Status:** ● Unreachable ● Has Disk Error ● Disk Above 80% Full ▲ Has Alerts ● Under Maintenance ● All Clear

DC1 DC2

RAC1 RAC1

5 node(s) 5 node(s)

+ NEW DC

**SERVICE STATUS**

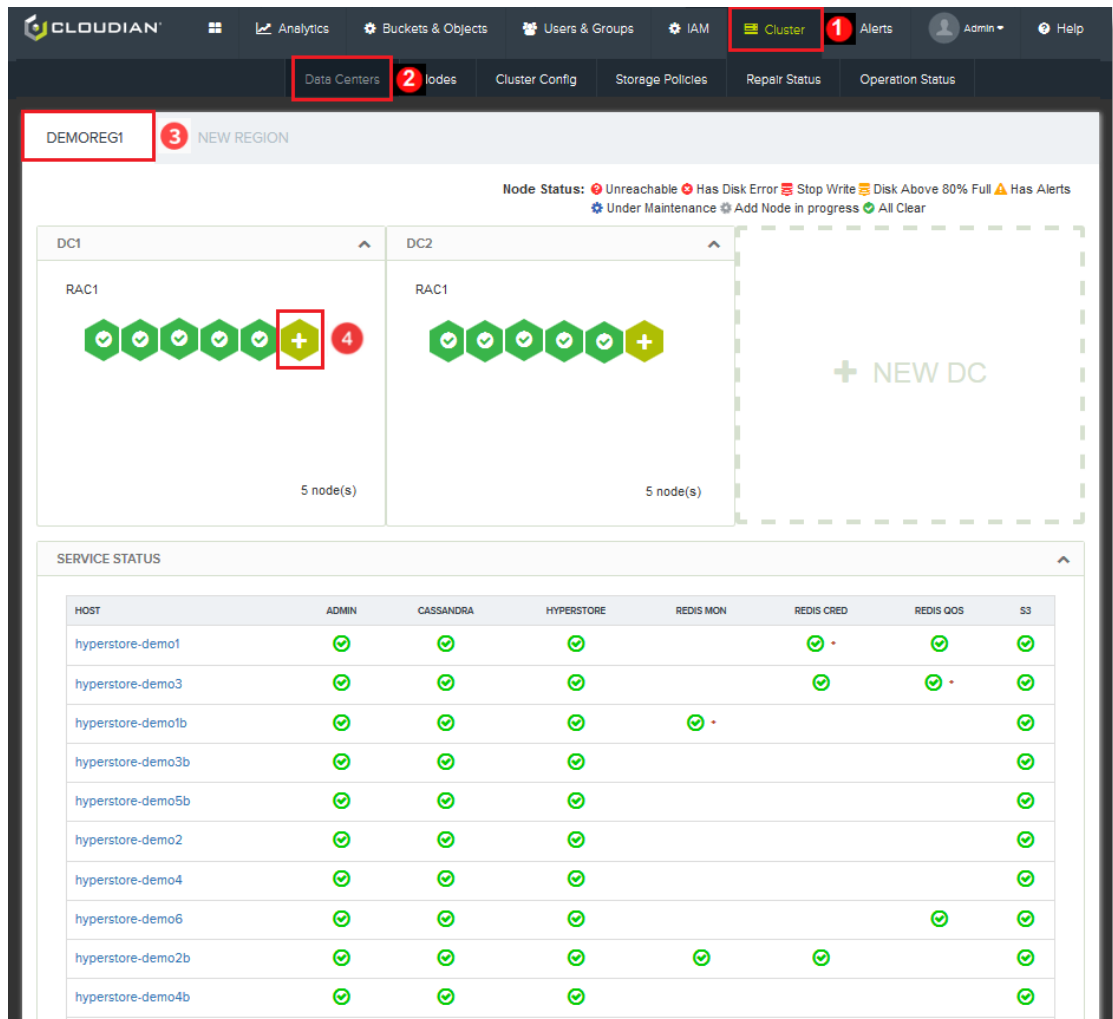
HOST	ADMIN	CASSANDRA	HYPERSTORE	REDIS MON	REDIS CRED	REDIS QOS	S3
hyperstore-demo1	✓	✓	✓		✓ -	✓	✓
hyperstore-demo3	✓	✓	✓		✓	✓ -	✓
hyperstore-demo1b	✓	✓	✓	✓ -			✓
hyperstore-demo3b	✓	✓	✓				✓
hyperstore-demo5b	✓	✓	✓				✓
hyperstore-demo2	✓	✓	✓				✓
hyperstore-demo4	✓	✓	✓				✓
hyperstore-demo6	✓	✓	✓			✓	✓
hyperstore-demo2b	✓	✓	✓	✓	✓		✓
hyperstore-demo4b	✓	✓	✓				✓

The **Add Node** function will not let you add nodes if any existing nodes or services in the region are down.

### 6.2.3. 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, click the light green cube icon that has a plus sign on it.

*More detail*



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.

2. In the **Add Node** interface that displays, complete the fields for the new node.

*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

Description: Add a new node to the cluster. If you are adding multiple nodes, add them one at a time here, waiting for the Add Node operation to successfully complete for each node before you add the next node. 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; 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 (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 do it in such a way 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 Puppet master 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 Puppet master, 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**. This initiates a background operation that will take anywhere from a half-hour up to a full day to complete depending on your environment. When it completes successfully an orange node icon with a gear inside of it displays 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 Cassandra).

There are two CMC locations where you can monitor the **Add Node** operation progress:

- The **Data Centers** page. A new node icon appears, with color-coding to indicate the status of the **Add Node** operation. You can hold your cursor over the node icon for a text description of the status.

**Note** The new node status icon will not display until the system successfully completes preliminary actions such as verifying that the new node meets HyperStore requirements and adding the node to the Cassandra ring. Prior to the appearance of the grey new node icon, if the **Add Node** operation encounters errors the system will automatically roll back the operation. Once the grey new node icon appears, the node has been added to the Cassandra ring and the operation can no longer be rolled back.



Grey node with gear -- The node has been added to the Cassandra ring, and Cassandra repair (which streams metadata to the new node) is in progress.



Orange node with gear -- Cassandra repair has completed successfully, and the node requires that you run a "rebalance" operation to stream object data to it (as described later in this procedure).

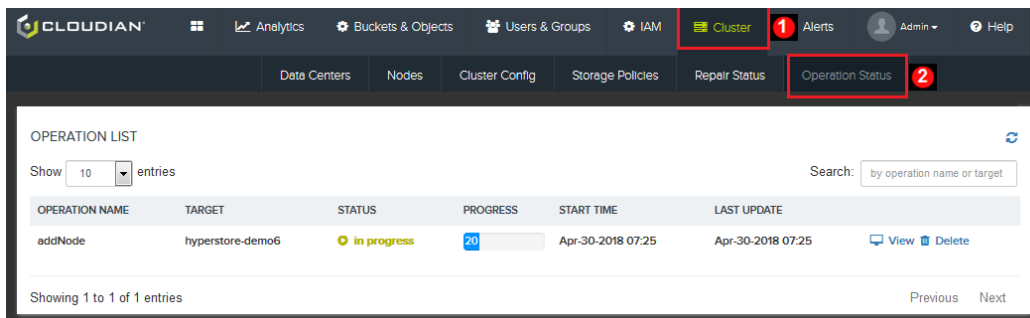


Red node with gear -- Cassandra repair has failed. If this status occurs, first check the **Data Center** page's **Service Status** section to make sure that Cassandra is up and running on all nodes (if it's down on any node, go to the **Node Status** page for that node and start Cassandra). After making 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.

**Note** Do not try to add any more new nodes until the **Data Centers** page shows the orange icon for the currently added node.

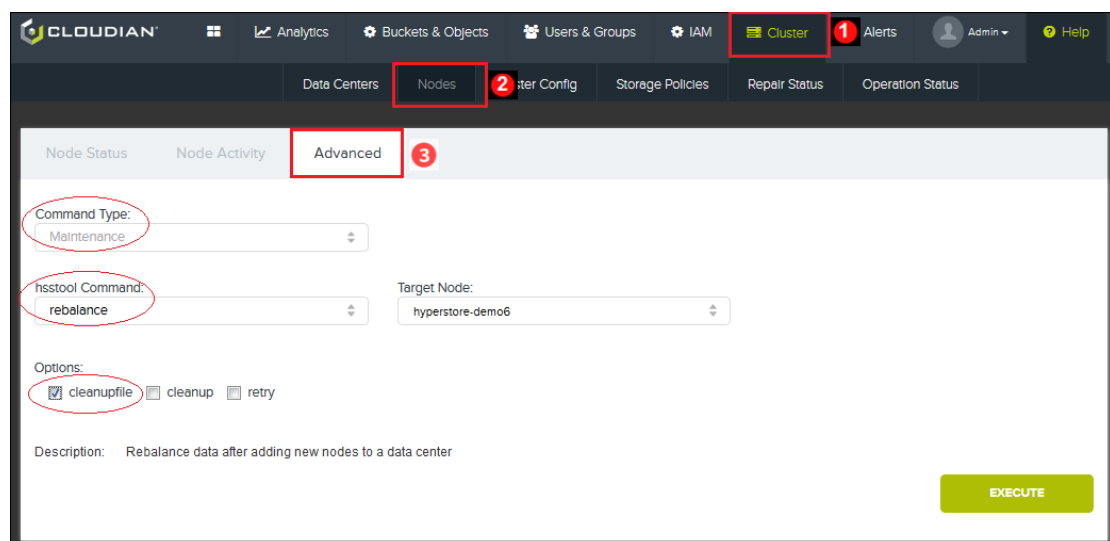
- The **Operation Status** page. Click **View** to the right of the summary status line to display detailed progress information as the operation proceeds through pre-installation checks and initial set-up

of the node.



4. When the **Data Centers** page shows the orange icon for the currently added node, you can add more nodes (if you wish) by repeating Steps 1, 2, and 3 above for each new node. If you have a multi-DC cluster, be careful to click the add node icon within the data center panel for your desired DC. You can add different nodes to different DCs if you wish, as long as they're in the same service region.
5. After the **Add Node** operation has completed successfully for each new node, **update your DNS and/or load balancer** configurations to include the new node(s), so that the new node(s) can participate in servicing user request traffic (if you have not already done so). For more information see "DNS Set-Up" and "Load Balancing" in the *HyperStore Installation Guide*.
6. If your system is currently using HyperStore software that you have previously "patched" by running the HyperStore patch installer, log into the Puppet master node and manually run the patch installer again to apply the patch to the new nodes. For more information on this task see "**Adding Nodes to a Patched System**" (page 64). After the patch installation completes successfully, proceed to Step 7. (If your current system is not a patched version of HyperStore you can skip Step 6 and go directly to Step 7.)
7. In the **Node Advanced** page, from the Maintenance command type group, **execute *hsstool rebalance* on each new node**. 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. When all new nodes have completed rebalancing, for each node a green, check-marked cube icon will display in the **Data Centers** page.

More detail



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 to complete, depending on factors such as data volume and network bandwidth. When rebalance is performed with the **cleanup-file** 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 Parameters**" (page 682).

**Note** During in-progress rebalance operations the affected data remains readable by S3 client applications. Meanwhile the new node(s) 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 the new node(s) will be color-coded, and you can hold your cursor over the icon(s) 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, go to the **Nodes Advanced** page and run rebalance on the node again, using the "retry" option this time (select the retry checkbox when you run rebalance on the node), as well as the "cleanupfile" option. This will try the rebalance again, just for the failed token range(s).



Green node with check mark -- The rebalance operation has completed successfully. If you have added multiple nodes, this status will not display for any of the new nodes until rebalance has completed on all of the new nodes.

Another source of status information for the rebalance operation is the **Operation Status** page.

The screenshot shows the Cloudian CMC interface. In the top navigation bar, the 'Cluster' tab is highlighted with a red box and a red '1'. Below it, the 'Operation Status' sub-tab is also highlighted with a red box and a red '2'. The main content area shows the 'OPERATION LIST' with a search bar and a table of operations.

OPERATION NAME	TARGET	STATUS	PROGRESS	START TIME	LAST UPDATE	
rebalance	hyperstore-demo6	<span style="color: green;">in progress</span>	<div><div style="width: 20%;">20</div></div>	Apr-30-2018 08:27	Apr-30-2018 08:39	<a href="#">View</a> <a href="#">Delete</a>
addNode	hyperstore-demo6	<span style="color: green;">completed</span>	<div><div style="width: 100%;">100%</div></div>	Apr-30-2018 07:25	Apr-30-2018 07:39	<a href="#">View</a> <a href="#">Delete</a>

Showing 1 to 2 of 2 entries

Previous Next

For status detail click **View** to the right of the summary status line.

8. 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.
9. After all rebalance operations have completed (and repair operations have completed if Step 8 was applicable to you), run `hsstool cleanup allkeyspaces` on each node except the new node(s) -- one node at a time, waiting for the cleanup to complete before you start the cleanup on the next node. This cleans up any metadata replicas (in Cassandra) that no longer belong on the older nodes due to the addition of the new node(s).

This completes the procedure for adding nodes to your cluster.

## 6.3. 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

**IMPORTANT !** Once you add nodes to your cluster, HyperStore does not support adding disks to those nodes. Make sure that the nodes that you are adding have sufficient disk capacity to meet your needs.

### 6.3.1. 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 443) 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.

### 6.3.2. 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 OS Requirements" in the HyperStore installation Guide
  - Open listening ports: See "HyperStore Listening Ports" in the Reference section of the *HyperStore Installation Guide*
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 Puppet master 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 data disk formatting for each new node.

*More detail*

- a. In your existing cluster, on your Puppet master node change into the [installation staging directory](#) and then launch the `system_setup.sh` tool.

```
# ./system_setup.sh
```

*If you are using the HyperStore Shell on the Puppet master node*

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Puppet master 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.
  - 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 Puppet master 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...
```

6. 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	✓	✓	✓				✓

### 6.3.3. Adding a Data Center

1. 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**

**RAC1** **RAC1**

5 node(s) 5 node(s)

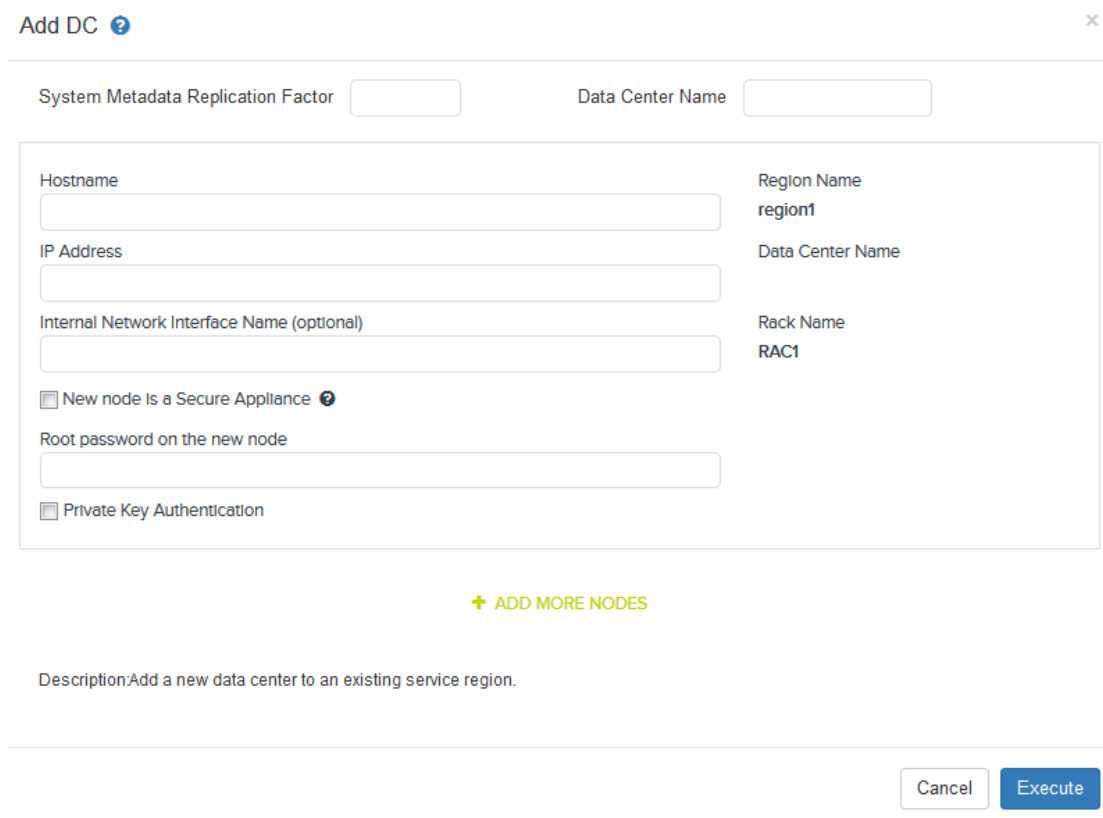
**+ NEW DC**

**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 Puppet master 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

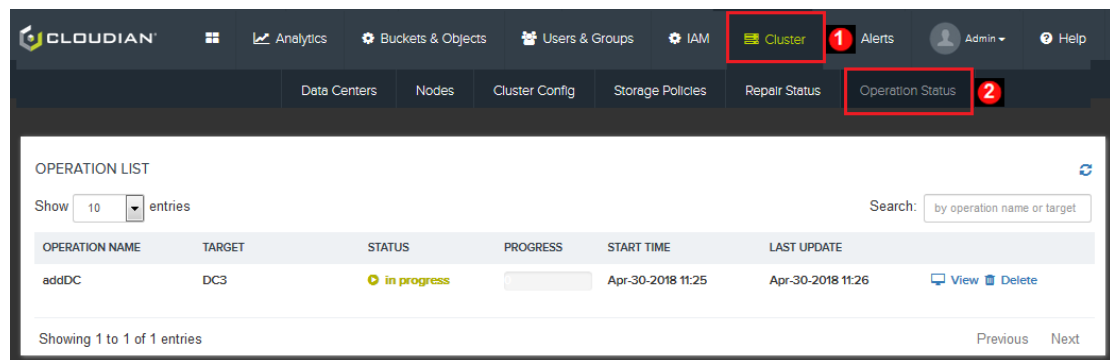
and the public key into the installation staging directory on the Puppet master, then distribution of the public key to the new node will be taken care of automatically by the installer.

- 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.



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 Puppet master 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" and "Load Balancing" in the *HyperStore Installation Guide*.
5. If your system is currently using HyperStore software that you have previously "patched" by running the HyperStore patch installer, log into the Puppet master node and manually run the patch installer again to apply the patch to the new nodes. For more information on this task see **"Adding Nodes to a Patched System"** (page 64). After the patch installation completes successfully, proceed to Step 6. (If

your current system is not a patched version of HyperStore you can skip Step 5 and go directly to Step 6.)

6. 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.

## 6.4. 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

**IMPORTANT !** Once you add nodes to your system, HyperStore does not support adding disks to those nodes. Make sure that the nodes that you are adding have sufficient disk capacity to meet your needs.

### 6.4.1. 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 OS Requirements" in the HyperStore installation Guide
  - Open listening ports: See "HyperStore Listening Ports" in the Reference section of the *HyperStore Installation Guide*
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 Puppet master 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 data disk formatting for each new node.

#### *More detail*

- a. In your existing cluster, on your Puppet master node change into the [installation staging directory](#) and then launch the *system\_setup.sh* tool.

```
# ./system_setup.sh
```

*If you are using the **HyperStore Shell** on the Puppet master node*

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Puppet master 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.
  - 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 Puppet master 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...
```

## 6.4.2. Adding a Region

1. In the CMC's **Data Centers** page, click the tab that says **+NEW REGION**.

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 'Data Centers' tab is selected (highlighted with a red box and '2'). The 'Data Centers' page shows a 'DEMOREG1' region with a '+ NEW REGION' button (highlighted with a red box and '3'). Below this, there are two data centers, DC1 and DC2, each with a RAC1 and 5 nodes. A 'NEW DC' button is visible. Below the data centers is a 'SERVICE STATUS' table.

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 Puppet master 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 Puppet master, then distribution of the public key to the new node will be taken care of

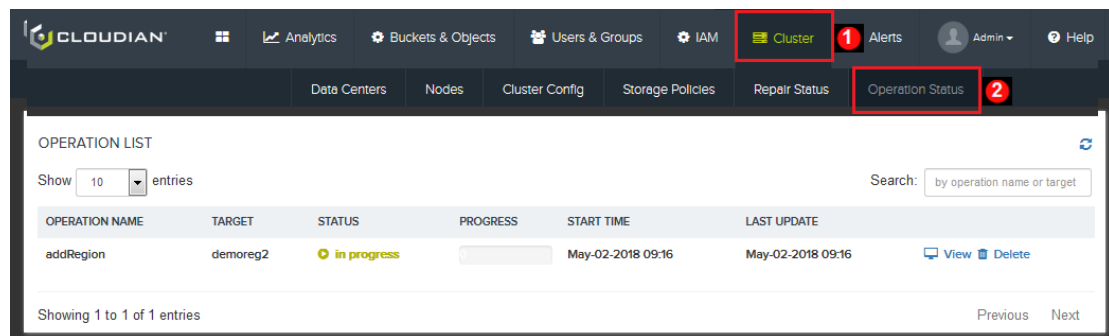
automatically by the installer.

- 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 Puppet master 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" and "Load Balancing" in the *HyperStore Installation Guide*.
5. If your system is currently using HyperStore software that you have previously "patched" by running the HyperStore patch installer, log into the Puppet master node and manually run the patch installer again to apply the patch to the new nodes. For more information on this task see **"Adding Nodes to a**

**Patched System"** (page 64). After the patch installation completes successfully, proceed to Step 6. (If your current system is not a patched version of HyperStore you can skip Step 5 and go directly to Step 6.)

6. 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.
7. 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) Controls"** (page 285). 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.

## 6.5. 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" -- Cassandra down or unreachable before removing 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 420).

**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 stored on that node is unavailable to client applications. Once data has been streamed from the decommissioning node to the other nodes that data is again available to support client requests. But depending on the data volume and network bandwidth, it may take up to several days or more until the decommissioning process has completed for all of the node's data. Until all of the decommissioning node's data has been streamed to other nodes, some objects will have fewer than your configured number of replicas or erasure coded fragments available within the live system.

### 6.5.1. 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 420) (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 Cloudian 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.

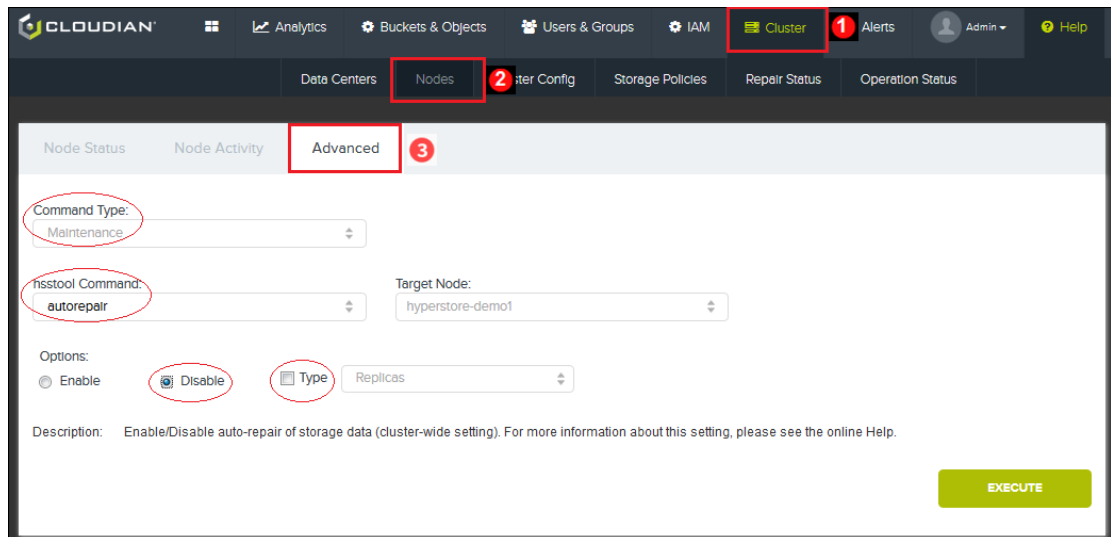
**Note** If any of the other nodes are in the ["stop-write" condition](#) -- or if any disks on a node are in stop-write condition -- at a time when you decommission a different node from your system, the decommissioning process overrides the stop-write restriction on the node(s) or disk(s) where it 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 Cloudian 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*

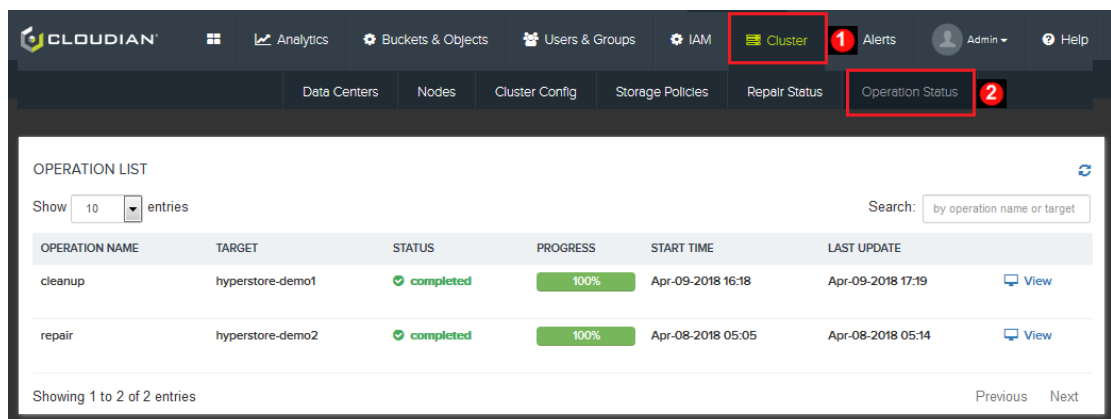
The target node for the command can be any node in the region. Leave the "Type" option unselected so that all automated repair types are disabled. This will prevent any new schedule-based auto-repairs or proactive repairs from launching during the node removal process.



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.



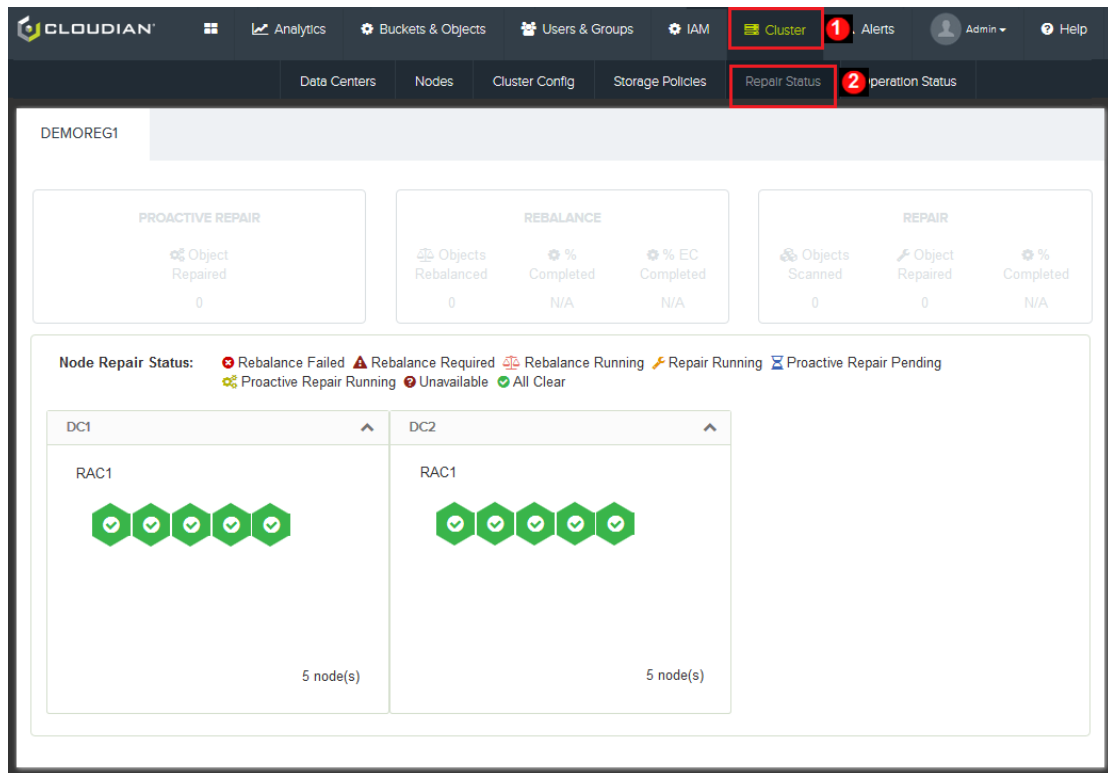
**Note** If you don't want to wait for an in-progress repair or cleanup of a node to complete you have the option of terminating the operation. To do so, go to the **Node Advanced** page and for that node execute [hsstool repair](#) or [hsstool repairec](#) or [hsstool cleanup](#) or [hsstool cleanupec](#) 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 proactiverepair](#) with the "stop" option selected.

6. If the node you want to remove is the active Puppet Master node, follow the instructions to **"Manually Fail Over the Puppet Master Role from the Primary to the Backup"** (page 469).

#### More detail

If you're not sure whether the node you want to remove is the Puppet Master host, check the CMC's **Cluster Information** page. That page also shows which host is the Puppet Master backup host.

The screenshot shows the Cloudian management console interface. The top navigation bar includes tabs for Analytics, Buckets & Objects, Users & Groups, IAM, Cluster (selected), Alerts (1), Admin, and Help. Below this, a secondary navigation bar shows Data Centers, Nodes, Cluster Config (selected), Storage Policies, Repair Status, and Operation Status. The main content area is titled 'Cluster Information' and 'Configuration Settings'. It is divided into three sections: Version Information, License Information, and Service Information. The Service Information section contains a table of hosts and their roles. Two hosts are circled in red: 'hyperstore-demo1.cloudianhyperstore.com' under 'PUPPET MASTER HOST' and 'hyperstore-demo3' under 'PUPPET MASTER BACKUP HOST'.

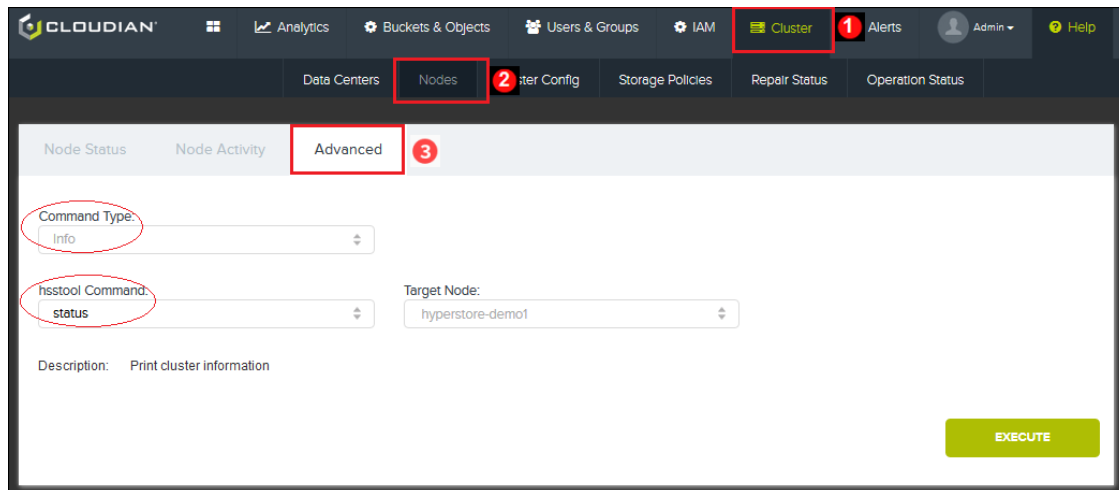
Host	Role
hyperstore-demo1.cloudianhyperstore.com	PUPPET MASTER HOST
hyperstore-demo3	PUPPET MASTER BACKUP HOST

**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, execute the `status` command on any healthy node to **verify that both the Cassandra Service and the HyperStore Service are up on all nodes** in the region.

#### 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.



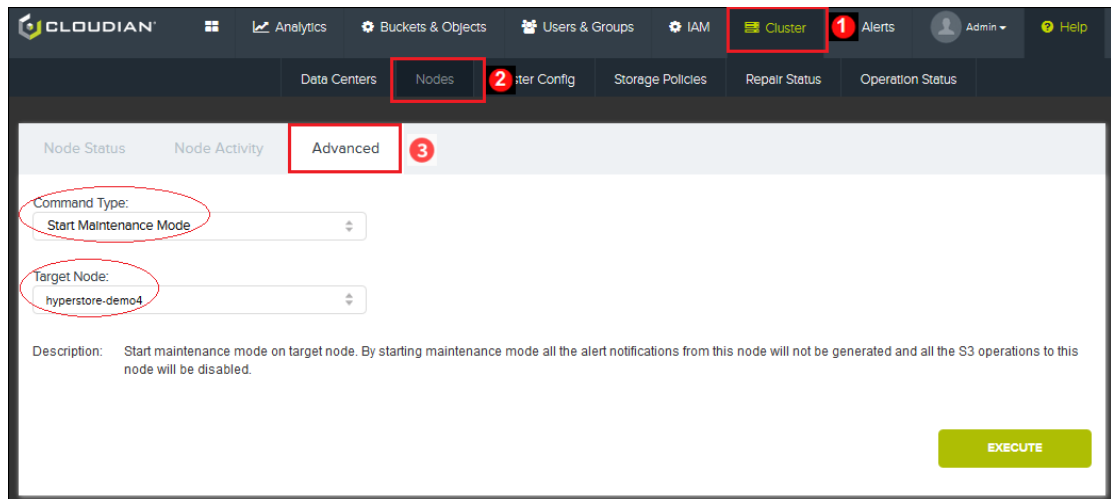
- a. In the command response, for the **node that you want to remove** check the status of the Cassandra Service (in the "Cassandra" column) and the HyperStore Service (in the "Hss" column).
  - If the **Cassandra status and the HyperStore Service status are both Up**, the data redistribution that will 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.

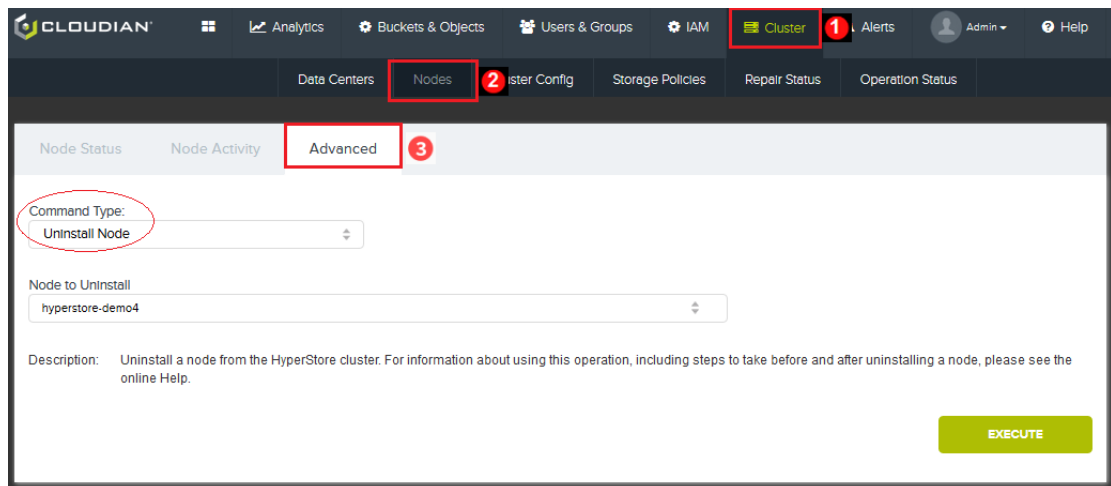
### 6.5.2. Removing a Node

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*
2. 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.



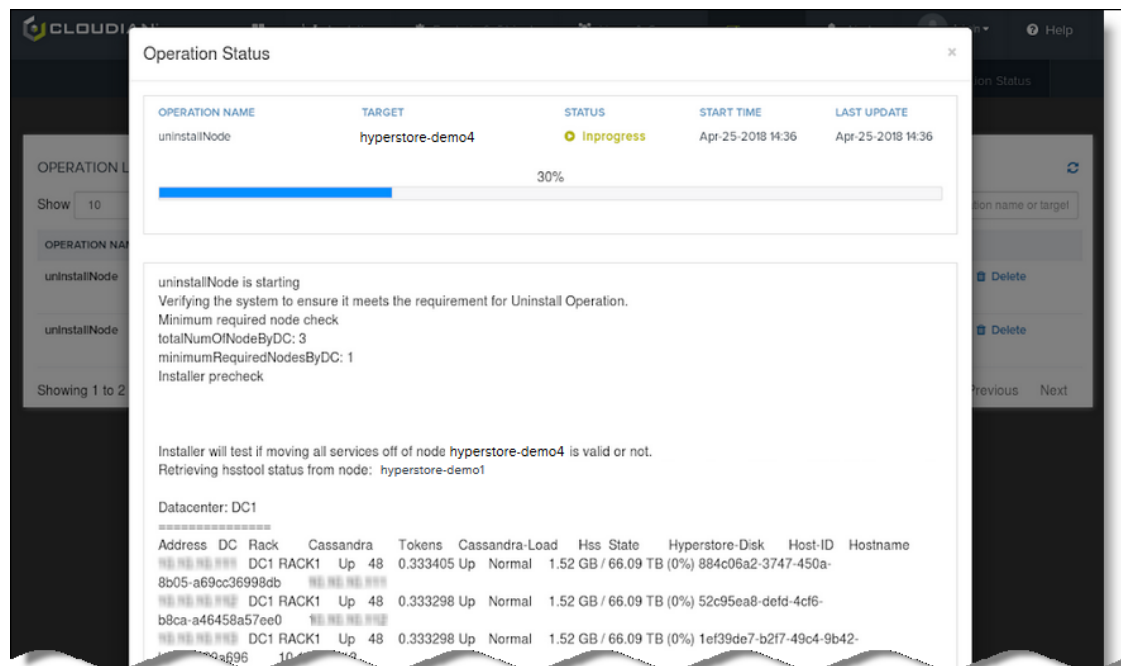
Then click **Execute**. This directs the rest of the cluster to stop sending S3 requests to the specified node.

3. 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.



Then click **Execute**. After you confirm that you want to proceed, the operation is initiated.

4. 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 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 443)) -- 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 420) 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 Cloudian management console. The top navigation bar includes 'Cluster' (1), 'Alerts', 'Admin', and 'Help'. The main navigation bar has 'Data Centers', 'Nodes' (2), 'Cluster Config', 'Storage Policies', 'Repair Status', and 'Operation Status'. The 'Nodes' page is active, showing 'Node Status', 'Node Activity', and 'Advanced' (3) tabs. In the 'Advanced' tab, the 'Command Type' is set to 'Maintenance'. The 'hsstool Command' is set to 'repair'. The 'Target Node' is 'hyperstore-demo1'. Under 'Options', the 'allkeyspaces' radio button is selected, and the checkboxes for 'pr', 'l', and 'm' are all checked. 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 at the bottom right.

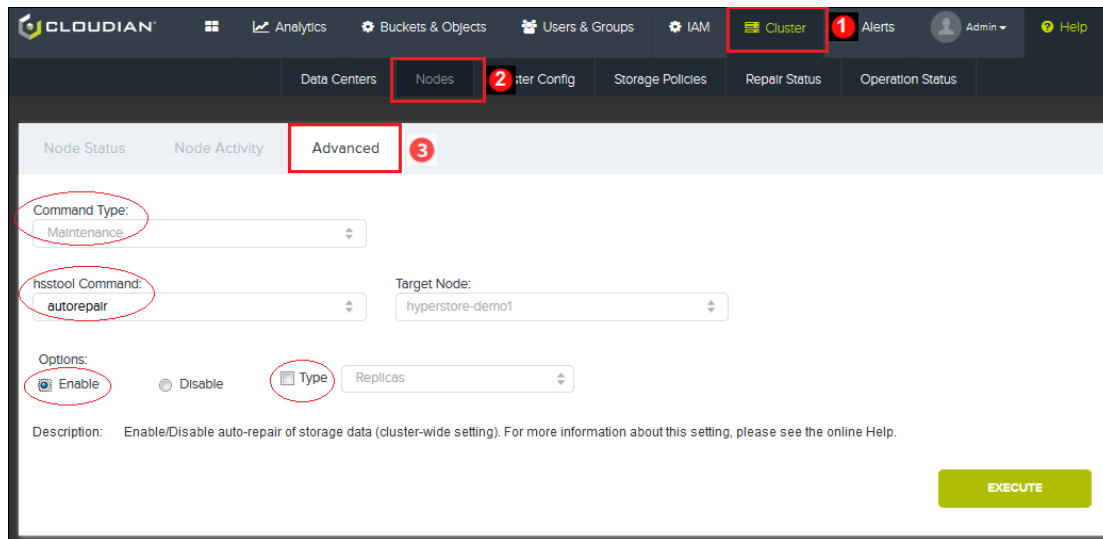
Also, if you have any erasure coded object data in your system, from the **Node Advanced** page run [hsstool repairrec](#) on one node in each HyperStore data center in the region. It doesn't matter which node you run it on, as long as you do it for one node in each DC in the region. (Note that you do 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 Cloudian management console with the same navigation as the previous image. In the 'Nodes' page, the 'Advanced' tab is active. The 'Command Type' is 'Maintenance'. The 'hsstool Command' is set to 'repairrec'. The 'Target Node' is 'hyperstore-demo1'. Under 'Options', the 'l' checkbox is checked, and the 'computedigest', 'stop', 'resume', and 'rebuild' checkboxes are unchecked. There are also fields for 'Mount Point', 'Token Range', 'start', and 'end'. A description at the bottom reads: 'Repair erasure code HyperStore node data'. An 'EXECUTE' button is 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.

## 6.6. 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 443) (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 Cloudian Support for guidance.

- Then perform the procedure for **"Adding Nodes"** (page 420) (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 420) (for the new node).
- Then perform the procedure for **"Removing a Node"** (page 443) (for the live node that you want to remove).

## 6.7. Restoring a Node That Has Been Offline

When a node has been down or inaccessible for a while 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 **"6.7.1 Repairing a Node That's Been Down for Longer than the Proactive Repair Limit"** (page 454).

- 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 Puppet master node [do a Puppet push](#) 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 644) (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...
Redis Credentials master	Automatically failed over to a Redis Credentials slave	Redis Credentials slave
Redis QoS 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
Puppet Master primary	An operator may (or may not) have manually triggered a fail-over to the Puppet Master backup	Still Puppet Master primary, or else Puppet Master 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 457).

### 6.7.1. 6.7.1 Repairing a Node That's Been Down for Longer than the Proactive Repair Limit

In [mts.properties.erb](#) the setting **"hyperstore.proactiverepair.queue.max.time"** (page 579) 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 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.proactiverepair.queue.max.time*, 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 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 Cassandra 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.

## 6.8. Changing a Node's IP Address

Changing the IP address of an existing node in your cluster would have many repercussions for the cluster and 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 Clouidian Support.

## 6.9. Backing Up and Restoring a Cluster

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Backing Up an Entire Cluster"** (page 455)
- **"Restoring an Entire Cluster"** (page 456)

This procedure is for backing up and restoring an entire HyperStore cluster (i.e. an entire HyperStore service region). This procedure should not be used for partial backups and restores.

**IMPORTANT !** Make sure you have enough space for the backup. If you back up on to nodes in your HyperStore cluster, this will double disk usage within the cluster.

**Note** Throughout this procedure, it's assumed that you have used the default installation directories for HyperStore binaries. If not, adjust the command paths stated in the procedure.

### 6.9.1. Backing Up an Entire Cluster

**Note** Daily at 11PM HyperStore implements a Redis cron job (configured in `/etc/cron.d/redis-crontab`) that invokes a script (`/opt/redis/redisBackup`) on your Redis Credentials master node and on your Redis QoS master node. The script backs up the Redis Credentials database into a `/var/lib/redis/dump-credentials.rdb` file and backs up the Redis QoS database into a `/var/lib/redis/dump-qos.rdb` file. Prior to doing so it first moves the previous day's `dump-credentials.rdb` or `dump-qos.rdb` file into a `/var/lib/redis/backup` directory, compresses the file, and appends a date-time stamp to its name (for example `/var/lib/redis/backup/dump-credentials.rdb.2019111616.gz`). In the `/var/lib/redis/backup` directory, files are rotated such that the most recent seven days worth of Redis backups are retained and older backups are deleted.

So you always have access to Redis backup files that are generated at 11PM daily (more specifically, the backup starts at 11PM -- it may take some time to complete). However, if you want a more current Redis backup you can follow steps 1a and 1b below to re-generate the `dump-credentials.rdb` and `dump-qos.rdb` files.

Logging output from the `redisBackup` script runs is written to `/var/log/redis/redisBackup.log`. (A separate log file `clouidian-redisBackup.log` merely records information regarding the launching of the backup script by `cron.d`.)

1. Back up the Redis Credentials and Redis QoS databases.

- a. On the Redis Credentials master node, use the Redis CLI to perform a [BGSAVE](#) operation:

```
# /opt/redis/redis-cli -h <hostname> -p 6379 BGSAVE
```

The above command will update the *dump-credentials.rdb* file in the Redis data directory (*/var/lib/redis* by default).

**Note** The BGSAVE operation saves the database in the background. To check when the save is done you can use the Redis CLI [LASTSAVE](#) command, which returns the Unix time of the most recently completed save.

- b. On the Redis QoS master node, use the Redis CLI to perform a [BGSAVE](#) operation:

```
# /opt/redis/redis-cli -h <hostname> -p 6380 BGSAVE
```

The above command will update the *dump-qos.rdb* file in the Redis data directory.

- c. Place a copy of the *dump-credentials.rdb*, *dump-qos.rdb*, *appendonly\_cred.aof*, and *appendonly\_qos.aof* files in a safe place.
2. Back up Cassandra and HyperStore data **on each of your HyperStore nodes**, using the third party backup tool of your choice:
- For Cassandra data, on each node:
    - First flush Cassandra:

```
# /opt/cassandra/bin/nodetool -h <hostname> flush
```
    - Then with your third party tool, back up the Cassandra data directory. (To check which directory is the Cassandra data directory, see the configuration setting *common.csv*: "**cas-sandra\_data\_directory**" (page 535)).
  - For HyperStore data, on each node: With your third party tool, back up the HyperStore data directories. (To check which directories are the HyperStore data directories, see the configuration setting *common.csv*: "**hyperstore\_data\_directory**" (page 517)).
3. Back up HyperStore binaries and configuration files:
- On the Puppet master node:
    - */etc/cloudian-<version>-puppet* for HyperStore version 5.2 or later, or */etc/puppet* for versions older than 5.2
    - The current HyperStore installation staging directory (where the *cloudianInstall.sh* tool is)
  - On each of your HyperStore nodes:
    - */opt/cloudian*
    - */opt/cassandra*
    - */opt/redis*
    - */opt/tomcat*
    - */opt/cloudianagent*
    - */opt/dnsmasq*

## 6.9.2. Restoring an Entire Cluster

This restore procedure assumes that the restored cluster is the same configuration as what you backed up — i.e. the same IP addresses and mount points.

1. Restore HyperStore binaries and configuration files.
  - On the Puppet master node:
    - `/etc/cloudian-<version>-puppet` for HyperStore version 5.2 or later, or `/etc/puppet` for versions older than 5.2
    - The HyperStore installation staging directory (where the `cloudianInstall.sh` tool is)
  - On each of your HyperStore nodes:
    - `/opt/cloudian`
    - `/opt/cassandra`
    - `/opt/redis`
    - `/opt/tomcat`
    - `/opt/cloudianagent`
    - `/opt/dnsmasq`
2. Restore Cassandra and HyperStore data directories **on each of your HyperStore nodes**.
3. Restore the Redis Credentials and Redis QoS databases.

## 6.10. 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 Puppet master node. The installer supports the role assignment operations listed below.

**Note** Each of the role assignment change operations listed below entails doing a Puppet 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 Puppet push and affected service restart at the end of each operation. Do **not** perform multiple different role assignment change operations while deferring the Puppet push and service restart.

- **"Move the Redis Credentials Master or QoS Master Role"** (page 458)
- **"Move or Add a Redis Credentials Slave or Redis QoS Slave"** (page 461)
- **"Move the Cassandra Seed Role"** (page 462)
- **"Reduce or Change the List of CMC Hosts"** (page 464)
- **"Move the Redis Monitor Primary or Backup Role"** (page 465)
- **"Move the Cron Job Primary or Backup Role"** (page 466)
- **"Move the Puppet Master Primary or Backup Role"** (page 468)
- **"Change Internal NTP Servers or External NTP Servers"** (page 471)

### 6.10.1. Move the Redis Credentials Master or QoS Master Role

The [Redis Credentials master](#) role is assigned to just one node in your entire HyperStore system (the system has just one master even if there are multiple service regions). The [Redis QoS master](#) role is assigned to one node in each of your service regions (each region has its own master). The HyperStore installer automatically makes these role assignments when you install your system.

If you wish you can move the Redis Credentials master role to a current Redis Credentials slave node, or move a Redis QoS master role to a current Redis QoS 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.

#### 6.10.1.1. Preparing to Move the Redis Master Role

For safety, before moving a Redis 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 **Redis Credentials** master database:

```
# /opt/redis/redis-cli -h <hostname_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 **Redis QoS** master database:

```
# /opt/redis/redis-cli -h <hostname_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.

#### 6.10.1.2. Moving the Redis Master Role

1. Submit a Redis CLI *info* command to the **Redis 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 Redis Credentials slave, or use port 6380 if the target node is a Redis QoS slave.

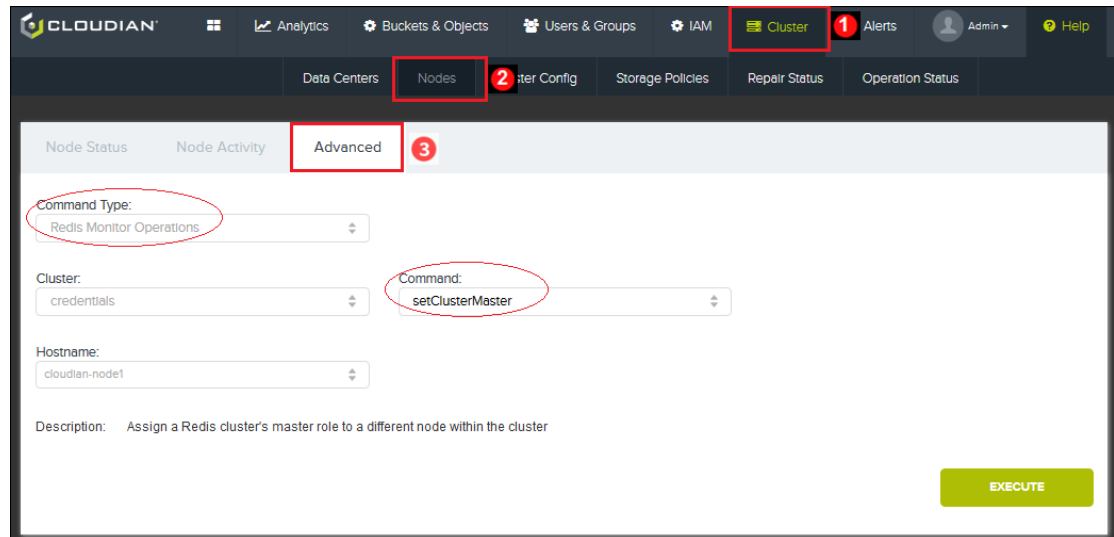
For example, if the Redis Credentials master role is currently on host "cloudian-node2" and you want to move it to host "cloudian-node7" where there is currently a Redis Credentials slave:

```
# /opt/redis/redis-cli -h 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 Redis master role. The drop-down list will show only nodes that are currently slaves within the Cluster type 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 Puppet master node, change into the [installation staging directory](#) 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 Puppet 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.

- 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
6 ) CMC nodes
7 ) Admin nodes
8 ) Primary Redis monitor node
9 ) Backup Redis monitor node
A ) Cronjob/Cluster Monitor node
B ) Installer/Config Manager backup node
C ) NTP Server Configuration
D ) Audit log nodes
R ) Review cluster configuration
X ) Return to Main Menu

Choice: █

```

- c. From the Change Server Role Assignments menu select the option for the master that you want to move — Credentials master or QoS master.
- d. At the prompt indicate the host to which you want to move the Redis 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 to the Puppet master.
- f. Return to the installer's main menu, then choose "Cluster Management" → "Push Configuration Settings to Cluster" and follow the prompts. This triggers Puppet to push your configuration change out to the cluster.
- g. Return to the "Cluster Management" menu, then choose "Manage Services" and restart the following services:
  - The Redis service for which you've changed the master role (either Redis Credentials or Redis QoS)
  - Redis Monitor
  - S3 Service (restarting this service also results in a restart of the Admin Service)

After these services have successfully restarted you can exit the installer.

To verify your change, log into the CMC and go to the [Cluster Information](#) page. Review the service information section to confirm that the Redis master that you moved is now where you want it to be.

Now, the former Redis 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 Redis Credentials Slave or Redis QoS Slave"** (page 461).

### 6.10.2. Move or Add a Redis Credentials Slave or Redis QoS Slave

By default the HyperStore installer deploys two [Redis Credentials slaves](#) and one [Redis QoS slave](#) per data center. The system supports moving a slave from its current host to a different host -- for example, moving a Redis Credentials slave to a host that is not currently running the Redis Credentials service. It also supports adding a new Redis Credentials or QoS slave rather than moving one — so that you end up with more slaves than when you started.

1. On the Puppet master node, change into the [installation staging directory](#) 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 Puppet 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.

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
6 ) CMC nodes
7 ) Admin nodes
8 ) Primary Redis monitor node
9 ) Backup Redis monitor node
A ) Cronjob/Cluster Monitor node
B ) Installer/Config Manager backup node
C ) NTP Server Configuration
D ) Audit log nodes
R ) Review cluster configuration
X ) Return to Main 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 to the Puppet master.
6. Go to the installer's main menu, then choose "Cluster Management" → "Push Configuration Settings to Cluster" and follow the prompts. This triggers Puppet to push your configuration change out to the cluster.
7. Return to the "Cluster Management" menu, then choose "Manage Services" and 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)

After these services have successfully restarted you can exit the installer.

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.

### 6.10.3. Move the Cassandra Seed Role

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** Cassandra runs on every one of your HyperStore nodes. So 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 Puppet master node, change into the [installation staging directory](#) and then launch the HyperStore installer.

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

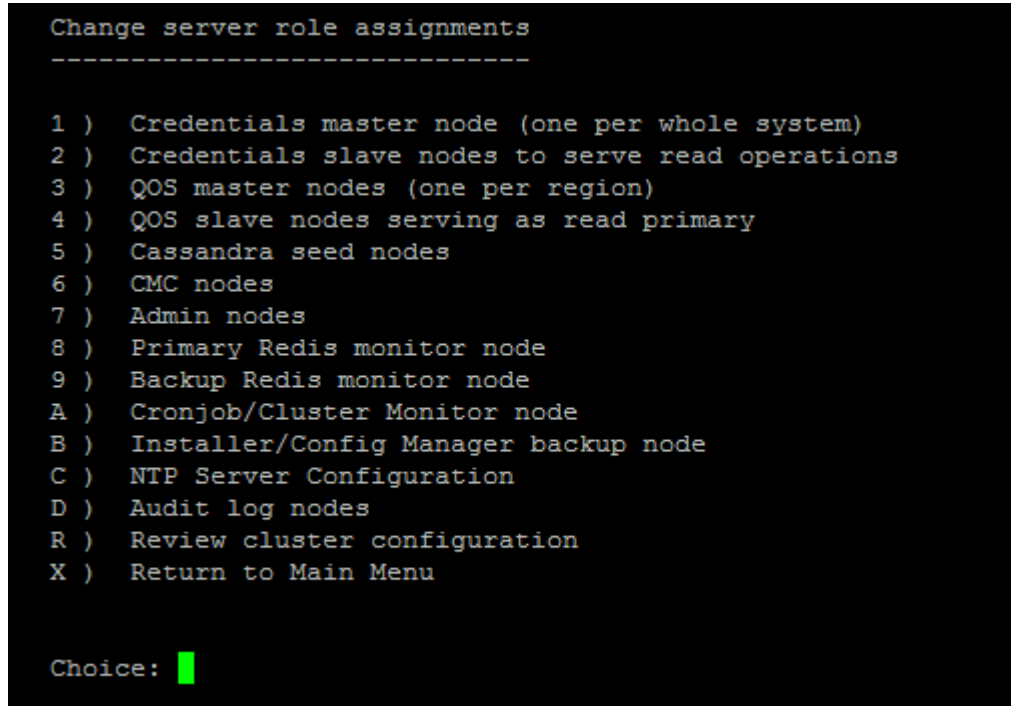
*If you are using the HyperStore Shell*

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Puppet 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.

- 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
6 ) CMC nodes
7 ) Admin nodes
8 ) Primary Redis monitor node
9 ) Backup Redis monitor node
A ) Cronjob/Cluster Monitor node
B ) Installer/Config Manager backup node
C ) NTP Server Configuration
D ) Audit log nodes
R ) Review cluster configuration
X ) Return to Main 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 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 to the Puppet master.
- Go to the installer's main menu and choose "Cluster Management" → "Push Configuration Settings to Cluster" and follow the prompts. This triggers Puppet to push your configuration change out to the cluster.
- Return to the "Cluster Management" menu, then choose "Manage Services" and restart the Cassandra service. After the Cassandra service has successfully restarted you can exit the installer.

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.

### 6.10.4. Reduce or Change the List of CMC Hosts

By default the [Cloudian Management Console](#) (CMC) 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.

below.

**Note** Cassandra runs on every one of your HyperStore nodes. So 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 Puppet master node, change into the [installation staging directory](#) 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 Puppet 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.

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
6 ) CMC nodes
7 ) Admin nodes
8 ) Primary Redis monitor node
9 ) Backup Redis monitor node
A ) Cronjob/Cluster Monitor node
B ) Installer/Config Manager backup node
C ) NTP Server Configuration
D ) Audit log nodes
R ) Review cluster configuration
X ) Return to Main 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 to the Puppet master.
6. Go to the installer's main menu and choose "Cluster Management" → "Push Configuration Settings to Cluster" and follow the prompts. This triggers Puppet to push your configuration change out to the cluster.
7. Return to the "Cluster Management" menu, then choose "Manage Services" and restart the CMC service. After the CMC service has successfully restarted you can exit the installer.

### 6.10.5. 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.

**Note** In a multi- data center HyperStore system, the Redis Monitor backup should remain in the same data center as the Redis Monitor primary, and this should be the same data center as where the Redis Credentials master is located.

The system supports moving the primary or backup Redis Monitor to a different host as described below.

1. On the Puppet master node, change into the [installation staging directory](#) 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 Puppet 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.

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
6 ) CMC nodes
7 ) Admin nodes
8 ) Primary Redis monitor node
9 ) Backup Redis monitor node
A ) Cronjob/Cluster Monitor node
B ) Installer/Config Manager backup node
C ) NTP Server Configuration
D ) Audit log nodes
R ) Review cluster configuration
X ) Return to Main 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 to the Puppet master.
6. Go to the installer's main menu and choose "Cluster Management" → "Push Configuration Settings to Cluster" and follow the prompts. This triggers Puppet to push your configuration change out to the cluster.
7. From the "Cluster Management" menu choose "Manage Services" and restart the Redis Monitor service. After the Redis Monitor successfully restarts you can exit the installer.

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.

### 6.10.6. 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 Puppet Master role. If the cron job primary and the Puppet Master 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 Puppet master node, change into the [installation staging directory](#) 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 Puppet 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.

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
6 ) CMC nodes
7 ) Admin nodes
8 ) Primary Redis monitor node
9 ) Backup Redis monitor node
A ) Cronjob/Cluster Monitor node
B ) Installer/Config Manager backup node
C ) NTP Server Configuration
D ) Audit log nodes
R ) Review cluster configuration
X ) Return to Main 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 to the Puppet master.
6. Go to the installer's main menu and choose "Cluster Management" → "Push Configuration Settings to Cluster" and follow the prompts. When Puppet pushes the current configuration settings to the cluster it will also automatically restart *cron.d* on the affected nodes. You do not need to manually restart any services. When the Puppet push completes you can exit the installer.

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.

### 6.10.7. Move the Puppet Master Primary or Backup Role

When you install your HyperStore system, you choose the node that you want to serve as the Puppet Master for cluster configuration management, and you run the installer on that node. The installer configures that node to be the primary Puppet Master, and also configures a second node to be the Puppet Master backup. Any edits that you make to configuration templates on the Puppet Master primary are automatically sync'd to the Puppet Master backup. If the primary goes down, you can **manually** fail over the active Puppet Master role to the backup host.

There are two different operations that you can perform in regard to the Puppet Master role:

#### Move the Puppet Master Backup

In this scenario your primary Puppet Master is fine, and you're simply looking to relocate the Puppet Master backup role to a different host than it is currently on.

**Note** You can find out which host is currently the Puppet Master backup host in the CMC's [Cluster Information](#) page.

1. On the Puppet Master primary host, change into the [installation staging directory](#) 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 Puppet 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.

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
6 ) CMC nodes
7 ) Admin nodes
8 ) Primary Redis monitor node
9 ) Backup Redis monitor node
A ) Cronjob/Cluster Monitor node
B ) Installer/Config Manager backup node
C ) NTP Server Configuration
D ) Audit log nodes
R ) Review cluster configuration
X ) Return to Main Menu

Choice: █

```

3. From the Change Server Role Assignments menu select "Installer/Config Manager backup node".
4. At the prompt specify the host to which you want to move the Puppet Master backup role.
5. After completing the interaction for specifying the Puppet Master 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 to the Puppet master.
6. Go to the installer's main menu and choose "Cluster Management" → "Push Configuration Settings to Cluster" and follow the prompts. This triggers Puppet to push your configuration change out to the cluster. There is no need to restart any services.
7. Exit the installer, 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 Puppet Master primary and backup hosts are what you want them to be.

## Manually Fail Over the Puppet Master Role from the Primary to the Backup

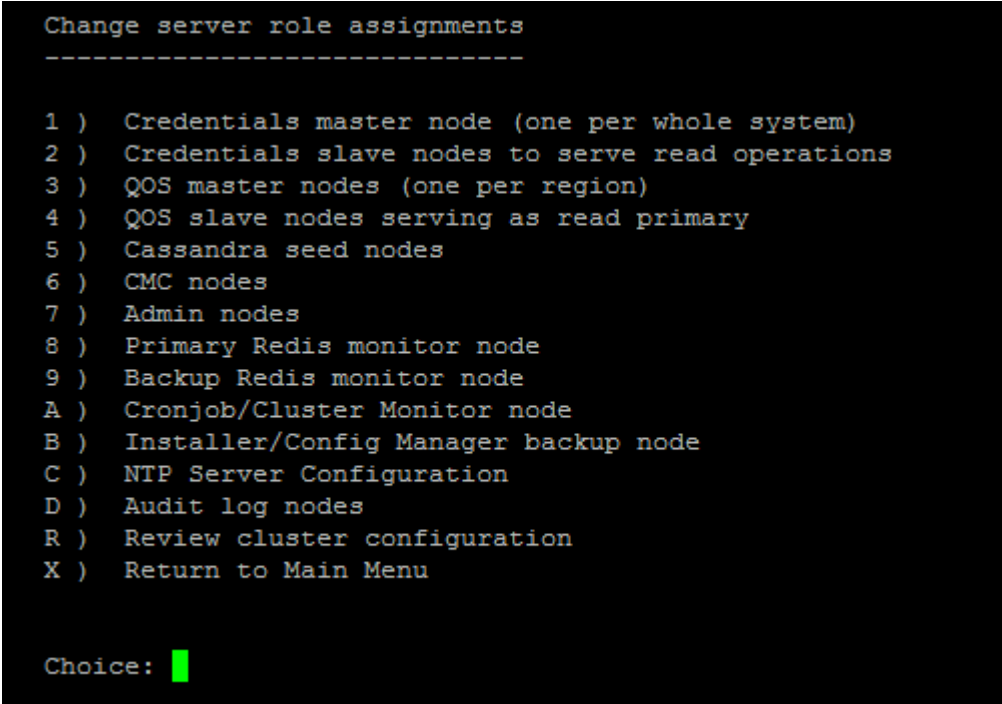
In this scenario there is a problem with your primary Puppet Master, and you want the backup Puppet Master to become active.

**IMPORTANT !** For this procedure you **log into the current Puppet Master backup host** and implement the whole procedure from that host. You can find out which host is currently the Puppet Master backup host in the CMC's [Cluster Information](#) page.

1. **On the Puppet Master backup host**, change into the installation directory and then launch the Hyper-Store installer.  

```
# ./cloudianInstall.sh
```
2. From the installer's main menu, select "Advanced Configuration Options". Then from the Advanced Configuration Options menu select "Start or stop Puppet daemon".

3. At the prompt specify that you want to stop Puppet. This stops any Puppet daemons that are currently running in your cluster.
4. After the installer indicates that Puppet has been stopped, return to the Advanced Configuration Options menu and select "Remove existing Puppet SSL certificates". This will remove existing Puppet SSL certificates, with no further prompts.
5. Return to the Advanced Configuration Options menu and 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
6 ) CMC nodes
7 ) Admin nodes
8 ) Primary Redis monitor node
9 ) Backup Redis monitor node
A ) Cronjob/Cluster Monitor node
B ) Installer/Config Manager backup node
C ) NTP Server Configuration
D ) Audit log nodes
R ) Review cluster configuration
X ) Return to Main Menu

Choice: █
```

6. From the Change Server Role Assignments menu select "Installer/Config Manager backup node".
7. At the prompt specify the host to which you want to move the Puppet Master (config manager) 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).
8. After completing the interaction for specifying the Puppet Master 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 to the Puppet master.
9. Go to the installer's main menu and choose "Cluster Management" → "Push Configuration Settings to Cluster" and follow the prompts. This triggers Puppet to push your configuration change out to the cluster. Then do "Cluster Management" → "Manage Services" and restart the CMC.
10. Optionally, if you want to leave the Puppet daemons running, from the installer's main menu select "Advanced Configuration Options". Then from the Advanced Configuration Options menu select "Start or stop Puppet daemon", and choose to start the daemons. After the daemons have successfully started you can exit the installer.

To verify your change, log into the CMC and go to the [Cluster Information](#) page. Review the service information section to confirm that your Puppet Master 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 Puppet Master 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 Puppet Master 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.

### 6.10.8. 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. These internal NTP servers synchronize with external NTP servers. By default the set of external servers is:

- 0.centos.pool.ntp.org
- 1.centos.pool.ntp.org
- 2.centos.pool.ntp.org
- 3.centos.pool.ntp.org

**Note** For more on how HyperStore automatically configures an NTP set-up for the cluster, see **"NTP Automatic Set-Up"** (page 598)

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 Puppet master node, change into the [installation staging directory](#) 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 Puppet 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.

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
6 ) CMC nodes
7 ) Admin nodes
8 ) Primary Redis monitor node
9 ) Backup Redis monitor node
A ) Cronjob/Cluster Monitor node
B ) Installer/Config Manager backup node
C ) NTP Server Configuration
D ) Audit log nodes
R ) Review cluster configuration
X ) Return to Main 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 to the Puppet master.
7. Go to the installer's main menu and choose "Cluster Management" → "Push Configuration Settings to Cluster" and follow the prompts. When Puppet pushes the current configuration settings to the cluster it will also automatically restart *ntpd* on the affected nodes. You do not need to manually restart *ntpd*. When the Puppet push completes you can exit the installer.

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.

## 6.11. 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 473)
- **"Scheduled Auto-Repair"** (page 478)
- **"Cassandra Data Compaction"** (page 478)

### 6.11.1. 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. In the 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 configured in the `/etc/cron.d/cloudian-crontab` file on the host node. If you want to adjust the scheduling of these cron jobs you should do so via Puppet configuration management, by editing the configuration template file `/etc/cloudian-<version>-puppet/modules/cloudians3/templates/cloudian-crontab.erb` on the Puppet master node. For general information on how to configure cron job scheduling, refer to any reputable resource on the topic.

Note that most of the cron jobs configured in `cloudian-crontab.erb` have `"> /dev/null 2>&1"` appended to them and therefore they direct all output to `/dev/null`.

System cron jobs are implemented for the following system maintenance tasks:

#### System Monitoring Data Collection

Scope: One job per region

Frequency: Every 1 minute

Operation invoked: S3 Service internal process `/bin/snapshotData`

This cron job invokes a process that logs "snapshot" system statistics each minute in support of the Cloudian Management Console's system monitoring functionality.

#### Diagnostic Log Upload

Scope: One job per region

Frequency: Once per day

Operation invoked: S3 Service internal process `/bin/phoneHome`

This cron job uploads a daily diagnostics file to a configurable S3 URI, if the HyperStore "Phone Home" feature (also known as "Smart Support") is enabled. Typically this would be the S3 URI of Cloudian Support.

**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 Cloudian customer environments are uploading to Cloudian Support at the same time.

## Usage Data Processing

Several cron jobs are involved in processing service usage data for users and groups.

**Note** For an overview of how the HyperStore system tracks service usage by groups and users, see **"Usage Reporting and Billing Feature Overview"** (page 138).

### 6.11.1.0.1. Saving Storage Usage Data for Active Users

Scope: One job per region

Frequency: Every 5 minutes

Operation invoked: Admin API method [POST /usage/storage](#)

This cron job writes snapshots of per-user and per-group counts for stored bytes and number of stored objects from the Redis QoS database over to the "Raw" column family in the Cassandra "Reports" keyspace. The operation is applied only to users and groups that have uploaded or deleted objects in the time since the operation was last executed.

### 6.11.1.0.2. Saving Storage Usage Data for All Users

Scope: One job per region

Frequency: Once a day

Operation invoked: Admin API method [POST /usage/storageall](#)

This cron job writes snapshots of per-user and per-group counts for stored bytes and number of stored objects from the Redis QoS database over to the "Raw" column family in the Cassandra "Reports" keyspace. The operation is applied to **all** users and groups.

### 6.11.1.0.3. Repairing Usage Data for Active Users

Scope: One job per region

Frequency: Every 12 hours

Operation invoked: Admin API method [POST /usage/repair/dirtyusers](#)

This cron job repairs Redis QoS stored bytes and stored object counts for up to 1000 active or "dirty" users. See the Admin API method description for more detail.

### 6.11.1.0.4. Creating Usage Roll-Up Reports

Scope: One job per region for each roll-up granularity (hour, day, month)

Frequency: Hourly, daily, monthly

Operation invoked: Admin API method [POST /usage/rollup](#)

- One job with granularity=hour (runs once per hour)
- One job with granularity=day (runs once per day)
- One job with granularity=month (runs once per month)

These three cron jobs create summary (or "roll-up") usage reports data from more granular reports data. The hourly roll-up data is derived from the raw data (the transactional data and stored bytes/stored object count snapshot data). The daily roll-up data and monthly roll-up data are both derived from the hourly roll-up data.

**Note** Hourly rollup jobs that fail -- such as if a relevant service is down or unreachable -- will be retried. The time span for which failed rollup jobs are eligible for retry is configuration by the **"usage.rollup.hour.maxretry"** (page 569) property in [mts.properties.erb](#). The default is 6 hours.

## Bucket Log Processing

Scope: One job per region

Frequency: Once every 10 minutes

Operation invoked: S3 Service internal process `./system/dumpbucketlogs`

This cron job moves bucket logging data from the Cassandra BucketLog column family into the S3 storage system, in support of the S3 Bucket Logging feature.

## Bucket Lifecycle Processing

Two cron jobs are involved in S3 Bucket Lifecycle implementation.

### 6.11.1.0.5. Auto-Tiering and Auto-Expiring Objects

Scope: One job per region

Frequency: Once a day

Operation invoked: S3 Service internal process `./system/autoexpire`

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.
- 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.

**Note** This cron job for auto-tiering and auto-expiring objects distributes the required processing work across all nodes in the same service region as the cron job primary host.

**Note** For more information on the HyperStore auto-tiering feature, see **"Auto-Tiering Feature Overview"** (page 176).

### 6.11.1.0.6. Restoring Auto-Tiered Objects

Scope: One job per region

Frequency: Every six hours

Operation invoked: S3 Service internal process `./system/restore`

This cron job executes queued object-restore jobs. Restore jobs are placed in queue when S3 clients invoke the S3 API method **"RestoreObject"** (page 988), to restore a local copy of objects that have been auto-tiered to a remote storage system. The cron job executes queued restored jobs every six hours.

## Tombstone Cleanup Processing

Scope: One job per region

Frequency: Every hour

Operation invoked: S3 Service internal process `./system/cleantombstones`

This hourly operation cleans (removes) Cassandra "tombstones", which are markers that indicate that a cell or row has been deleted.

### 6.11.1.0.7. Dealing with Excessive Tombstone Build-Up

Under normal circumstances the hourly running of the `./system/cleantombstones` cron job should ensure that there is no excessive build-up of Cassandra tombstones. However, it is possible to encounter `TombstoneOverwhelmingException` errors in Cassandra logs and an inability to successfully execute an S3 [GET Bucket \(List Objects\)](#) operation against a specific bucket, in either of these unusual circumstances:

- An S3 client application has attempted to delete more than 100,000 objects from the bucket in less than an hour.
- Over the course of multiple hours an S3 client application has attempted to delete more than 100,000 objects from the bucket and during that time the hourly `./system/cleantombstones` cron job has failed to purge tombstones for one reason or another.

In such circumstances you can trigger tombstone removal by connecting to any S3 server's JMX port (19080) and submitting a `purgeTombstone` command with the bucket name as input. If you are using JConsole, after connecting to port 19080 on an S3 node select the *MBeans* tab, then select `com.cloudian.ss.cassandra.cl` → *BatchJobs* → *Operations* → `purgeTombstone`. On the right side of the console enter the bucket name as the *p1* value and then execute the operation (by clicking `purgeTombstone` on the right side of the console).

## User Deletion Cleanup

Scope: One job per HyperStore system (run only in default region)

Frequency: Once a day

Operation invoked: Admin API method `POST /system/deletedUserCleanup` (for system use only; this API method is not documented)

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.

## Storage Policy Deletion or Creation Processing

Scope: One job per HyperStore system (run only in default region)

Frequency: Once a day

Operation invoked: Admin API method [POST /system/processProtectionPolicy](#)

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 Cassandra 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.

## Retries of Pending Cross-Region Replication Jobs

Scope: One job per region

Frequency: Every four hours

Operation invoked: S3 Service internal process `./system/processscr`

This cron job processes any pending [cross-region replication](#) jobs. These pending jobs result when an attempt to replicate an object from the source bucket to the destination bucket results in 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.

## Retries of Pending "Bridge Mode" Auto-Tiering Jobs

Scope: One job per HyperStore system (run only in default region)

Frequency: Every six hours

Operation invoked: S3 Service internal process `./system/processproxy`

This cron job processes any pending "bridge mode" auto-tiering jobs. When a HyperStore source bucket has auto-tiering configured in bridge mode, HyperStore moves objects to the auto-tiering destination immediately after they are uploaded to the source bucket. This cron job is for retrying to move objects for which the original attempt at moving the objects failed. For such objects, the once-every-six-hours retries will continue indefinitely until the objects are successfully moved to the tiering destination system.

For more information on this feature see **"Bridge Mode"** (page 178).

## Processing of Pending Elasticsearch Update Requests

Scope: One job per HyperStore system (run only in default region)

Frequency: Every hour

Operation invoked: S3 Service internal process `./system/processelasticsearch`

All insertions, updates, and deletes of HyperStore object metadata in your Elasticsearch cluster are implemented by this cron job. Until the cron job runs the Elasticsearch update requests are queued in Cassandra. Any requests that fail when this cron job runs are retried at the next running of the cron job.

**Note** In the event that your Elasticsearch cluster is unavailable for more than a few hours, the request queue will become full and when the cluster is available again you should use the `elasticsearchSync` tool to update the metadata in Elasticsearch. For more information about this tool see **"Elasticsearch Integration for Object Metadata"** (page 171). For general information about the Elasticsearch integration feature see **"Elasticsearch Integration for Object Metadata"** (page 171).

### 6.11.2. Scheduled Auto-Repair

On a configurable schedule, the HyperStore "auto-repair" feature automatically checks and if necessary repairs S3 object data in the HyperStore File System as well as metadata in Cassandra. For more information on scheduled auto-repair and other HyperStore mechanisms for ensuring that data in your system is complete and correct, see **"Automated Data Repair Feature Overview"** (page 150).

### 6.11.3. Cassandra Data Compaction

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 by using JConsole or another JMX client to connect to a Cassandra node'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.

# Chapter 7. Disk Operations

## 7.1. Disabling a HyperStore Data Disk

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"The Impact of Disabling a Disk"** (page 479)
- **"Disabling a Disk"** (page 479)

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** If you are **replacing** a disk, follow the instructions for **"Replacing a HyperStore Data Disk"** (page 482) rather than the instructions below.

### 7.1.1. The Impact of Disabling a Disk

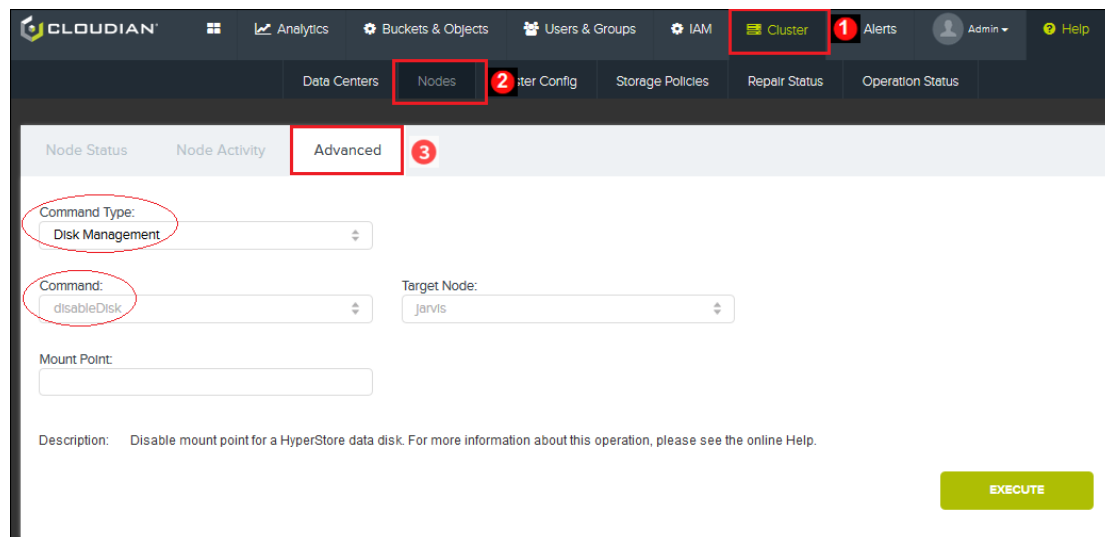
When you execute the HyperStore *disableDisk* function, the system automatically does the following:

- 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.

### 7.1.2. 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 480). When you re-enable the disk, the tokens that had been moved away from the disk will be moved back to it.

## 7.2. Enabling a HyperStore Data Disk

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"The Impact of Enabling a Disk"** (page 481)
- **"Enabling a Disabled Disk"** (page 481)

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 479)) or because the HyperStore system automatically disabled it in response to disk errors (as described in **"Automatic Disk Failure Handling"** (page 160)).

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** If you are **replacing** a disk, follow the instructions for "**Replacing a HyperStore Data Disk**" (page 482) rather than the instructions below.

### 7.2.1. 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 160).

### 7.2.2. 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 '1'), 'Alerts', 'Admin', and 'Help'. Below this, the 'Nodes' tab is selected (highlighted with a red box and '2'). The 'Advanced' sub-tab is active (highlighted with a red box and '3'). In the 'Command Type' dropdown, 'Disk Management' is selected (circled in red). In the 'Command' dropdown, 'enableDisk' is selected (circled in red). The 'Target Node' dropdown shows '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 in the bottom right corner.

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.

## 7.3. Replacing a HyperStore Data Disk

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"The Impact of Replacing a Disk"** (page 482)
- **"Replacing a Disk"** (page 483)

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 479)) or because the HyperStore system automatically disabled it in response to disk errors (as described in **"Automatic Disk Failure Handling"** (page 160)).
- You are replacing a disk that is not currently disabled. In this case, it is not necessary for you to use the `disableDisk` function before replacing the disk. 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.

**After you've pulled the bad disk and physically installed the replacement disk**, HyperStore will take care of the rest when you follow the steps in this section.

**Note** If you have a HyperStore HSA1500, HSA4000, or HSX-4500 series appliance, or a Lenovo Storage DX8200C appliance, and you need assistance physically locating the failed disk that you want to pull and replace, you can use the blink light feature on the CMC **Node Status** page.

**Note** The procedure below is for HyperStore data disks only. For an OS/Cassandra drive (typically an SSD), see **"Replacing a Cassandra Disk"** (page 484).

### 7.3.1. The Impact of Replacing a 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 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.

**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 160).

### 7.3.2. Replacing a Disk

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.

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 active (highlighted with a red box and '3'). In the 'Command Type' dropdown, 'Disk Management' is selected. In the 'Command' dropdown, 'replaceDisk' is selected. The 'Target Node' dropdown shows 'jarvis'. The 'Mount Point' field is empty. A description at the bottom reads: 'Mount a physically installed replacement disk and restore data to it. For more information about this operation, please see the online Help.' An 'EXECUTE' button is in the bottom right corner.

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):



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 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 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.

**Note** You do not need to manually run a repair operation after you replace a disk. The system automatically runs *repair* and *repairec* on the disk mount point. You can monitor the repair progress in the [Operation Status](#) page.

## 7.4. Replacing a Cassandra Disk

**Note** The procedure that follows is valid for systems on CentOS/RHEL 6.x. For systems on CentOS/RHEL 7.x the procedure that follows is **not** valid. Please contact Cloudbian Support if your system is on CentOS/RHEL 7.x and you need to replace a Cassandra disk.

Cloudbian, Inc. recommends using RAID mirroring (RAID1) for the drives that store the OS and Cassandra. HyperStore Appliance machines are configured in this way, using software RAID1. If a single drive fails, the other mirrored drive will continue to serve I/O. The recommended procedure for replacing a failed OS/Cassandra drive in the context of software RAID1 is described below.

The procedure described below uses the following sample setup:

- `/dev/sda` with partitions `/dev/sda1` and `/dev/sda2`
- `/dev/sdb` with partitions `/dev/sdb1` and `/dev/sdb2`
- `/dev/sda1 + /dev/sdb1 = /dev/md0` (RAID1 array `/dev/md0` mounted on `/boot`)
- `/dev/sda2 + /dev/sdb2 = /dev/md1` (RAID1 array `/dev/md1` mounted on `/`)

In this example `/dev/sda` has failed, and we want to replace it.

1. Identify the failed drive by checking `/proc/mdstat`.

```
# cat /proc/mdstat
Personalities : [raid1]
md0 : active raid1 sdb1[3]
      511936 blocks super 1.0 [2/1] [_U]

md1 : active raid1 sdb2[2]
      468206592 blocks super 1.1 [2/1] [_U]
      bitmap: 3/4 pages [12KB], 65536KB chunk
```

```
unused devices: <none>
```

To identify the status of the RAID array, look at the string containing [UU]. Each "U" is a representation of a healthy partition member of the array.

From the above output, you can tell that RAID arrays *md0* and *md1* are missing a "U" [\_U]. This means that they are degraded and faulty. (For an example of the output for healthy arrays, look ahead to Step 4-d of this procedure.)

2. Remove the failed drive.

- a. If the drive is not in the failed state yet, fail the partitions belonging to the drive:

```
# mdadm --manage /dev/md0 --fail /dev/sda1
# mdadm --manage /dev/md1 --fail /dev/sda2
```

- b. Remove the partitions from the RAID array:

```
# mdadm --manage /dev/md0 --remove /dev/sda1
# mdadm --manage /dev/md1 --remove /dev/sda2
```

3. Physically replace the failed drive with a new drive of equal or greater capacity.

4. Add the new drive to the RAID array.

- a. Create the exact same partitioning on the new drive.

```
# sfdisk -d /dev/sdb | sfdisk --force /dev/sda
```

**Note** */dev/sda* is the newly replaced drive.

- b. Check that both drives have the same partition:

```
# fdisk -l /dev/sdb

# fdisk -l /dev/sda
```

- c. Add the new partitions to the RAID array:

```
# mdadm --manage /dev/md0 --add /dev/sda1
# mdadm --manage /dev/md1 --add /dev/sda2
```

- d. Wait for the arrays to be synchronized. You can check status of synchronization by executing:

```
# cat /proc/mdstat
```

The output should look like this:

```
Personalities : [raid1]
md0 : active raid1 sda1[2] sdb1[3]
      511936 blocks super 1.0 [2/2] [UU]

md1 : active raid1 sda2[0] sdb2[2]
      468206592 blocks super 1.1 [2/1] [_U]
      [=====>.....]  recovery = 55.5% (260239360/468206592)
      finish=13.4min speed=258048K/sec
      bitmap: 2/4 pages [8KB], 65536KB chunk

unused devices: <none>
```

Once synchronization is complete, it will look like the output below:

```
Personalities : [raid1]
md0 : active raid1 sda1[2] sdb1[3]
511936 blocks super 1.0 [2/2] [UU]

md1 : active raid1 sda2[0] sdb2[2]
468206592 blocks super 1.1 [2/2] [UU]
bitmap: 0/4 pages [0KB], 65536KB chunk

unused devices: <none>
```

5. Install [GRUB bootloader](#) on the new hard drive.

a. Check if GRUB bootloader exists on the mirrored drives:

```
# file -s /dev/sdb

/dev/sdb: x86 boot sector; GRand Unified Bootloader, stage1 version 0x3, stage2 address
0x2000,
stage2 segment 0x200, GRUB version 0.94; partition 1: ID=0xfd, active, starthead 32,
startsector 2048, 1024000 sectors; partition 2: ID=0xfd, starthead 221, startsector
1026048,
936675328 sectors, code offset 0x48

# file -s /dev/sda

/dev/sda: x86 boot sector; partition 1: ID=0xfd, active, starthead 32, startsector
2048,
1024000 sectors; partition 2: ID=0xfd, starthead 221, startsector 1026048, 936675328
sectors,
code offset 0x0
```

From the above outputs, it looks like GRUB is only installed on `/dev/sdb`. It also needs to be installed on `/dev/sda` to ensure that the system will boot regardless of which drive fails in the future.

b. Check the device maps.

```
# cat /boot/grub/device.map
```

The output will look like this:

```
# this device map was generated by anaconda
(hd0)      /dev/sda
(hd1)      /dev/sdb
```

c. Locate GRUB setup files.

Enter the grub command line by executing:

```
# grub
```

At the `grub>` prompt, type `find /grub/stage1`

For example:

```
grub> find /grub/stage1
find /grub/stage1
```

```
(hd0,0)
(hd1,0)
```

- d. Install GRUB on `/dev/sda`.

First:

```
grub> root (hd0,0)
```

You will see the following output:

```
root (hd0,0)
Filesystem type is ext2fs, partition type 0xfd
```

Then:

```
grub> setup (hd0)
```

You will see the following output:

```
setup (hd0)
  Checking if "/boot/grub/stage1" exists... no
  Checking if "/grub/stage1" exists... yes
  Checking if "/grub/stage2" exists... yes
  Checking if "/grub/e2fs_stage1_5" exists... yes
  Running "embed /grub/e2fs_stage1_5 (hd0)"... 27 sectors are
  embedded.
succeeded
  Running "install /grub/stage1 (hd0) (hd0)1+27 p (hd0,0)/grub/stage2
  /grub/grub.conf"...
succeeded
Done.
```

Then:

```
grub> quit
```

- e. Check that the GRUB bootloader is installed on `/dev/sda`.

```
# file -s /dev/sda

/dev/sda: x86 boot sector; GRand Unified Bootloader, stage1 version 0x3, stage2 address
0x2000,
stage2 segment 0x200, GRUB version 0.94; partition 1: ID=0xfd, active, starthead 32,
startsector 2048,
1024000 sectors; partition 2: ID=0xfd, starthead 221, startsector 1026048, 936675328
sectors,
code offset 0x48
```

This completes the process of replacing a mirrored OS/Cassandra drive.

## 7.5. 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 71).

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 cleanuppec](#) 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 deletion of S3 object data. The batch processing is triggered hourly by a cron job (see **"System cron Jobs"** (page 473)).

**Note** For additional guidance on removing data to free up disk space, consult with Clodian 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 420). 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 Cassandra) that is nearing capacity see **"Responding to Cassandra Disks Nearing Capacity"** (page 488).

## 7.6. Responding to Cassandra Disks Nearing Capacity

If a Cassandra data drive (a disk or SSD storing the Cassandra database) 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 Clodian 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 420). 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.

## 7.7. Adding Disks 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 one or more nodes as described in **"Adding Nodes"** (page 420).

**Note** For guidance about HyperStore capacity management and cluster resizing, see **"Capacity Monitoring and Expansion"** (page 71).

This page left intentionally blank

# Chapter 8. System Monitoring

## 8.1. Using the CMC to Monitor Your HyperStore System

The Cloudian 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	<a href="#">Dashboard</a>
	Check the remaining available storage capacity in each service region, as well as capacity remaining in each data center and on each node	<a href="#">Capacity Explorer</a>
	Check how your storage capacity usage has changed over the past 30 days, per service region	<a href="#">Cluster Usage</a>
	For each data center, see which nodes have any alerts and whether any HyperStore services are down on any node	<a href="#">Data Centers</a>
	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.	<a href="#">Node Status</a>
	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.	<a href="#">Node Activity</a>
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.	<a href="#">Alert Rules</a>
	Review and acknowledge alerts generated by any node in the system	<a href="#">Alerts</a>
	Review and acknowledge alerts generated by a specific node	<a href="#">Node Status</a>

## 8.2. Cloudian HyperIQ

Cloudian HyperIQ is a solution for dynamic visualization and analysis of HyperStore monitoring 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.



## 8.3. Additional Monitoring Tools

### 8.3.1. 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 the Admin API methods associated with the **"monitor"** (page 781) resource.

### 8.3.2. Doing an HTTP Health Check

The HyperStore system supports a health check that lets you quickly determine whether certain HTTP interfaces are responsive on particular hosts. This function is supported for:

- **S3 Service** (HTTP port 80 by default; or HTTPS port 443 if you've enabled SSL for the S3 service)
- **Admin Service** (HTTPS port 19443 by default; or HTTP port 18081 if you've configured the Admin Service to accept regular HTTP connections)
- **HyperStore Service** (HTTP port 19090 by default)

To do the health check on a particular host, submit an HTTP(S) HEAD request to `<host>:<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.

For the **CMC**, you can check responsiveness by submitting an 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.

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, 14 Aug 2019 12:51:50 GMT
Server: CloudianS3
```

**Note** To do health checks against an HTTPS port, the client executing the check must support TLS/SSL. Note also that health checks against HTTPS ports are a more expensive operation (in terms of resource utilization) than health checks against HTTP ports.

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:

```
2019-08-16 15:01:33,757|127.0.0.1||healthCheck|||81|0|0|0|81|11820||200|
544cdd90-822f-1c98-b780-525400e89933|0|0|
```

For more information on the S3 Request Log, see "**S3 Service Logs (including Auto-Tiering, CRR, and WORM)**" (page 619)

### 8.3.3. Using JMX to Monitor Java-Based 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.

**Note** 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-jmx-client* 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\_Cloudian<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 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. This is the same information as is available through the [nodetool tpstats](#) command.

**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.

### 8.3.4. Using Native Linux Utilities for System Resource Monitoring

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 (tomcat).

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*

### 8.3.5. Using nodetool to Monitor Cassandra

Through the CMC's [Node Status](#) page you can monitor high-level status information for the Cassandra service instance on a particular node. For more granular Cassandra monitoring you can optionally use the native Cassandra utility *nodetool*.

The *nodetool* utility resides in each Cassandra host's */opt/cassandra/bin* directory. From that directory, the general syntax for *nodetool* is:

```
# ./nodetool -h <host> [-p <CassandraJMXport>] <COMMAND>
```

The Cassandra JMX port defaults to 7199.

Some *nodetool* commands that you may find useful for monitoring a Cassandra cluster are summarized below.

*cfstats* [*<Keyspace.ColumnFamily>*]

Returns information about each keyspace and each column family, including:

- Read count
- Read latency
- Write count
- Write latency
- Pending tasks

For column families only (not keyspaces):

- Memtable stats
- Key cache capacity
- Key cache hit rate

*cfhistograms* <Keyspace> <ColumnFamily>

Returns information for a specific column family, including:

- Read latency
- Write latency
- Row size
- Column count

*tpstats*

Returns information for each thread pool, including:

- Active tasks
- Completed tasks
- Pending tasks

**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.

*compactstats*

Returns information for an in-progress compaction, including:

- Compaction type (major or minor)
- Column family for which the compaction is being performed
- Bytes compacted so far

*netstats*

Returns network information, including:

- Status of streaming operations such as bootstrap, repair, move, or decommission
- Active, pending, and completed command counts

For more information about *nodetool*, see either the Apache Cassandra online documentation or the DataStax Cassandra online documentation.

### 8.3.6. Using the Redis CLI to Monitor Redis

Through the CMC's [Node Status](#) page you can monitor high-level status information for the Redis service (the Redis Credentials master instance or slave instance or the Redis QoS master instance or slave instance) on a particular node. For more granular Redis monitoring you can optionally use the native Redis CLI.

To launch the Redis CLI on a Redis node, change into the node's */opt/redis* directory and do either of the following:

```
### to connect to the CLI for the Redis Credentials DB:
# ./redis-cli
redis 127.0.0.1:6379>
```

```
### to connect to the CLI for the Redis QoS DB:  
# ./redis-cli -p 6380  
redis 127.0.0.1:6380>
```

Once you're in the Redis CLI mode, you can issue commands. Use "quit" to exit the CLI.

With the Redis [INFO command](#) you can retrieve statistics such as:

- Uptime since last start
- Memory usage as allocated by Redis
- Memory usage as seen by OS
- Memory fragmentation ratio
- Number of dataset-changing operations processed since last save
- Keyspace hits
- Keyspace misses
- Total keys
- Total expired keys

You can reset Redis statistics with the [CONFIG RESETSTAT command](#).

To check just on the number of keys currently in the database, you can use the [DBSIZE command](#).

Also useful is the [MONITOR command](#), which enables you to monitor the complete sequence of commands received by the DB in near real-time.

In addition to monitoring Redis through its CLI, you should also monitor the size of the Redis data files (in default directory `/var/lib/redis/`) to ensure files are not growing excessively.

# Chapter 9. System Configuration

## 9.1. 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 337).

## 9.2. 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. In most cases, these are configuration tasks that can also be performed manually by editing system configuration files. The installer makes these tasks easier by enabling you to perform the configurations by responding to prompts from the installer.

**Note** As a best practice, you should complete basic HyperStore installation first and confirm that things are working properly (by running the installer's Validation Tests, under the "Cluster Management" menu) before you consider using the advanced configuration options to make changes such as customizing port assignments or implementing SSL for the S3 Service.

To access the advanced configuration options, on your Puppet master node change into your [installation staging directory](#) 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 Puppet 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.

At the installer main menu's Choice prompt enter **4** for Advanced Configuration Options.

```
Cloudian HyperStore(R) 7.2 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: █
```

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, S3-Website, Admin, or CMC endpoints
d ) Configure diagnostic data collection options
e ) Configure SSL for S3
f ) Configure SSL for CMC
g ) Configure SSL for IAM
h ) Remove existing Puppet SSL certificates
i ) Start or stop Puppet daemon
j ) Remove Puppet access lock
k ) Enable or disable DNSMASQ
l ) Configure Performance Parameters on Nodes
m ) Disable the root password
r ) Exclude host(s) from configuration push and service restarts
s ) Configure Firewall
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 Redis QoS master role or Redis Credentials master role — from one of the hosts in your cluster to another. For the complete procedures, see **"Change Node Role**

**Assignments"** (page 457).

#### b) Change S3, Admin or CMC ports

This lets you interactively change listening ports for the S3, Admin, and CMC services. For instructions see **"Changing S3, Admin, or CMC Listening Ports"** (page 599).

#### c) Change S3, Admin, CMC, or IAM endpoints

This lets you interactively change the S3 service endpoint, S3 static website endpoint, Admin service endpoint, CMC endpoint, or IAM service endpoint. For instructions see **"Changing S3, Admin, CMC, or IAM Service Endpoints"** (page 600).

#### 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. The final prompt will ask whether you want to save your changes -- type *yes* to do so.
2. Go to the installer's main menu again and choose "Cluster Management" → "Push Configuration Settings to Cluster" and follow the prompts.
3. Go to the "Cluster Management" menu again, choose "Manage Services", and restart the S3 Service.

#### e, f, g) Configure SSL for S3, CMC, or IAM

These options are for managing HTTPS and the associated certificate keystores, for the S3, CMC, and IAM services. For instructions see **"HTTPS Support (TLS/SSL)"** (page 114).

#### h) Remove existing Puppet SSL certificates

This is a troubleshooting measure in the event of Puppet connectivity problems, as described in **Installation Troubleshooting**.

This operation is specifically in regard to Puppet certificates and has nothing to do with SSL for the S3 Service.

#### i) Start or stop Puppet daemon

By default, after HyperStore installation the Puppet master and agent daemons are left running, and every 10 minutes the agents check the master to see if there are updated HyperStore configuration settings to download to the agent nodes. Alternatively you can use this option from the Advanced menu to stop the Puppet agent daemons.

Note that if you don't leave the Puppet agent daemons running, you must remember to trigger a one-time Puppet sync-up each time you make a HyperStore configuration change on the Puppet master. By contrast, if you leave the Puppet agent daemons running, a sync-up will happen automatically every 10 minutes even if you don't specifically trigger a sync-up after making a configuration change.

If you ever want to check on the current status of your Puppet master and agent daemons, you can do so by choosing "Cluster Management" from the installer's main menu; then choose "Manage Services"; then in the Service Management sub-menu choose "Puppet service (status only)".

### j) Remove Puppet access lock

When Puppet is performing a sync-up, the system temporarily locks Puppet access (so that no additional Puppet instances are launched by other operators or systems). If the sync-up operation is terminated unexpectedly — such as by a <CTRL>-c command, or a Puppet master node shutdown — the temporary lock may fail to release. This unreleased lock would prevent any subsequent sync-ups from being implemented.

You can clear the lock by running the "Remove Puppet access lock" operation.

Afterwards, to confirm that the lock is cleared you can check to make sure that no `/tmp/cloudian.installer.lock` directory exists on the Puppet master node.

### k) Enable or disable DNSMASQ

This menu option is for either of these circumstances:

- When you ran the `cloudianInstall.sh` script to install your HyperStore system, you had the script automatically install and configure [dnsmasq](#) to perform DNS resolution for HyperStore service domains. (That is, you used the `configure-dnsmasq` option when you launched the install script.) However, now you want to use your own DNS solution for resolving HyperStore domains, and you've completed your DNS configuration as described in **DNS Set-Up**. You can use the installer's "Enable or disable DNSMASQ" menu option to disable `dnsmasq`. This stops `dnsmasq` on all HyperStore nodes and disables `dnsmasq` by configuration.
- When you ran the `cloudianInstall.sh` script to install your HyperStore system, you did **not** have it install `dnsmasq` (the default installer behavior is not to install `dnsmasq`). However, now you want to use `dnsmasq` for HyperStore domain resolution. With the installer's "Enable or disable DNSMASQ" menu option you can enable `dnsmasq`:
  1. After selecting this option from the Advanced Configuration Options menu, follow the prompts to choose to enable `dnsmasq`.
  2. Go to the installer's main menu again and choose "Cluster Management" → "Push Configuration Settings to Cluster" and follow the prompts.
  3. Go to the "Cluster Management" menu again, choose "Manage Services", and restart the DNSMASQ service.

### 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 instructions on using the option see **"Tuning HyperStore Performance Parameters"** (page 602).

### m) Disable the root password

For information on this option see **"Enabling the HSH and Managing HSH Users"** (page 90).

### 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 installer-driven configuration pushes and service restarts. 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. With a node down, the installer's "Cluster Management" → "Push Configuration Settings to

Cluster" function will fail if you try to push to the whole cluster. So before doing a push, use the "Advanced Configuration Options" → "Exclude host(s) from configuration push and service restarts" to specify the down node. Then, when you subsequently do a configuration push to the cluster, the installer will automatically exclude that node and the push to the cluster will succeed.

The excluded host will also be excluded when you use the installer's "Cluster Management" → "Manage Services" menu to stop, start, or restart particular services in the cluster (such as the S3 Service or the Cassandra Service).

**Note** This "exclude from configuration push and service restarts" status is different than "maintenance mode" (see **"Start Maintenance Mode"** (page 329)). 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. By contrast, when you put a node into "maintenance mode" the system stops sending S3 requests to the node and stops collecting alerts from the node.

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 installer-managed 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 instructions see **"HyperStore Firewall"** (page 100).

## t) Configure 'force' behaviour

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 Command Line Options" in the Reference section of the *HyperStore Installation Guide*.

## 9.3. Pushing Configuration File Edits to the Cluster and Restarting Services

*Subjects covered in this section:*

- **"Puppet Overview"** (page 506)
- **"Installation Staging Directory"** (page 507)
- **"Using the Installer to Push Configuration Changes and Restart Services"** (page 507)
- **"Option for Triggering a Puppet Sync-Up from the Command Line"** (page 509)
- **"Excluding a Down Node from an Installer-Driven Configuration Push"** (page 509)
- **"Automatic Puppet Sync-Up on an Interval"** (page 510)

### 9.3.1. Puppet Overview

During HyperStore installation, the open source version of [Puppet](#) is automatically installed and set up. The Puppet framework consists of:

- A **Puppet master node** on which all the HyperStore configuration templates reside. This is one of your HyperStore nodes -- specifically, the node on which you ran the HyperStore installation script.
- A **Puppet agent on every node** in your HyperStore system (including the node on which the master is running).

The Puppet system enables you to edit HyperStore configuration templates in one location — on the Puppet master node — and have those changes propagate to all the nodes in your HyperStore cluster, even across multiple data centers and multiple service regions. There are two options for implementing Puppet sync-up: you can use the HyperStore installer to trigger an immediate sync-up, or you can wait for an automatic sync-up which by default occurs on a 10 minute interval. With either approach, after the sync-up **you must restart the affected service(s) to apply the configuration changes**.

**IMPORTANT !** Do not directly edit configuration files on individual HyperStore nodes. If you make edits on an individual node, those local changes will be overwritten when the local Puppet agent does its next sync-up with the Puppet master.

**Note** To ensure high availability of the Puppet master role, HyperStore automatically sets up a backup Puppet master node and supports a method for [manually failing over the master role](#) in the event of problems with the primary Puppet master node.

### 9.3.2. Installation Staging Directory

During installation or upgrade of your HyperStore system, when you unpack the HyperStore product package on your Puppet master node an installation staging directory is created. For HyperStore version 7.2.3, the installation staging directory is:

```
/opt/cloudian-staging/7.2.3
```

If you forget the location of your staging directory, you can find it displayed in the CMC's [Cluster Information](#) page toward the bottom of the "Service Information" section.

Among the important files in your installation staging directory is the HyperStore installation script *cloudianInstall.sh* -- also known as the HyperStore installer -- which you can use for a variety of purposes including pushing configuration changes out to the system and restarting services.

### 9.3.3. Using the Installer to Push Configuration Changes and Restart Services

After you've edited configuration files on the Puppet master you can use the HyperStore installer to trigger a Puppet sync-up restart the affected service(s):

1. After logging into the Puppet master node as *root*, change into your [installation staging directory](#). Once in the staging directory, launch the HyperStore installer:

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

This displays the installer's main menu:

```
Cloudian HyperStore(R) 7.2 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: █
```

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

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Puppet 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.

2. Enter "2" for Cluster Management. This displays the Cluster Management menu.

```
Cluster Management
-----

a ) Review Cluster Configuration
b ) Push Configuration Settings to Cluster
c ) Manage Services
d ) Run Validation Tests
x ) Return to Main Menu

Choice: █
```

3. Enter "b" for Push Configuration Settings to Cluster. You will then be prompted to list the hosts on which you want the Puppet agents to sync up with the Puppet master. The default is all your HyperStore hosts; for this you can just press enter at the prompt. In a multi-region system you are also given the option to sync-up only the agents in a particular region.
4. After the Puppet run completes for all the Puppet agents (and a success message displays on the console), **restart the affected service(s) to apply your configuration change**:
  - a. From the Cluster Management menu, enter "c" for Manage Services. This displays the Service Management menu.

```
Service Management
-----

0 ) All services
1 ) Redis Credentials
2 ) Redis QOS
3 ) Cassandra
4 ) HyperStore service
5 ) S3 service
6 ) Redis Monitor
7 ) Cloudian Agent
8 ) DNSMASQ
9 ) Cloudian Management Console (CMC)
10) IAM
11) SQS
P ) Puppet service (status only)
X ) Quit

You can execute the following list of commands:
start,stop,status,restart,version,node-start,node-stop

Select a service to manage: █
```

- b. From the Service Management menu, enter the number for the service to restart. The service to restart will depend on which configuration setting(s) you edited. For example:

File in Which You Edited Setting(s)	Service to Restart
<i>hyperstore-server.properties.erb</i>	HyperStore Service
<i>mts-ui.properties.erb</i>	CMC
<i>mts.properties.erb</i>	Typically the S3 Service (but instead Cassandra in a small number of cases; see <a href="#">mts.properties.erb</a> )
<i>common.csv</i>	Depends on the specific setting(s); see <a href="#">common.csv</a>

- c. After entering the service to manage, enter "restart". Watch the console for messages indicating a successful stop and restart of the service. Note that the service restart occurs on each node on which the service is running.

### 9.3.4. Option for Triggering a Puppet Sync-Up from the Command Line

HyperStore supports an alternative means of triggering a Puppet sync-up, from the command line. To use this method you would run the install script like this:

```
# ./cloudianInstall.sh runpuppet="[<region>,<host>,<host>,...]"
```

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

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Puppet master node you can run the following command to trigger a Puppet sync-up:

```
$ hspkg install runpuppet="[<region>,<host>,<host>,...]"
```

Here are some examples for how to specify the *runpuppet* option:

- *runpuppet=""* — Sync-up all the agents in your whole HyperStore system.
- *runpuppet="region1"* — Sync-up only all the agents in region1.
- *runpuppet="region1,host1"* — Sync-up only host1 in region1.
- *runpuppet="region1,host1,host2"* — Sync-up only host1 and host2 in region1.

**Note** If you use this method, you would still need to subsequently launch the installer in the normal way and then **restart the affected service(s)** as described in Step 4 above, in order to apply your changes.

### 9.3.5. Excluding a Down Node from an Installer-Driven Configuration Push

If you use the installer to push a configuration change out to your cluster (by triggering a Puppet sync-up throughout the cluster) and the installer detects that a node is down or unreachable, the whole configuration push operation will fail. Therefore, if you want to make a system configuration change at a time when a node is down or unreachable you need to configure the installer to exclude that node from the cluster configuration push. For instructions on place a node into "excluded" status (and how to take a node out of this status), see **"r) Exclude host(s) from configuration push and service restarts"** (page 504).

**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

failover 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.

### 9.3.6. Automatic Puppet Sync-Up on an Interval

If you leave the Puppet master and Puppet agents running as background daemons (as is the default behavior after HyperStore installation), the Puppet agents on all of your up HyperStore nodes will automatically check every 10 minutes to see if changes have been made to the HyperStore configuration templates on the Puppet master node. If configuration changes have been made, each Puppet agent will download those changes to its local host machine.

Note however that this automatic Puppet sync-up **does not apply the configuration changes to the currently running services**. Applying the configuration changes requires a **service restart**.

To apply your changes, first wait long enough to be sure that the automatic Puppet sync-up has occurred (again the default is every 10 minutes). Then, restart the affected service(s) by logging into your Puppet master node, changing into the installation staging directory, launching the installer, then going to the Cluster Management menu and restarting the affected service(s) as described in Step 4 of the procedure above.

## 9.4. Using the HSH to Manage Configuration Files

If you are using the [HyperStore Shell \(HSH\)](#) to manage your HyperStore nodes, the HSH supports commands for viewing and editing HyperStore configuration files. To use the HSH to view or edit HyperStore configuration files, first log into the Puppet master node (via SSH) as an HSH user. Upon successful login the HSH prompt will appear as follows:

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

For example:

```
sa_admin@hyperstore1$
```

**Note** To use the HSH to manage configuration files you must be an [HSH Trusted user](#).

**To view the list of configuration files that you can view and edit using the HSH:**

To see the complete list of configuration files that you can view and edit in the HSH run this command:

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

The list includes all the files covered in the "HyperStore Configuration Files" section of this documentation as well as a few additional system configuration files.

**To view a configuration file using the HSH:**

```
$ hspkg config <filename>
```

Specify just the configuration file name (such as *common.csv*), not the full path to the file.

In the background this invokes the Linux command `/less` to display the configuration file. Therefore you can use the standard keystrokes supported by `/less` to navigate the display; for example:

- **f** key or Space bar -- Page down
- **b** key -- Page up
- Down arrow key or Enter -- Go down one line
- Up arrow key -- Go up one line
- `<n>f` key or `<n>Space bar` -- Go down `<n>` number of lines
- `<n>b` key -- Go up `<n>` number of lines
- `/string` Enter-- Search down for the specified string
- `?string` Enter -- Search up for the specified string
- **q** key -- Quit the file display and return to the HSH prompt

**To edit a configuration file using the HSH:**

```
$ hspkg config -e <filename>      (or $ hspkg config --edit <filename>)
```

Specify just the configuration file name (such as `common.csv`), not the full path to the file.

In the background this invokes the Linux text editor `vi` to display and modify the configuration file. Therefore you can use the standard keystrokes supported by `vi` to make and save changes to the file; for example:

- Up or down arrow keys -- Move cursor up or down one line
- Left or right arrow keys -- Move cursor left or right one character
- **i** key -- Start insert mode (to actually make edits to file)
- Escape key -- End insert mode and return to command mode (so that in command mode you can save or discard your changes)
- **:w** key combination -- In command mode, save changes and keep the file open so you can keep working on it
- **:wq** key combination -- In command mode, save changes, close the file, and return to the HSH prompt
- **:q!** key combination -- In command mode, discard changes, close the file, and return to the HSH prompt
- **:q** key combination -- In command mode, close the file and return to the HSH prompt (appropriate only if you made no changes to the file)

**Note** For more information about using the `vi` text editor, see any reputable online source.

If you made and saved a change to a configuration file, to apply the change you must use the installer to push the change out to the cluster and restart the relevant service(s). From the HSH you can launch the installer as follows:

```
$ hspkg install
```

For more information about using the installer to push your configuration change and restart services, see **"Using the Installer to Push Configuration Changes and Restart Services"** (page 507).

## 9.5. HyperStore Configuration Files

The main configuration file for your HyperStore system is [common.csv](#). Under typical circumstances that is the only configuration file that you might edit.

On rare occasions you might -- in consultation with Cloudian Support -- edit the [mts.properties.erb](#) file, the [hyperstore-server.properties.erb](#) file, or the [mts-ui.properties.erb](#) file.

**There is no requirement to manually edit any HyperStore configuration file.** Configuration customizations that are required in order to tailor HyperStore to your environment are implemented automatically by the installation script during the initial installation of your HyperStore system and during cluster expansions or contractions.

**Note** The configuration settings that operators would most commonly want to adjust during the operation of a HyperStore system are editable through the CMC's [Configuration Settings](#) page. The only reason to manually edit configuration files is for less-frequently-used settings that are not in the CMC's [Configuration Settings](#) page.

### 9.5.1. common.csv

The *common.csv* file is the main HyperStore configuration file and under typical circumstances this is the only configuration file that you may want to edit. On the Puppet master the path to the file is:

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

The settings in *common.csv* are grouped according to which HyperStore component they configure. The documentation that follows describes the settings in each section of *common.csv*, in the order in which they appear in the file.

*If you are using the HyperStore Shell*

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Puppet master node you can edit this configuration file with this command:

```
$ hspkg config -e common.csv
```

Specify just the configuration file name, not the full path to the file.

In the background this invokes the Linux text editor *vi* to display and modify the configuration file. Therefore you can use the [standard keystrokes supported by vi](#) to make and save changes to the file.

**IMPORTANT !** If you make any edits to *common.csv*, be sure to push your edits to the cluster and restart the affected services to apply your changes. For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

**Note** The HyperStore system's interactive installation script *cloudianInstall.sh* writes to this file. All *common.csv* settings that **require** environment-specific customization are automatically pre-configured by the install script, based on information that you provided during the installation process. Making any further customizations to this configuration file is optional.

## GENERAL Section

*release\_version*

Current HyperStore release version.

Default = 7.2.3

Do not edit.

*cloudian\_license\_file\_name*

Name of your Clodian license file.

Default = Set during installation based on operator input.

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.

*default\_region*

The default [service region](#) for your S3 service. Must be one of the regions listed for the regions setting. In a multi-region HyperStore system, the default region plays several roles in the context of S3 storage bucket LocationConstraint functionality. For example:

- For PUT Bucket requests that lack a CreateBucketConfiguration element in the request body, the bucket will be created in the default region.
- For PUT Bucket requests that do include a CreateBucketConfiguration element (with LocationConstraint attribute), the PUT requests typically resolve to the default region, and S3 Service nodes in the default region then initiate the process of creating the bucket in the requested region.

If your HyperStore system has only one service region, make that region the default region.

Default = Set during installation based on operator input.

Do not change this setting after your system is installed and running. If for some reason you need to change which region is your default region, please consult with Clodian Support.

*regions*

Comma-separated list of [service regions](#) within your HyperStore system. Region names must be lower case with no dots, dashes, underscores, or spaces. Even if you have only one region, you must give it a name and specify it here.

Default = Set during installation based on operator input.

*javahome*

Home directory of Java on your HyperStore nodes.

Default = Path to OpenJDK (set automatically during installation)

*java\_minimum\_stack\_size*

Memory stack size to use for the HyperStore system's Java-based services (Cassandra, S3 Service, HyperStore Service, Admin Service)

Default = 256k

**Note** Optionally you can override this stack size value on a per-server-type basis by adding any of the following settings to the configuration file and assigning them a value:

*cassandra\_stack\_size*

*cloudian\_s3\_stack\_size*

*cloudian\_hss\_stack\_size*  
*cloudian\_admin\_stack\_size*

*installation\_root\_directory*

Directory in which HyperStore service packages will be physically installed.

Default = /opt/cloudian-packages

*run\_root\_directory*

Root directory in which HyperStore services will be run. Links will be created from this directory to the physical installation directory.

Default = /opt

*pid\_root\_directory*

Root directory in which HyperStore service PID (process ID) files will be stored. A */cloudian* sub-directory will be created under this root directory, and the PID files will be stored in that sub-directory.

Default = /var/run

*cloudian\_user*

User information for the user as which to run Cloudian HyperStore services, in format *<user\_name>,<group\_name>,<optional\_numeric\_UID>,<login\_shell>*. If you want this user to be something other than the default, edit this setting and also the *cloudian\_runuser* setting.

Default = "cloudian,cloudian,/,bin/bash"

*cloudian\_runuser*

User name of the user as which to run Cloudian HyperStore services. This must match the first field from the *cloudian\_user* setting. If you edit the *cloudian\_runuser* setting you must also edit the first field from the *cloudian\_user* setting.

Default = cloudian

*user\_bin\_directory*

Directory in which certain user-invokable scripts are stored.

Default = /usr/local/bin

*user\_home\_directory*

Directory under which to create the home directory of the HyperStore services runtime user. The system will append the *cloudian\_runuser* value to the *user\_home\_directory* value to get the full home directory path. For example, with "cloudian" as the *cloudian\_runuser* and "/export/home" as the *user\_home\_directory*, the Cloudian user's home directory is "/export/home/cloudian".

Default = /export/home

*cloudian\_userid\_length*

Maximum number of characters allowed in a HyperStore user ID. The highest value you can set this to is 256.

**Note** This maximum applies also to the user full name. 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.

Default = 64

To apply change, after Puppet propagation restart S3 Service and CMC

*service\_starts\_on\_boot*

Whether to have HyperStore services start on host reboot, true or false.

Default = true

*cloudian\_log\_directory*

Directory into which to write application logs for the S3 Service, Admin Service, HyperStore Service, Redis Monitor, Cloudian performance monitoring Agent, and Cloudian performance monitoring Data Collector.

Default = /var/log/cloudian

*cleanup\_directories\_byage\_withmatch*

The *cleanup\_directories\_byage\_\** settings configure Puppet to automatically delete certain files from your HyperStore nodes after the files reach a certain age. The *cleanup\_directories\_byage\_withmatch* setting is a comma-separated list of directories in which to look for such files.

This feature works only if you leave the Puppet daemons running on your HyperStore nodes (which is the default behavior), or if you regularly perform a Puppet push.

To apply change, do a Puppet push. No service restart is necessary.

Default = "/var/log/cloudian,/tmp,/var/log/puppetserver"

*cleanup\_directories\_byage\_withmatch\_timelimit*

The *cleanup\_directories\_byage\_\** settings configure Puppet to automatically delete certain files from your HyperStore nodes after the files reach a certain age. The *cleanup\_directories\_byage\_withmatch\_timelimit* setting specifies the age at which such files will be deleted (based on time elapsed since file creation). The age can be specified as *<x>m* or *<x>h* or *<x>d* where *<x>* is a number of minutes, hours, or days.

This feature works only if you leave the Puppet daemons running on your HyperStore nodes (which is the default behavior), or if you regularly perform a Puppet push.

To apply change, do a Puppet push. No service restart is necessary.

Default = 15d

*cleanup\_directories\_byage\_matches*

The *cleanup\_directories\_byage\_\** settings configure Puppet to automatically delete certain files from your HyperStore nodes after the files reach a certain age. The *cleanup\_directories\_byage\_matches* setting specifies the file types to delete.

This feature works only if you leave the Puppet daemons running on your HyperStore nodes (which is the default behavior), or if you regularly perform a Puppet push.

To apply change, do a Puppet push. No service restart is necessary.

Default = "diagnostics\*.gz,diagnostics\*.tgz,jna\*.tmp,liblz4-java\*.so,snappy-\*.so,\*.cloudian-bak,cloudian\_system\_info\*.tar.gz,puppetserver\*.log.zip"

#### *cleanup\_sysinfo\_logs\_timelimit*

Retention period for Node Diagnostics packages. After a package reaches this age, Puppet will automatically delete the package.

A Node Diagnostics package is created under a */var/log/cloudian/cloudian\_sysinfo* directory on a node if you use the Collect Diagnostics feature for that node. For general information on Node Diagnostics see **"Smart Support and Diagnostics Feature Overview"** (page 190).

This cleanup feature works only if you leave the Puppet daemons running on your HyperStore nodes (which is the default behavior).

Specify this value as a number of days, hours, or minutes, with the value formatted as *<n>d* or *<n>h* or *<n>m* respectively (such as 15d or 12h or 30m).

Default = 15d

**Note** When you use the CMC to [collect diagnostics on a node](#), the UI gives you the option to have the diagnostics package automatically uploaded to Cloudian Support, and also the option to have the system delete the package immediately after it's been successfully uploaded to Cloudian Support. The retention period set by *cleanup\_sysinfo\_logs\_timelimit* comes into play only if you do not use the "upload and then immediately delete" options.

#### *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
```

If the CMC (or any other S3 client applications that you are using) uses 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](#).

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 supports both path style access and virtual host style access -- regardless of which method is being used by S3 clients.

Default = true

To apply change, after Puppet propagation restart CMC

*fips\_enabled*

If you set this to *true*, then on each HyperStore node, *sshd* (the OpenSSH server daemon) will support **only FIPS-approved ciphers**. This prevents SSH clients from connecting with weaker, non-FIPS-approved ciphers. For more information regarding FIPS, see **"FIPS Support"** (page 113).

If this setting is left at its default value of *false*, then *sshd* will support its full, default set of ciphers -- which includes some ciphers that are not FIPS-approved as well as ciphers than are FIPS-approved.

Default = false

To apply change, after Puppet propagation restart the S3 Service and the CMC

## HyperStore SERVICE Section

*hyperstore\_data\_directory*

A quote-enclosed, comma-separated list of mount points to use for S3 object storage. In a production environment, use dedicated disks for S3 object storage. Do not use the same disk(s) that are storing the OS and Cassandra.

The system will automatically assign virtual nodes (vNodes) to each of your S3 data mount points, in a manner that allocates an approximately equal **total token range** to each mount point. For more information about HyperStore vNodes see **"How vNodes Work"** (page 42).

Do not change the *hyperstore\_data\_directory* setting once your cluster is operational.

Do not use symbolic links when specifying your mount points for the *hyperstore\_data\_directory* 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 host has multiple disks. For hosts with only one disk, default is /var/lib/cloudian

*hyperstore\_listen\_ip*

The IP interface on which each HyperStore Service node listens for data operations requests from clients. This setting must match the *cassandra\_listen\_address* setting.

**Specify this as an IP address alias.** Puppet will use the alias to determine the actual IP address for each node.

Options are %{:cloudian\_ipaddress}, %{:ipaddress\_eth#}, %{:ipaddress\_lo}, or %{:ipaddress\_bind#}

Default = %{:cloudian\_ipaddress}

To apply change, after Puppet propagation restart HyperStore Service

*hyperstore\_timeout*

For the S3 Service's connections to the HyperStore Service, the transaction completion timeout (session timeout) in milliseconds.

Default = 10000

To apply change, after Puppet propagation restart S3 Service

For a diagram showing the place of this timeout within the S3 request processing flow, see the description of *mts.properties.erb*: **"cassandra.cluster.CassandraThriftSocketTimeout"** (page 554).

*hyperstore\_connection\_timeout*

For the S3 Service's connections to the HyperStore Service, the connection establishment timeout in milliseconds.

Default = 10000

To apply change, after Puppet propagation restart S3 Service

For a diagram showing the place of this timeout within the S3 request processing flow, see the description of *mts.properties.erb*: "**cassandra.cluster.CassandraThriftSocketTimeout**" (page 554).

*hyperstore.maxthreads.repair*

Maximum number of simultaneous client threads for one S3 Service node to use on HyperStore File System data repairs automatically performed during read operations. For more information on the "repair on read" mechanism see "**Automated Data Repair Feature Overview**" (page 150).

Default = 50

*hyperstore\_jetty\_minThreads*

Each HyperStore Service node maintains a thread pool to process incoming HTTP requests from clients (S3 Service nodes). Idle threads are terminated if not used within a timeout period — unless the number of threads in the pool is down to the required minimum pool size, in which case the idle threads are kept.

The *hyperstore\_jetty\_minThreads* parameter sets the minimum number of threads to keep in the thread pool.

Default = 100

*auto\_repair\_computedigest\_run\_number*

This property configures the [scheduled auto-repair feature](#) such that every *Nth* run of [hsstool repair](#) (for replicated object data) and [hsstool repairec](#) (for erasure coded object data) will use the "-computedigest" option in order to detect and repair any data corruption on disk ("bit rot"). For example, if this property 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 *repairec* will use the "-computedigest" option.

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 *auto\_repair\_computedigest\_run\_number* 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 *repairec* is 29 days. With erasure coded data repair, running *hsstool repairec* 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 *auto\_repair\_computedigest\_run\_number* to 3, then for a given data center the automatically triggered *repairec* runs would be implemented like this:

- Day 0: *repairec* without "-computedigest"
- Day 29: *repairec* without "-computedigest"
- Day 58: *repairec* with "-computedigest"
- Day 87: *repairec* without "-computedigest"
- Day 116: *repairec* without "-computedigest"
- Day 145: *repairec* with "-computedigest"
- etc

Setting `auto_repair_computedigest_run_number` to 1 would result in all auto-repair runs using "-computedigest".

By default `auto_repair_computedigest_run_number` 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.

**Note** The `auto_repair_computedigest_run_number` setting has no impact on *hsstool repair* or *hsstool repaierc* runs that you manually execute on a node. The setting only impacts the auto-repair feature.

#### *hyperstore.maxthreads.write*

Maximum number of simultaneous client threads for one S3 Service node to use on writes to the HyperStore File System.

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

#### *hyperstore.maxthreads.read*

Maximum number of simultaneous client threads for one S3 Service node to use on reads of the HyperStore File System.

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

#### *hyperstore\_maxperrouteconnections*

The maximum allowed number of concurrently open connections between each S3 Service node and each HyperStore Service node. This allows for limiting the traffic load between each front-end S3 Service node (as it processes incoming requests from S3 clients) and any single HyperStore Service node.

Note that each of your S3 Service nodes has its own pool of connections to the HyperStore storage layer, so the total possible connections from the S3 Service as a whole to a single HyperStore Service node would be the number of S3 Service nodes multiplied by the value of "Max Connections from One S3 Service Node to One HyperStore Service Node".

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

*hyperstore\_maxtotalconnections*

The maximum allowed number of concurrently open connections between each S3 Service node and all HyperStore Service nodes, combined. This allows for limiting the traffic load between each front-end S3 Service node (as it processes incoming requests from S3 clients) and the whole back-end HyperStore storage layer.

Note that each of your S3 Service nodes has its own pool of connections to the HyperStore storage layer, so the total possible connections from the front-end S3 Service as a whole to the back-end HyperStore storage layer as a whole would be the number of S3 Service nodes multiplied by the value of "Max Connections from One S3 Service Node to All HyperStore Service Nodes".

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

*hyperstore\_jetty\_maxThreads*

Each HyperStore Service node maintains a thread pool to process incoming HTTP requests from clients (S3 Service nodes). When there is a request to be serviced, a free thread from the pool is used and then returned to the pool afterward. If a thread is needed for a job but no thread is free, a new thread is created and added to the pool — unless the maximum allowed number of threads in the pool has been reached, in which case queued jobs must wait for a thread to become free.

The *hyperstore\_jetty\_maxThreads* parameter sets the maximum number of threads to allow in the thread pool.

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

*hyperstore\_messaging\_service\_threadpool*

Maximum size of the thread pool used by the HyperStore inter-node messaging service. When there is a new task and there are no idle threads available in the thread pool:

- If fewer than this many threads are in the thread pool, the thread pool executor will create a new thread.
- If this many or more threads are in the thread pool, the thread pool executor will queue the new task.

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

*hyperstore\_repair\_session\_threadpool*

Maximum size of the thread pool used for [hsstool repair](#) or [hsstool repairec](#) operations. When there is a new task and there are no idle threads available in the thread pool:

- If fewer than this many threads are in the thread pool, the thread pool executor will create a new thread.
- If this many or more threads are in the thread pool, the thread pool executor will queue the new task.

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

*hyperstore\_repair\_digest\_index\_threadpool*

Maximum size of 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 task and there are no idle threads available in the thread pool:

- If fewer than this many threads are in the thread pool, the thread pool executor will create a new thread.
- If this many or more threads are in the thread pool, the thread pool executor will queue the new task.

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

#### *hyperstore\_rangerepair\_threadpool*

Maximum size of the thread pool used for running multiple range repair tasks in parallel during [hsstool repair](#). When there is a new task and there are no idle threads available in the thread pool:

- If fewer than this many threads are in the thread pool, the thread pool executor will create a new thread.
- If this many or more threads are in the thread pool, the thread pool executor will queue the new task.

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

#### *hyperstore\_stream\_outbound\_threadpool*

Maximum size of the thread pool used for streaming files from one HyperStore node to another during Merkle Tree based [hsstool repair](#). When there is a new task and there are no idle threads available in the thread pool:

- If fewer than this many threads are in the thread pool, the thread pool executor will create a new thread.
- If this many or more threads are in the thread pool, the thread pool executor will queue the new task.

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

#### *hyperstore\_downloadrange\_session\_threadpool*

Maximum size of the thread pool used for range download sessions conducted by a HyperStore Service node. When there is a new task and there are no idle threads available in the thread pool:

- If fewer than this many threads are in the thread pool, the thread pool executor will create a new thread.
- If this many or more threads are in the thread pool, the thread pool executor will queue the new task.

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

#### *hyperstore\_uploadrange\_session\_threadpool*

Maximum size of the thread pool used for range upload sessions conducted by a HyperStore Service node. When there is a new task and there are no idle threads available in the thread pool:

- If fewer than this many threads are in the thread pool, the thread pool executor will create a new thread.
- If this many or more threads are in the thread pool, the thread pool executor will queue the new task.

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

#### *hyperstore\_decommission\_threadpool*

Maximum size of the thread pool used for uploading files away from a node that is being decommissioned. When there is a new task and there are no idle threads available in the thread pool:

- If fewer than this many threads are in the thread pool, the thread pool executor will create a new thread.
- If this many or more threads are in the thread pool, the thread pool executor will queue the new task.

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

*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 rather should be deleted.

The maximum number of "jobs" running in parallel is set by "**cleanupjobs.threadpool.corepoolsize**" (page 549) in *hyperstore-server.properties.erb*.

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

*hyperstore\_auto\_repair\_threadpool*

Maximum size of the thread pool used by the HyperStore auto-repair feature. When there is an auto-repair to kick off and there are no idle threads available in the thread pool:

- If fewer than this many threads are in the thread pool, the thread pool executor will create a new thread.
- If this many or more threads are in the thread pool, the thread pool executor will queue the new task.

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

**Note** For more information about the auto-repair feature see "**Automated Data Repair Feature Overview**" (page 150).

*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 size of the thread pool per node.

Default = 50

Reloadable via JMX (HyperStore Service's JMX port 19082; MBean attribute = com.gemini.cloudian.hybrid.server → FileRepairService → Attributes → RepairECSessionScanThreadPoolCorePoolSize)

*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 parameter sets the maximum size of the thread pool per node.

Default = 30

Reloadable via JMX (HyperStore Service's JMX port 19082; MBean attribute = com.gemini.cloudian.hybrid.server → FileRepairService → Attributes → RepairECDigestRequestThreadPoolFixedPoolSize)

*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 parameter sets the maximum size of the thread pool per node.

Default = 60

Reloadable via JMX (HyperStore Service's JMX port 19082; MBean attribute = com.gemini.cloudian.hybrid.server → FileRepairService → Attributes → RepairECThreadPoolCorePoolSize)

*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 size of the thread pool per RocksDB instance.

Default = 30

Reloadable via JMX (HyperStore Service's JMX port 19082; MBean attribute = com.gemini.cloudian.hybrid.server → FileRepairService → Attributes → RepairECRock-sDBScanThreadPoolFixedPoolSize)

*hyperstore\_disk\_check\_interval*

The interval (in minutes) at which each HyperStore Service node will run a check to see if there is significant imbalance in disk usage on the node. If an imbalance is found in excess of that configured by *disk.balance.delta*, 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. For more information see "**Automated Disk Management Feature Overview**" (page 157).

Default = 4320 (72 hours)

To apply change, after Puppet propagation restart the HyperStore Service

**Note** This feature applies only to HyperStore data disks (on which are stored S3 object data). It does not apply to disks that are storing only the OS and Cassandra.

## S3/ADMIN SERVICES Section

*phonehome\_proxy\_host*

If you want HyperStore to use a local forward proxy when the Smart Support (or "Phone Home") 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 apply change, after Puppet propagation restart the S3 Service

**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 [sysinfo.proxy.\\*](#) settings in *mts.properties.erb* to separately configure proxy information for Node Diagnostics sending.

For more background information on these features see "**Smart Support and Diagnostics Feature Overview**" (page 190).

#### *phonehome\_proxy\_port*

If you want HyperStore to use a local forward proxy when the Smart Support (or "Phone Home") feature sends daily system diagnostics packages to Clouidian Support, use this setting to specify the proxy's port number.

Default = empty

To apply change, after Puppet propagation restart the S3 Service

#### *phonehome\_proxy\_username*

If you want HyperStore to use a local forward proxy when the Smart Support (or "Phone Home") feature sends daily system diagnostics packages to Clouidian 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

#### *phonehome\_proxy\_password*

If you want HyperStore to use a local forward proxy when the Smart Support (or "Phone Home") feature sends daily system diagnostics packages to Clouidian Support, use this setting to specify the password that HyperStore should use when connecting to the proxy (if are username and password are required by the proxy).

Default = empty

#### *phonehome\_uri*

S3 URI to which to upload system-wide diagnostics data each day. By default this is the S3 URI for Clouidian Support.

If you set this to a different S3 destination, include the HTTP or HTTPS protocol part of the URI (*http://* or *https://*).

Default = `https://s3-support.clouidian.com:443`

To apply change, after Puppet propagation restart the S3 Service

**Note** If you set *phonehome\_uri* to a URI for your own HyperStore S3 Service (rather than the Clouidian 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 Clouidian Support for guidance on modifying the phone home launch script. For information on HTTPS set-up for the S3 Service, see "**HTTPS Support (TLS/SSL)**" (page 114).

#### *phonehome\_bucket*

- If you leave *phonehome\_uri* at its default value -- which is Clouidian Support S3 URI -- you can leave the *phonehome\_bucket*, *phonehome\_access\_key*, and *phonehome\_secret\_key* properties empty. The Smart Support feature will automatically extract the Clouidian Support S3 bucket name and security credentials from your encrypted HyperStore license file.

- If you set *phonehome\_uri* to an S3 URI other than the Clodian Support URI, set the *phonehome\_bucket*, *phonehome\_access\_key*, and *phonehome\_secret\_key* properties to the destination bucket name and the applicable S3 access key and secret key.

Default = empty

To apply change, after Puppet propagation restart the S3 Service

*phonehome\_access\_key*

See *phonehome\_bucket*.

Default = empty

To apply change, after Puppet propagation restart the S3 Service

*phonehome\_secret\_key*

See *phonehome\_bucket*.

Default = empty

To apply change, after Puppet propagation restart the S3 Service

*phonehome\_gdpr*

To have the Smart Support feature comply with the European Union's General Data Protection Regulation (GDPR) requirements, set this to true. The impact of this setting is as follows:

- With *phonehome\_gdpr* set to true, the Smart Support feature will remove user IDs and client IP addresses from the **copies** of the S3 request log ([cloudian-request-info.log](#)) and S3 application log ([cloudian-s3.log](#)) that are uploaded to Clodian Support each day. However the user IDs and client IP addresses will remain in the **original** S3 request and application logs on your HyperStore nodes. In the log file copies that get sent to Clodian the user ID and IP address fields will say "Not Available".
- With *phonehome\_gdpr* set to false, user IDs and client IP addresses will be not be removed from the S3 request and application log copies that are uploaded to Clodian Support each day.

Default = false

To apply change, after Puppet propagation restart the S3 Service

**Note** If you set this to true, the system will also remove user IDs and client IP addresses from the copies of logs that get sent to Clodian Support in on-demand Node Diagnostics packages (by way of the [Collect Diagnostics](#) feature), while leaving this information in the original logs on your nodes.

*admin\_auth\_user*

If the Admin Service is configured to require HTTP(S) Basic Authentication from clients (if [admin\\_auth\\_enabled](#) is set to *true*), this is the username for clients to use when submitting HTTP(S) requests to the Admin Service.

Default = sysadmin

To apply change, after Puppet propagation restart S3 Service and CMC

*admin\_auth\_pass*

If the Admin Service is configured to require HTTP(S) Basic Authentication from clients (if [admin\\_auth](#)

[enabled](#) is set to *true*), this is the password for clients to use when submitting HTTP(S) requests to the Admin Service. In this setting the password is configured as a comma-separated pair of "<Jetty\_obfuscated\_password>,<cleartext\_password>".

For information about creating a Jetty-obfuscated password, see **"HTTP/S Basic Authentication for Admin API Access"** (page 748).

Default if original HyperStore install was version 7.2.2 or newer = "<obfuscated>,<cleartext>" of a random password generated by the system upon installation.

Default if original HyperStore install was older than version 7.2.2 = "1uv91x1n1tv91vt1x0z1uuq,public"

To apply change, after Puppet propagation restart S3 Service and CMC

*admin\_auth\_realm*

If the Admin Service is configured to require HTTP(S) Basic Authentication from clients (if [admin\\_auth\\_enabled](#) is set to *true*), this is the realm name used by the Admin Service for Basic Authentication purposes.

Default = ClouidianAdmin

To apply change, after Puppet propagation restart S3 Service and CMC

*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 if original HyperStore install was version 6.0.2 or newer = *true*

Default if original HyperStore install was older than version 6.0.2 = *false*

To apply change, after Puppet propagation restart S3 Service

*admin\_secure*

If set to "true", the Admin Service will accept **only** HTTPS connections from clients (through port 19443 by default). If set to "false", the Admin Service will allow regular HTTP connections from clients (through port 18080) as well as HTTPS connections (through port 19443).

This setting also controls CMC client-side behavior when the CMC calls the Admin Service for tasks such as creating users, creating storage policies, and retrieving system monitoring data. If *admin\_secure* is "true", the CMC will exclusively use HTTPS when making requests to the Admin Service. If *admin\_secure* is "false", the CMC will exclusively use regular HTTP when making requests to the Admin Service.

Note however that even if you set *admin\_secure* to "false" -- so that the Admin Service accepts HTTP requests as well as HTTPS requests; and so that the CMC sends only HTTP requests to the Admin Service -- the Admin Service's HTTPS port will still be accessed by other HyperStore system components. In particular, some of the **"System cron Jobs"** (page 473) use HTTPS to make calls to the Admin Service.

Default = *true*

**Note** If your original HyperStore install was **older than version 6.0.2** and you have upgraded to the current version, the *admin\_secure* setting does not appear in the *common.csv* file and an internal default value of "false" is used. In such systems, if you want the Admin Service to accept only HTTPS connections from clients, add the line *admin\_secure,true* to the *common.csv* file.

To apply change, after Puppet propagation restart S3 Service and CMC

*cmc\_admin\_secure\_port*

If the CMC is using HTTPS to connect to the Admin Service (as it will if *admin\_secure* is set to "true"), this is the Admin Service listening port number to which it will connect. Note that this setting controls CMC client-side configuration, not Admin Service configuration.

Default = 19443

To apply change, after Puppet propagation restart CMC.

*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 apply change, after Puppet propagation restart the S3 Service and CMC.

*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. The system will reject a user's attempt to set a new password that does not meet this requirement.

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 apply change, after Puppet propagation restart the S3 Service and CMC.

*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 apply change, after Puppet propagation restart the S3 Service and CMC.

*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.

If this is set to 0, then this password requirement is disabled.

Default = 0

To apply change, after Puppet propagation restart the S3 Service and CMC.

*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.

Default = 0

To apply change, after Puppet propagation restart the S3 Service and CMC.

#### *iam\_service\_enabled*

If this is set to "true" then HyperStore's IAM Service is enabled and IAM functionality will display in the CMC. For more information see **"HyperStore Support for the AWS IAM API"** (page 991).

Default = true

To apply change, after Puppet propagation restart the IAM Service.

#### *iam\_port*

Port on which the HyperStore IAM Service listens for regular HTTP connections.

Default = 16080

To apply change, after Puppet propagation restart the IAM Service.

#### *iam\_secure*

If set to "true", the IAM Service will accept **only** HTTPS connections from clients (through port 16443 by default). If set to "false", the IAM Service will allow regular HTTP connections from clients (through port 16080) as well as HTTPS connections (through port 16443).

This setting also controls CMC client-side behavior when the CMC calls the IAM Service for tasks such as creating IAM or creating IAM policies. If *iam\_secure* is "true", the CMC will exclusively use HTTPS when making requests to the IAM Service. If *iam\_secure* is "false", the CMC will exclusively use regular HTTP when making requests to the IAM Service.

Default = false

To apply change, after Puppet propagation restart the IAM Service and the CMC.

#### *iam\_secure\_port*

Port on which the HyperStore IAM Service listens for HTTPS connections.

Default = 16443

To apply change, after Puppet propagation restart the IAM Service.

#### *iam\_service\_endpoint*

IAM Service endpoint. This setting is controlled by the installer. Do not edit this setting directly. For instructions on changing the IAM Service endpoint, see **"Changing S3, Admin, CMC, or IAM Service Endpoints"** (page 600).

Default = Set during installation based on operator input

#### *cloudian\_s3admin\_min\_threads*

Minimum number of threads to keep in each Admin Service node's HTTP request processing thread pool.

The Admin Service uses a thread pool to process incoming HTTP requests from clients. An initial pool of threads is created at server initialization time, and additional threads may be added if needed to process queued jobs. Idle threads are terminated if not used within a thread timeout period — unless the number of threads in the pool is down to *cloudian\_s3admin\_min\_threads*. 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 apply change, after Puppet propagation restart S3 Service

*cloudian\_s3admin\_max\_threads*

Maximum number of threads to allow in the Admin Service's HTTP request processing thread pool. If there are fewer than this many threads in the pool, new threads may be created as needed in order to handle queued HTTP request processing jobs. If the maximum thread pool size is reached, no more threads will be created — instead, queued HTTP request processing jobs must wait for an existing thread to become free.

Default = 50

To apply change, after Puppet propagation restart S3 Service

*cloudian\_s3admin\_max\_idletime*

When the Admin Service processes HTTP 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 apply change, after Puppet propagation restart S3 Service

*cloudian\_s3admin\_lowres\_maxidletime*

Special, "low resource" maximum idle time to apply to Admin Service HTTP connections when the number of simultaneous connections to an Admin Service node exceeds *cloudian\_s3admin\_lowres\_maxconnections*. 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 general idle timer behavior, see the description of *cloudian\_s3admin\_max\_idletime* above.)

Default = 5000

To apply change, after Puppet propagation restart S3 Service

*cloudian\_s3admin\_lowres\_maxconnections*

If the number of simultaneous HTTP connections to an Admin Service node exceeds this value, the special idle timer configured by *cloudian\_s3admin\_lowres\_maxidletime* is applied to that node rather than the usual *cloudian\_s3admin\_max\_idletime*.

Default = 1000

To apply change, after Puppet propagation restart S3 Service

*cloudian\_s3\_max\_threads*

Maximum number of threads to allow in the S3 Service's HTTP request processing thread pool. If there are fewer than this many threads in the pool, new threads may be created as needed in order to handle queued HTTP request processing jobs. If the maximum thread pool size is reached, no more threads will be created —

instead, queued HTTP request processing jobs must wait for an existing thread to become free.

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

#### *cloudian\_s3\_max\_idletime*

When the S3 Service processes HTTP 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 apply change, after Puppet propagation restart S3 Service

#### *cloudian\_s3\_lowres\_maxidletime*

Special, *low resource* maximum idle timer to apply to S3 Service HTTP connections when the number of simultaneous connections to an S3 Service node exceeds *cloudian\_s3\_lowres\_maxconnections*. Configured in milliseconds.

Default = 5000

To apply change, after Puppet propagation restart S3 Service

#### *cloudian\_s3\_lowres\_maxconnections*

If the number of simultaneous HTTP connections to an S3 Service node exceeds this value, the special idle timer configured by *cloudian\_s3\_lowres\_maxidletime* is applied to that node.

Default = 2000

To apply change, after Puppet propagation restart S3 Service

#### *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 Clodian/1.0 HyperStore/"

To apply change, after Puppet propagation restart the S3 Service

#### *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\_proxy\_protocol\_enabled* to *true* if you want the S3 Service to support the PROXY Protocol. If you do so, the following will happen:

- The S3 Server will create dedicated PROXY Protocol connectors listening on port 81 (for regular PROXY Protocol) and port 4431 (for PROXY Protocol with SSL/TLS). These connectors will be enabled and configured in *s3.xml.erb*. By default these connectors are disabled that template.
- 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 client IP address as the source address, rather than showing the loader balancer's IP address as the source. Also, the true 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 "whitelist" certain source IP addresses.

Setting `s3_proxy_protocol_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 layer, 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 "[HTTPS Support \(TLS/SSL\)](#)" (page 114).

If `s3_proxy_protocol_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.

Default = false

To apply change, after Puppet propagation restart the S3 Service

*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 `cloudian_s3_max_heap_percent`.

Default = 30g

To apply change, after Puppet propagation restart S3 Service

*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 `cloudian_s3_heap_limit`.

Default = 15

To apply change, after Puppet propagation restart S3 Service

*cloudian\_hss\_heap\_limit*

Maximum heap size limit for the HyperStore 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 `cloudian_hss_max_heap_percent`.

Default = 30g

To apply change, after Puppet propagation restart HyperStore Service

*cloudian\_hss\_max\_heap\_percent*

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 value and the limiting value set by `cloudian_hss_`

*heap\_limit.*

Default = 15

To apply change, after Puppet propagation restart HyperStore Service

*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 *cloudian\_admin\_max\_heap\_percent*.

Default = 16g

To apply change, after Puppet propagation restart S3 Service

*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 *cloudian\_admin\_heap\_limit*.

Default = 5

To apply change, after Puppet propagation restart S3 Service

*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 apply change, after Puppet propagation restart S3 Service

*cloudian\_hss\_init\_heap\_percent*

Initial heap size (JAVA\_OPTS -Xms value) for the HyperStore Service application, as a percentage of the -Xmx value.

Default = 25

To apply change, after Puppet propagation restart HyperStore Service

*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 apply change, after Puppet propagation restart S3 Service

*cloudian\_s3\_new\_heap\_percent*

New heap size (JAVA\_OPTS -Xmn value) for the S3 Service application, as a percentage of the -Xmx value. This is the heap size specifically for "young generation" objects.

Default = 25

To apply change, after Puppet propagation restart S3 Service

*cloudian\_hss\_new\_heap\_percent*

New heap size (JAVA\_OPTS -Xmn value) for the HyperStore Service application, as a percentage of the -Xmx

value. This is the heap size specifically for "young generation" objects.

Default = 25

To apply change, after Puppet propagation restart HyperStore Service

*cloudian\_admin\_new\_heap\_percent*

New heap size (JAVA\_OPTS -Xmn value) for the Admin Service application, as a percentage of the -Xmx value. This is the heap size specifically for "young generation" objects.

Default = 25

To apply change, after Puppet propagation restart S3 Service

*cassandras\_per\_s3node*

Maximum number of Cassandra nodes to which an individual S3 Service node may keep simultaneous live connections.

Default = 9

To apply change, after Puppet propagation restart S3 Service

*cassandra\_max\_active*

The maximum allowed number of simultaneously active connections in a Cassandra 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 *cassandra.cluster.MaxWaitTimeWhenExhausted* in *mts.properties.erb* (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.

This setting is controlled by a HyperStore performance optimization script that takes the local hardware environment into account. **Do not manually edit this setting** -- your edits will not be applied by the system.

## AES-256 Encryption Section

*cloudian\_s3\_aes256encryption\_enabled*

This setting is for enabling AES-256 in HyperStore, for use with server-side encryption. If AES-256 is not enabled, AES-128 is used instead.

Default = false

## REDIS Section

*redis\_credentials\_master\_port*

Port on which the Redis Credentials master node listens for data storage requests from clients. The Redis Credentials slave nodes will listen on this port as well.

Default = 6379

To apply change, after Puppet propagation restart Redis Credentials, S3 Service, and HyperStore Service

*redis\_monitor\_subscription\_check*

This setting controls certain aspects of HyperStore services start-up behavior, including during a HyperStore version upgrade operation. Leave this setting at its default value.

Default = false

#### *redis\_qos\_master\_port*

Port on which the Redis QoS master node listens for data storage requests from clients. The Redis QoS slave nodes will listen on this port as well.

Default = 6380

To apply change, after Puppet propagation restart Redis QoS, S3 Service, and HyperStore Service

#### *redis\_monitor\_listener\_port*

Port on which the Redis Monitor can be queried for information about the Redis cluster state, via the Redis Monitor CLI.

Default = 9078

To apply change, after Puppet propagation restart Redis Monitor Service

#### *redis\_lib\_directory*

Directory in which to store Redis data files.

Default = `/var/lib/redis`

If you want to change this for a HyperStore system that's already in operation, consult with Clouidian Support.

#### *redis\_log\_directory*

Directory in which to store Redis log files.

Default = `/var/log/redis`

To apply change, after Puppet propagation restart Redis Credentials and Redis QoS

## CASSANDRA Section

#### *cassandra\_max\_heap\_size*

Max Heap size setting (memory allocation) for the Cassandra application. If this setting is assigned a value, this value will be used for `MAX_HEAP_SIZE` in `cassandra-env.sh`. By default the `cassandra_max_heap_size` setting is commented out and not used by the system. Instead, the default behavior for Clouidian HyperStore is for `MAX_HEAP_SIZE` in `cassandra-env.sh` to be automatically set based on the host's RAM size.

Default = commented out and unused

To apply change, after Puppet propagation restart Cassandra Service

#### *cassandra\_enable\_gc\_logging*

Enable Java garbage collection (GC) logging for Cassandra. The log is written to `/var/log/cassandra/gc.log`.

Default = `true`

To apply change, after Puppet propagation restart Cassandra Service

**Note** GC logging is also enabled for the S3 Service, Admin Service, and HyperStore Service. These GC logs are under `/var/log/cloudian` and are named `s3-gc.log`, `admin-gc.log`, and `hss-gc.log`, respectively.

#### *cassandra\_heapdump\_on\_oomemerr*

Whether to enable Java heap dump on a Cassandra application out-of-memory error. Set to "false" to disable heap dumps on out-of-memory errors. This removes the JVM\_OPTS -XX:+HeapDumpOnOutOfMemoryError option from *cassandra-env.sh*.

Default = false

To apply change, after Puppet propagation restart Cassandra Service

*cassandra\_lib\_directory*

Directory in which to store Cassandra application state data.

Default = /var/lib/cassandra

If you want to change this for a HyperStore system that's already in operation, consult with Clouidian Support.

*cassandra\_log\_directory*

Directory in which to store Cassandra application log files.

Default = /var/log/cassandra

To apply change, after Puppet propagation restart Cassandra Service

*cassandra\_saved\_cache\_directory*

Directory in which to store the Cassandra saved\_caches file.

Default = /var/lib/cassandra

If you want to change this for a HyperStore system that's already in operation, consult with Clouidian Support.

*cassandra\_commit\_log\_directory*

Directory in which to store the Cassandra commit log file.

Default = Set during installation based on operator input, if host has multiple disks. For hosts with only one disk, default is /var/lib/cassandra/commit

If you want to change this for a HyperStore system that's already in operation, consult with Clouidian Support.

*cassandra\_data\_directory*

Directory in which to store Cassandra data files. By default, Cassandra is used only for storing S3 object metadata (metadata associated with individual objects) and service metadata such as account information, usage data, and system monitoring data.

Default = Set during installation based on operator input, if host has multiple disks. For hosts with only one disk, default is /var/lib/cassandra/data

If you want to change this for a HyperStore system that's already in operation, consult with Clouidian Support.

**IMPORTANT !** The Cassandra data directory should **not** be mounted on a shared file system such as a NAS device.

*cassandra\_port*

Port on which Cassandra listens for data operations requests from clients.

Default = 9160

To apply change, after Puppet propagation restart Cassandra Service

*concurrent\_compactors*

Number of simultaneous Cassandra compactions to allow, 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

*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 (deleted data markers), 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 Get Bucket (List Objects) operation.

Default = 50000

To apply change, after Puppet propagation restart Cassandra

*cassandra\_tombstone\_failure\_threshold*

If while processing a Cassandra query it is found that a single row within a column family has more than this many tombstones (deleted data markers), 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 Get Bucket (List Objects) operation.

Default = 100000

To apply change, after Puppet propagation restart Cassandra

*cassandra\_tombstone\_cleanup\_threshold*

If while processing a Cassandra query it is found that a single row within a CLOUDIAN\_METADATA or MPSession column family has more than this many tombstones (deleted data markers), 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 Get Bucket (List Objects) operation.

Default = 75000

To apply change, after Puppet propagation restart the S3 Service

**Note** You can also manually trigger a tombstone purge for a specific bucket, as described in **"Tombstone Cleanup Processing"** (page 476).

*cassandra\_tombstone\_gcgrace*

While doing a purge of tombstones (deleted data markers) in a CLOUDIAN\_METADATA or MPSession column family, 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 apply change, after Puppet propagation restart the S3 Service

#### *cassandra\_listen\_address*

For each Cassandra node, the IP interface on which the node will listen for cluster management communications from other Cassandra nodes in the cluster. **Specify this as an IP address alias.** Puppet will use the alias to determine the actual IP address for each node.

Options are `%{::cloudian_ipaddress}`, `%{::ipaddress_eth#}`, `%{::ipaddress_lo}`, or `%{::ipaddress_bind#}`

Default = `%{::cloudian_ipaddress}`

To apply change, after Puppet propagation restart Cassandra Service

#### *cassandra\_rpc\_address*

For each Cassandra node, the IP interface on which the node will listen for data operations requests from clients, via Thrift RPC. **Specify this as an IP address alias.** Puppet will use the alias to determine the actual IP address for each node.

Options are `%{::cloudian_ipaddress}`, `%{::ipaddress_eth#}`, `%{::ipaddress_lo}`, or `%{::ipaddress_bind#}`

If desired, this can be the same IP address alias as used for *cassandra\_listen\_address*.

Default = `%{::cloudian_ipaddress}`

If you want to change this for a HyperStore system that's already in operation, consult with Clodian Support.

#### *cassandra\_default\_node\_datacenter*

This instance of this setting is not used. Instead, the instance of *cassandra\_default\_node\_datacenter* in *region.csv* is used. Typically you should have no need to edit that setting.

Default = commented out

## CMC Section

#### *admin\_whitelist\_enabled*

Whether to enable the billing "whitelist" feature that allows favorable billing terms for a specified list of source IP addresses or subnets. If this feature is enabled, whitelist management functionality displays in the CMC. This functionality is available only to HyperStore system administrators, not to group admins or regular users.

Default = true

To apply change, after Puppet propagation restart S3 Service and CMC

#### *cmc\_log\_directory*

Directory in which to store CMC application logs.

Default = `/var/log/cloudian`

To apply change, after Puppet propagation restart CMC

#### *cmc\_admin\_host\_ip*

Fully qualified domain name for the Admin Service. The CMC connects to this service.

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 = Set during installation based on operator input.

To apply change, after Puppet propagation restart CMC.

*cmc\_cloudian\_admin\_user*

For the CMC, the login user name of the default system admin user. The system admin will use this user name to log into the CMC.

Default = admin

**This cannot be changed.**

**Note** The default password for the default CMC system admin user is "public". You can change this password when you are logged into the CMC as the default admin user. Hold your cursor over the user name that displays in the upper right of the screen (by default, "Admin"), then select the **Security Credentials** option and change the password.

*cmc\_domain*

Service endpoint (fully qualified domain name) for the CMC. This endpoint must be resolvable for CMC clients.

Default = Set during installation based on operator input.

To change this endpoint, use the **"Installer Advanced Configuration Options"** (page 501).

*cmc\_web\_secure*

Whether the CMC should require HTTPS for all incoming client connections, true or false.

Set this property to "true" to require HTTPS. In this mode of operation, requests incoming to the CMC's regular HTTP port will be redirected to the CMC's HTTPS port.

Set this property to "false" to allow clients to connect through regular HTTP. In this mode of operation, requests incoming to the CMC's HTTPS port will be redirected to the CMC's regular HTTP port.

Default = true

To apply change, after Puppet propagation restart CMC.

*cmc\_http\_port*

Port on which the CMC listens for regular HTTP requests.

Default = 8888

To apply change, after Puppet propagation restart CMC.

*cmc\_https\_port*

Port on which the CMC listens for HTTPS requests.

Default = 8443

To apply change, after Puppet propagation restart CMC.

*cmc\_admin\_secure\_ssl*

If the CMC is using HTTPS to connect to the Admin Service (as it will if **"admin\_secure"** (page 526) is set to "true"), this setting controls the CMC's requirements regarding the Admin Service's SSL certificate:

- If set to "true", then when the CMC makes HTTPS connections to the Admin Service the CMC's HTTPS client will require that the Admin Service's SSL certificate be **CA validated** (or else will drop the connection).
- If set to "false", then when the CMC makes HTTPS connections to the Admin Service the CMC's HTTPS client will allow the Admin Service's SSL certificate to be self-signed.

**Note** Note that the SSL certificate that is used with the Admin Service by default is self-signed.

Default = false

To apply change, after Puppet propagation restart CMC.

#### *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://enterprise2:8443/Cloudian/dashboard.htm*).

To apply change, after Puppet propagation restart CMC.

#### *cmc\_storageuri\_ssl\_enabled*

If this is set to "true", the CMC uses HTTPS to connect to the HyperStore S3 Service (in implementing the CMC **Buckets & Objects** functionality). If "false", the CMC uses regular HTTP to connect to the HyperStore S3 Service.

Do not set this to "true" unless you have set up HTTPS for your HyperStore S3 Service (following the instructions in **"HTTPS Support (TLS/SSL)"** (page 114) . By default the HyperStore S3 Service does not use HTTPS.

Default = false

#### *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 the administrator can type the group name.

Default = true

To apply change, after Puppet propagation restart CMC.

**Note** If the number of groups in your system exceeds the value of the *cmc\_grouplist\_size\_max* 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\_login\_languageselection\_enabled*

This setting controls whether to display at the top of the CMC's **Sign In** screen 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 **Sign In** screen 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 **Sign In** screen, 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 apply change, after Puppet propagation restart CMC.

#### *cmc\_login\_grouplist\_enabled*

This setting controls whether to enable the **Group** drop-down list on the CMC's **Sign In** screen. The **Group** drop-down list lists all groups registered in the HyperStore system, and when users log in they can choose their group from the list. If disabled, the drop-down list will not display and instead users will need to enter their group name in a 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 apply change, after Puppet propagation restart CMC.

**Note** If the number of groups in your system exceeds the value of the *cmc\_grouplist\_size\_max* 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\_login\_grouplist\_enabled*.

#### *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 setting for *cmc\_login\_grouplist\_enabled* and *cmc\_grouplist\_enabled*.

For example, if *cmc\_login\_grouplist\_enabled* and *cmc\_grouplist\_enabled* are set to "true" and *cmc\_grouplist\_size\_max* 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 apply change, after Puppet propagation restart CMC.

#### *cmc\_keystore\_passwd*

The password for the CMC's TLS/SSL keystore. This setting is managed automatically by the HyperStore installer's Advanced Configuration functions. For more information see "**HTTPS Support (TLS/SSL)**" (page 114).

Default = 123cloudian456

#### *cmc\_keystore\_file*

The CMC's TLS/SSL keystore file. This setting is managed automatically by the HyperStore installer's Advanced Configuration functions. For more information see "**HTTPS Support (TLS/SSL)**" (page 114).

Default = .keystore\_cloudian

#### *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 apply change, after Puppet propagation restart CMC.

#### *cmc\_view\_user\_data*

Within the **Manage Users** function in the CMC GUI, this setting enables or disables the capability of admin users to access regular users' storage buckets. 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:

- true — This capability will display for users logged in as a system administrator or group administrator. For group admins this capability is restricted to their own group.
- false — This capability will not display for any admin users.
- SystemAdmin — This capability will display only for users logged in as a system administrator.
- GroupAdmin — This capability will display only for users logged in as a group administrator.

Default = false

To apply change, after Puppet propagation restart CMC.

**Note** Regardless of how the *cmc\_view\_user\_data* setting is configured, regular users can view and manage their own object data through the **Buckets & Objects** section of the CMC.

#### *cmc\_crr\_external\_enabled*

This controls whether settings for replicating 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** dialog. The default is false, so that such settings do not appear in the dialog. This setting is relevant only if your organization is a legacy user of cross region replication to external systems. If so, and if you want these settings to display for your users as they configure their buckets in the CMC, you can set *cmc\_crr\_external\_enabled* to true.

Default = false

To apply change, after Puppet propagation restart CMC.

#### *cmc\_login\_banner\_\**

For information about using the *cmc\_login\_banner\_size*, *cmc\_login\_banner\_title*, *cmc\_login\_banner\_message*, and *cmc\_login\_banner\_button\_confirm* settings see "**Configuring a Login Page Acknowledgment Gate**" (page 408).

#### *cmc\_bucket\_tiering\_default\_destination\_list*

The list of auto-tiering destinations to display in the CMC interface that bucket owners use to configure auto-

tiering for a bucket (for more information on this interface see **"Configure a Bucket Lifecycle Policy for Object Auto-Tiering or Expiration"** (page 227)). Specify this as a quote-enclosed list, with comma-separation between destination attributes and vertical bar separation between destinations, like this:

```
"<name>,<endpoint>,<protocol>|<name>,<endpoint>,<protocol>|..."
```

This can be multiple destinations (as it is by default), or you can edit the setting to have just one destination in the "list" if you want your users to only use that one destination.

For multiple destinations you can have as many as you want, within reason (bear in mind that in the interface the dialog box will expand to accommodate the additional destinations).

The *<name>* will display in the CMC interface that bucket owners use to configure auto-tiering, as the auto-tiering destination name. The *<protocol>* must be one of the following:

- s3
- glacier
- azure
- spectra

If you wish you can include multiple destinations of the same type, if those destinations have different endpoints. For example, "Spectra 1,<endpoint1>,spectra|Spectra 2,<endpoint2>,spectra". Each such destination will then appear in the CMC interface for users configuring their buckets for auto-tiering.

Default = "AWS S3,https://s3.amazonaws.com,s3|AWS GLACIER,https://s3.amazonaws.com,glacier|Google,https://storage.googleapis.com,s3|Azure,https://blob.core.windows.net,azure"

To apply change, after Puppet propagation restart CMC.

**Note** If your original HyperStore version was older than 7.1.4, then after upgrade to 7.1.4 or later your default value here will also include a Spectra destination.

**Note** For more information about setting up auto-tiering, see **"Setting Up Auto-Tiering"** (page 180). That section includes information about how to enable the auto-tiering feature (which is disabled by default in the CMC interface) and about the option of having all end users use the same system-configured security credentials for accessing the tiering destination (rather than supplying their own security credentials for tiering, which is the default behavior). Note that if you choose to have all users use the same tiering security credentials you will need to specify a single system default tiering destination -- using a setting in the CMC's **Configuration Settings** page, as described in **"Setting Up Auto-Tiering"** (page 180) -- and that configuration setting will **override** the *cmc\_bucket\_tiering\_default\_destination\_list* setting.

## AWS MMS Proxy Section

*awsmmsproxy\_host*

If you are using the AWS Marketplace Metering Service version of Clouidian HyperStore, you must set this property to the public IP address of your AWS Proxy Server running on an AWS EC2 instance. For more information about this version of HyperStore, contact your Clouidian representative.

Default = empty

To apply change, after Puppet propagation restart S3 Service.

## Usage Section

*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.

Default = false

To apply change, after Puppet propagation restart S3 Service.

## Elastic Search Section

*cloudian\_elasticsearch\_hosts*

For information about this setting, see **"Elasticsearch Integration for Object Metadata"** (page 171).

## SQS Section

*sqs\_\**

For information about these settings, see **"HyperStore Support for the AWS SQS API"** (page 1041).

## Alerting Section

*alert\_suppression\_list*

With this setting you can configure a list of one or more log error message codes for which the system will suppress alerting. When the listed message codes appear in HyperStore logs, alerts will **not** appear in the CMC and the system will **not** send notification emails or SNMP traps.

You may want to use this setting if you are seeing certain logging error based alerts too frequently or if you are seeing alerts for logging errors that you don't consider important enough to merit an alert. Note that this setting does not impact HyperStore logging behavior (it does not stop any messages from being written to HyperStore logs), it only affects HyperStore alerting behavior.

If you list multiple message codes use comma-separation between the message codes, and enclose the list in quotes.

For example, if you configure the setting as follows:

```
alert_suppression_list,"HS240113,HS233064"
```

Then if log entries with message codes HS240113 or HS233064 are written to HyperStore logs, no alerts regarding those log entries will be generated in the CMC and no email or SNMP traps will be sent regarding those log entries.

Default = empty

To apply change, after Puppet propagation restart S3 Service.

**Note** For a full list of log message codes see Log Message Codes.

**IMPORTANT !** Do not manually edit settings that appear below this point in the *common.csv* file.

## 9.5.2. hyperstore-server.properties.erb

The *hyperstore-server.properties* file configures the HyperStore Service. On each of your HyperStore nodes, the file is located at the following path by default:

```
/opt/cloudian/conf/hyperstore-server.properties
```

**Do not directly edit the *hyperstore-server.properties* file on individual HyperStore nodes.** Instead, if you want to make changes to the settings in this file, edit the configuration template file *hyperstore-server.properties.erb* on the Puppet master node:

```
/etc/cloudian-<version>-puppet/modules/cloudians3/templates/hyperstore-server.properties.erb
```

Certain *hyperstore-server.properties.erb* properties take their values from settings in [common.csv](#) or from settings that you can control through the CMC's [Configuration Settings](#) page. In the *hyperstore-server.properties.erb* file these properties' values are formatted as bracket-enclosed variables, like `<%= ... %>`. In the property documentation below, the descriptions of such properties indicate "Takes its value from `<location>`: `<setting>`"; use that setting instead." . The remaining properties in the *hyperstore-server.properties.erb* file -- those that are "hard-coded" with specific values -- are settings that in typical circumstances you should have no need to edit. Therefore **in typical circumstances you should not need to manually edit the *hyperstore-server.properties.erb* file.**

*If you are using the HyperStore Shell*

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Puppet master node you can edit this configuration file with this command:

```
$ hspkg config -e hyperstore-server.properties.erb
```

Specify just the configuration file name, not the full path to the file.

In the background this invokes the Linux text editor *vi* to display and modify the configuration file. Therefore you can use the [standard keystrokes supported by vi](#) to make and save changes to the file.

**IMPORTANT !** If you do make edits to *hyperstore-server.properties.erb*, be sure to push your edits to the cluster and restart the HyperStore Service to apply your changes. For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

The *hyperstore-server.properties.erb* file has the following settings:

*cloudian.storage.datadir*

Takes its value from *common.csv*: **"hyperstore\_data\_directory"** (page 517); use that setting instead.

*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 **"Secure Delete"** (page 120)

**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

*messaging.service.listen.address*

Takes its value from *common.csv*: "**hyperstore\_listen\_ip**" (page 517); use that setting instead.

*messaging.service.listen.port*

The port on which a HyperStore Service node listens for messages from other HyperStore Service nodes. This internal cluster messaging service is used in support of cluster management operations such as node repair.

Default = 19050

*messaging.service.read.buffer.size*

When a HyperStore Service node reads data it has received over the network from other HyperStore Service nodes -- such as during repair operations -- this is the read buffer size.

Default = 65536

*messaging.service.write.buffer.size*

When a HyperStore Service node writes data to the network for transferring to other HyperStore Service nodes -- such as during repair operations -- this is the write buffer size.

Default = 1048576

*messaging.service.threadpool.corepoolsize*

Takes its value from *common.csv*: "**hyperstore\_messaging\_service\_threadpool**" (page 520); use that setting instead.

*messaging.service.maxconnections*

Maximum simultaneous number of connections that the HyperStore messaging service will accept.

Default = 2000

*messaging.service.repairfile.timeout*

Maximum time in seconds to allow for repair of a single file on a HyperStore node. File repair entails checking other HyperStore nodes to find the most recent copy of the file and then downloading that copy.

Default = 120

*messaging.service.connection.timeout*

Maximum time in seconds that a HyperStore node will allow for establishing a connection to another HyperStore node, for conducting inter-node operations.

Default = 300

*repair.session.threadpool.corepoolsize*

Takes its value from *common.csv*: "**hyperstore\_repair\_session\_threadpool**" (page 520); use that setting instead.

*repair.session.rangeslice.maxrows*

During [hsstool repair](#) or [hsstool repairec](#) operations, the maximum number of row keys to retrieve per `get_range_slice` query performed on Cassandra `<GROUPID>_METADATA` column families.

Default = 2

**Note** Cloudian, Inc recommends that you leave this setting at its default value. Do **not** set it to a value lower than 2.

*repair.session.columnslice.maxcolumns*

During [hsstool repair](#) or [hsstool repairec](#) operations, the maximum number of columns to retrieve per `get_slice` or `get_range_slice` query performed on Cassandra `<GROUPID>_METADATA` column families.

Default = 1000

*repair.session.slicequery.maxretries*

During [hsstool repair](#) or [hsstool repairec](#) operations, the maximum number of times to retry `get_slice` or `get_range_slice` queries after encountering a timeout. The timeout interval is configured by `casandra.cluster.CassandraThriftSocketTimeout` in `mts.properties`, and retries are attempted as soon as a timeout occurs.

Default = 3

*repair.session.updateobjs.queue.maxlength*

During [hsstool repair](#) 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 object metadata on remote replicas versus object metadata on the local node.

This target maximum may be exceeded in certain circumstances as described for `repair.session.updateobjs.queue.maxwaittime` (below).

Default = 1000

Reloadable via JMX (HyperStore Service's JMX port 19082; MBean attribute = `com.gemini.cloudian.hybrid.server → FileRepairService → Attributes → RepairSessionJobQueueMaxLength`)

*repair.session.updateobjs.queue.waittime*

If during [hsstool repair](#) operations the differencer detects that the number of queued object update jobs is at or above the target maximum (as configured by `repair.session.updateobjs.queue.maxlength`), the number of seconds to wait before checking the queue size again. During this interval the differencer adds no more object update jobs to the queue.

Default = 2

*repair.session.updateobjs.queue.maxwaittime*

During [hsstool repair](#) operations, 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 it's larger than the target maximum size, the wait time procedure starts over again.

Default = 120

*repair.session.object.download.maxretries*

During [hsstool repair](#) or [hsstool repairrec](#) operations, the maximum number of times to retry object download requests after encountering a timeout. The timeout interval is configured by *mts.properties: cassandra.cluster.CassandraThriftSocketTimeout*, and retries are attempted as soon as a timeout occurs.

Default = 3

*repair.session.inmemory.fileindex*

When performing Merkle Tree based [hsstool repair](#) (the default repair type) for files in the HyperStore File System, whether to hold the file indexes in memory rather than writing them to disk. Options are:

- true — For each vNode being repaired, a file index directory is created and is held in memory unless its size exceeds a threshold in which case it is written to disk (under the HyperStore data mount point that the vNode is associated with). For most vNodes it will not be necessary to write the file indexes to disk. If file indexes are written to disk, they are automatically deleted after the repair operation completes.
- false — For each vNode being repaired, a file index directory is created and written to disk (under the HyperStore data mount point that the vNode is associated with), regardless of size. The file indexes are automatically deleted after the repair operation completes.

Default = true

Reloadable via JMX (HyperStore Service's JMX port 19082; MBean attribute = com.gemini.cloudian.hybrid.server → FileRepairService → Attributes → RepairSessionInMemoryFileIndex)

*repair.digest.index.threadpool.corepoolsize*

Takes its value from *common.csv*: "**hyperstore\_repair\_digest\_index\_threadpool**" (page 520); use that setting instead.

*repair.merkletree.response.waittime*

During *hsstool repair* operations, 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 *repair.merkletree.response.waittime* property 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.

Default = 120

*rangerepair.threadpool.corepoolsize*

Takes its value from *common.csv*: "**hyperstore\_rangerepair\_threadpool**" (page 521); use that setting instead.

*stream.outbound.threadpool.corepoolsize*

Takes its value from *common.csv*: "**hyperstore\_stream\_outbound\_threadpool**" (page 521); use that setting instead.

*repairrec.sessionscan.threadpool.corepoolsize*

Takes its value from *common.csv*: "**hyperstore\_repairrec\_sessionscan\_threadpool**" (page 522); use that setting instead.

*repairrec.digestrequest.threadpool.fixedpoolsize*

Takes its value from *common.csv*: "**hyperstore\_repairec\_digestrequest\_threadpool**" (page 522); use that setting instead.

*repairec.task.threadpool.corepoolsize*

Takes its value from *common.csv*: "**hyperstore\_repairec\_task\_threadpool**" (page 522); use that setting instead.

*repairec.rocksdbscan.threadpool.corepoolsize*

Takes its value from *common.csv*: "**hyperstore\_repairec\_rocksdbscan\_threadpool**" (page 523); use that setting instead.

*repairec.session.queue.maxlength*

During "**hsstool repairec**" (page 697) 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 object metadata on remote replicas versus object metadata on the local node.

This target maximum may be exceeded in certain circumstances as described for *repairec.session.queue.maxwaittime*(below).

Default = 2000

*repairec.session.queue.waittime*

If during [hsstool repairec](#) operations the differencer detects that the number of queued object update jobs is at or above the target maximum (as configured by *repairec.session.updateobjs.queue.maxlength*), the number of seconds to wait before checking the queue size again. During this interval the differencer adds no more object update jobs to the queue.

Default = 2

*repairec.session.queue.maxwaittime*

During [hsstool repairec](#) operations, 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 it's larger than the target maximum size, the wait time procedure starts over again.

Default = 120

*downloadrange.session.threadpool.corepoolsize*

Takes its value from *common.csv*: "**hyperstore\_downloadrange\_session\_threadpool**" (page 521); use that setting instead.

*uploadrange.session.threadpool.corepoolsize*

Takes its value from *common.csv*: "**hyperstore\_uploadrange\_session\_threadpool**" (page 521); use that setting instead.

*decommission.threadpool.corepoolsize*

Takes its value from *common.csv*: "**hyperstore\_decommission\_threadpool**" (page 521); use that setting instead.

*cleanup.session.threadpool.corepoolsize*

Takes its value from *common.csv*: "**hyperstore\_cleanup\_session\_threadpool**" (page 522); use that setting

instead.

*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.

This target maximum may be exceeded in certain circumstances as described for *cleanup.session.deleteobjs.queue.maxwaittime*.

Default = 1000

*cleanup.session.deleteobjs.queue.waittime*

If during [hsstool cleanup](#) or [hsstool cleanupec](#) operations a cleanup job detects that the number of queued object delete jobs is at or above the target maximum (as configured by *cleanup.session.deleteobjs.queue.maxlength*), the number of seconds to wait before checking the queue size again. During this interval the cleanup job adds no more object delete jobs to the queue.

Default = 2

*cleanup.session.deleteobjs.queue.maxwaittime*

During [hsstool cleanup](#) or [hsstool cleanupec](#) operations, the maximum total number of seconds for the differencer to wait for the object delete job queue to fall below its target maximum size. After this interval, the differencer goes ahead and writes its current batch of delete requests to the queue. In this scenario, the queue can grow beyond the target maximum size. The next time that the differencer has object delete requests, it again checks the queue size, and if it's larger than the target maximum size, the wait time procedure starts over again.

Default = 120

*cleanup.session.delete.graceperiod*

During [hsstool cleanup](#) or [hsstool cleanupec](#) operations, only consider an object for deletion if at least this many seconds have passed since the object's Last Modified timestamp.

Default = 86400

*cleanupjobs.threadpool.corepoolsize*

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 23).

When a physical HyperStore node is cleaned, there is a separate cleanup "job" for each *<policyid>* sub-directory on the physical node. The *cleanupjobs.threadpool.corepoolsize* 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 `common.csv:"hyperstore_cleanup_session_threadpool"` (page 522) 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 `cleanupjobs.threadpool.corepoolsize = 4` and `hyperstore_cleanup_session_threadpool = 10`, then on a given physical node being cleaned a maximum of 4 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 4 jobs.

Default = 4

*max.cleanup.operations.perdc*

Maximum number of [hsstool cleanup](#) or [hsstool cleanupec](#) operations to allow at one time, per data center.

The limit is applied separately to *cleanup* and *cleanupec* operations. For example, if this property 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 simultaneously.

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 this property 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 Clouddian Support.

Default = 1

*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 150) .

Default = 60

**Note** If for some reason you want to trigger proactive repair on a particular node immediately, you can do so by running the **"hsstool proactiverepairq"** (page 674) command with the **"-start"** option.

**Note** For information about temporarily disabling the proactive repair feature, see **"Disabling or Stopping Data Repairs"** (page 154).

*stream.throughput.outbound*

During [hsstool cleanup](#) or [hsstool cleanupec](#) operations, HyperStore nodes may stream large amounts of data to other HyperStore nodes. This setting places an upper limit on outbound streaming throughput during repair operations, in megabits per second.

Default = 800

Reloadable via JMX (HyperStore Service's JMX port 19082; MBean attribute = com.gem-ini.cloudian.hybrid.server → FileStreamingService → Attributes → MaxStreamThroughputOutbound)

*auto.repair.threadpool.corepoolsizesize*

Takes its value from *common.csv*: "**hyperstore\_auto\_repair\_threadpool**" (page 522); use that setting instead.

*auto.repair.scheduler.polltime*

Interval (in minutes) at which each HyperStore node's auto-repair scheduler will check the auto-repair queues for HSFS repair, Cassandra repair, and erasure coded data repair to see whether it's time to initiate a repair on that node.

Default = 10

Reloadable via JMX (HyperStore Service's JMX port 19082; MBean attribute = com.gem-ini.cloudian.hybrid.server → FileRepairService → Attributes → AutoRepairSchedulerPollTime)

*auto.repair.schedule.interval*

Takes its value from the "**Replicas Repair Interval (Minutes)**" (page 351) setting in the CMC's [Configuration Settings](#) page; use that setting instead.

*auto.repairec.schedule.interval*

Takes its value from the "**EC Repair Interval (Minutes)**" (page 352) setting in the CMC's [Configuration Settings](#) page; use that setting instead.

*auto.repaircassandra.schedule.interval*

Takes its value from the "**Cassandra Full Repair Interval (Minutes)**" (page 352) setting in the CMC's [Configuration Settings](#) page; use that setting instead.

*cloudian.storage.jmx.port*

The port on which the HyperStore Service listens for JMX requests.

Default = 19082 (set elsewhere in the manifest structure; do not edit this property)

*disk.fail.action*

Takes its value from the "**HyperStore Disk Failure Action**" (page 342) setting in the CMC's [Configuration Settings](#) page; use that setting instead.

*disk.repair.rebuild*

When HyperStore implements a [replaceDisk](#) operation it automatically executes *hsstool repair* and *hsstool repairec* for the replacement disk. With *disk.repair.rebuild=true* (the default setting), the automatic executions of *hsstool repair* and *hsstool repairec* will use the *-rebuild* option that those operations support. Using the *-rebuild* option is the most efficient way to rebuild data on to a replacement disk. Typically the only occasion you would have for setting *disk.repair.rebuild=false* -- so that the *-rebuild* option is not used -- is if you are instructed to do so by Cloudian Support in the context of troubleshooting a failed attempt to replace a disk.

Default = true

*disk.fail.error.count.threshold*

This setting in combination with the *disk.fail.error.time.threshold* setting provides you the option to specify a read/write error frequency threshold that must be met before the system takes the automated action that is specified by the "HyperStore Disk Failure Action" setting.

The threshold, if configured, is in the form of "If *disk.fail.error.count.threshold* number of HSDISKERROR messages are logged in *cloudian-hyperstore.log* in regard to the same disk within an interval of *disk.fail.error.time.threshold* seconds, then take the automated action specified by the *disk.fail.action* setting."

For example, if *disk.fail.error.count.threshold* = 3, and *disk.fail.error.time.threshold* = 60, and *disk.fail.action* = "disable", then the system will disable any HyperStore data disk for which 3 or more HSDISKERROR messages appear in *cloudian-hyperstore.log* within a 60 second time span.

If you set the two threshold settings to "0", then no threshold behavior is implemented, and instead the automated action specified by the *disk.fail.action* setting is triggered by any single occurrence of an HSDISKERROR message in *cloudian-hyperstore.log*.

Default = 10

#### *disk.fail.error.time.threshold*

Disk error threshold time span in seconds. For more description see *disk.fail.error.count.threshold* above.

Default = 300

#### *disk.check.interval*

Takes its value from *common.csv*: "**hyperstore\_disk\_check\_interval**" (page 523); use that setting instead.

#### *disk.balance.delta*

When the disk balance check is run (at the *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 the configured *disk.balance.delta*. If *disk.balance.delta* = 10 (the default), then, 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 on this feature see "**Automated Disk Management Feature Overview**" (page 157).

Default = 10

#### *disk.audit.interval*

At this configurable interval (in number of minutes), the system 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"). If any such message is found, the disk is automatically disabled in accordance with your configured "**HyperStore Disk Failure Action**" (page 342).

The scan of */var/log/messages* will be limited to the time period since the the last time the disk audit was run.

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.

Default = 60

#### *disk.error.check.fs*

When scanning */var/log/messages* as part of the disk audit, the messages will be first filtered by this file system

name string.

Default = "EXT4-fs"

*max.diskusage.percentage*

The [hsstool repair](#) operation will fail if any HyperStore data disk that stores token ranges impacted by the operation is more than this percent full.

Default = 95

*auto.repair.computedigest.run.number*

Takes its value from *common.csv*: "**auto\_repair\_computedigest\_run\_number**" (page 518); use that setting instead.

*hss.errorlogger.appender*

Do not edit this setting.

*hss.heallogger.appender*

Do not edit this setting.

*enable.cassandra.rangerepair*

If this is set to true, HyperStore uses a "range repair" approach when executing Cassandra 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 150).

Default = true

*retry.rebalance.ranges*

If this is set to true, HyperStore will automatically retry any failed sub-tasks from an *hsstool rebalance* operation that has completed on a newly added node. Like other types of [proactive repair](#), this retry of any failed rebalance sub-tasks occurs once per hour.

Default = true

**Note** The *rocksdb.\** properties at the bottom of the *hyperstore-server.properties.erb* file control the behavior and performance of the RockDB database in which object digests are stored. Do not edit these settings.

### 9.5.3. mts.properties.erb

The *mts.properties* file configures the S3 Service and the Admin Service. On each of your HyperStore nodes, the file is located at the following path by default:

```
/opt/cloudian/conf/mts.properties
```

**Do not directly edit the *mts.properties* file on individual HyperStore nodes.** Instead, if you want to make changes to the settings in this file, edit the configuration template file *mts.properties.erb* on the Puppet master node:

```
/etc/cloudian-<version>-puppet/modules/cloudians3/templates/mts.properties.erb
```

Certain *mts.properties.erb* properties take their values from settings in [common.csv](#) or from settings that you can control through the CMC's [Configuration Settings](#) page. In the *mts.properties.erb* file these properties' values are formatted as bracket-enclosed variables, like `<%= ... %>`. In the property documentation below, the descriptions of such properties indicate "Takes its value from *<location>*: *<setting>*; use that setting instead." The remaining properties in the *mts.properties.erb* file -- those that are "hard-coded" with specific values -- are settings that in typical circumstances you should have no need to edit. Therefore **in typical circumstances you should not need to manually edit the *mts.properties.erb* file.**

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

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Puppet master node you can edit this configuration file with this command:

```
$ hspkg config -e mts.properties.erb
```

Specify just the configuration file name, not the full path to the file.

In the background this invokes the Linux text editor *vi* to display and modify the configuration file. Therefore you can use the [standard keystrokes supported by vi](#) to make and save changes to the file.

**IMPORTANT !** If you do make edits to *mts.properties.erb*, be sure to push your edits to the cluster and restart the S3 Service to apply your changes. Note that restarting the S3 Service automatically restarts the Admin Service as well. For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

The *mts.properties.erb* file is divided into these sections:

## Cassandra Interfaces

The S3 Service, Admin Service, and HyperStore Service each separately maintain their own pool of connections to the Cassandra data store. The configuration settings in this section are applied separately to each of the three connection pools. For example, if *MaxActive=10*, then the S3 Service to Cassandra, Admin Service to Cassandra, and HyperStore Service to Cassandra connection pools are **each** allowed a maximum of 10 simultaneously active connections.

### *cassandra.cluster.name*

Takes its value from *region.csv*: *cassandra\_cluster\_name*. Typically you should have no need to edit that file.

### *cassandra.cluster.Hosts*

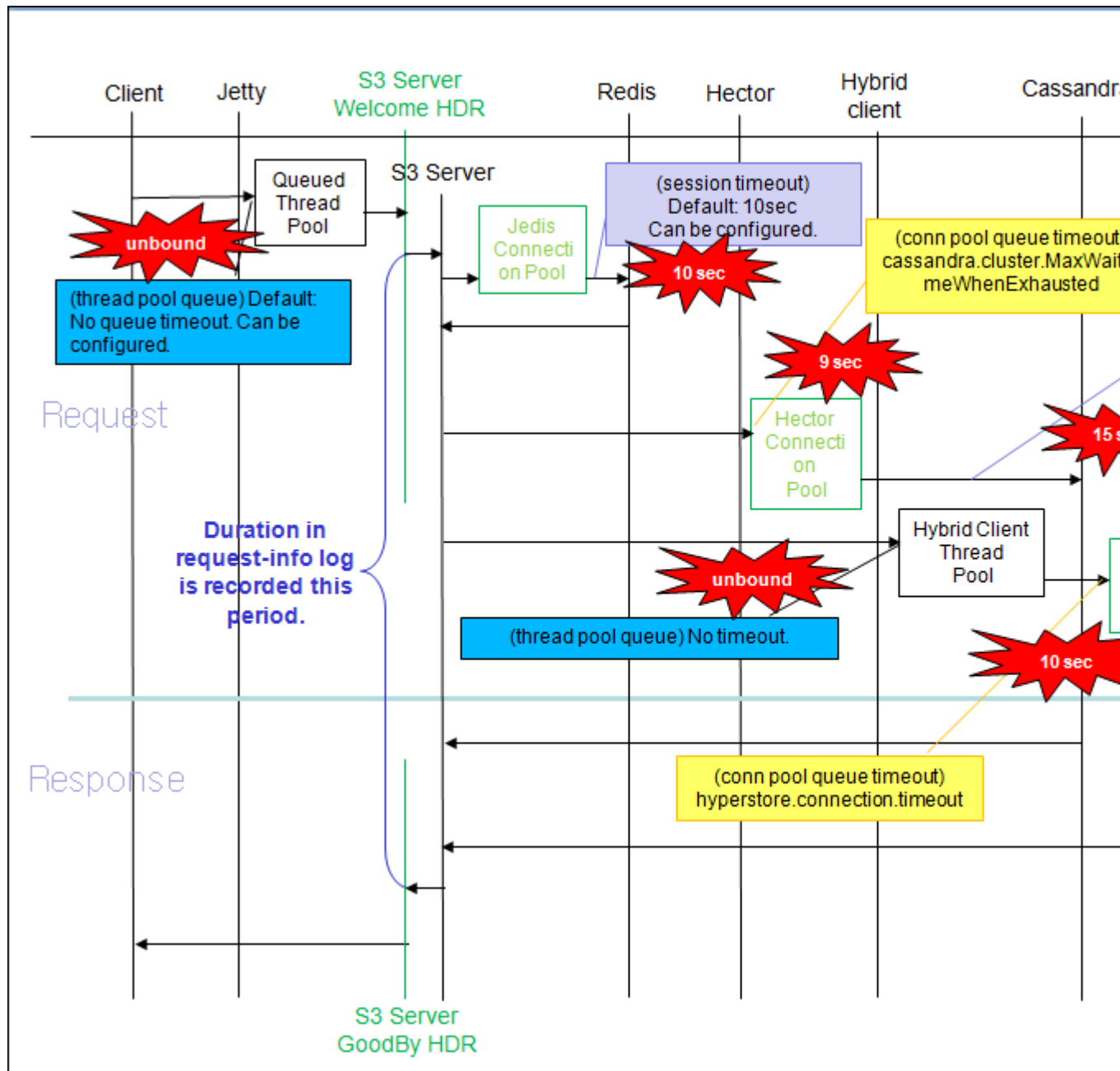
Takes its value from *topology.csv*. Typically you should have no need to edit that file.

### *cassandra.cluster.CassandraThriftSocketTimeout*

After submitting a request to a Cassandra instance via its Thrift socket, the amount of time in milliseconds to wait for a response from Cassandra before failing the operation.

Default = 15000

The diagram below shows the place of *cassandra.cluster.CassandraThriftSocketTimeout* and other timeouts within the S3 Service's request processing flow.



*cassandra.cluster.MaxActive*

Takes its value from *common.csv*: **"cassandra\_max\_active"** (page 533); use that setting instead.

*cassandra.cluster.MaxIdle*

The maximum allowed number of idle connections in a Cassandra connection pool. Any idle connections in excess of this limit are subject to being closed. Set to a negative value to disable this limit. Note this control is applicable only if *TimeBetweenEvictionRunsMillis* is set to a positive value.

Default = 20

*cassandra.cluster.MaxWaitTimeWhenExhausted*

If *cassandra.cluster.MaxActive* has been reached for a target Cassandra host and a thread requires a new

connection to the host, the thread will wait this many milliseconds before returning an error to the client.

Default = 9000

For a diagram showing the place of this timeout within the S3 request processing flow, see *cassandra.cluster.CassandraThriftSocketTimeout* (above).

#### *cassandra.cluster.RetryDownedHosts*

Whether or not to periodically retry Cassandra hosts that have been detected as being down, using a background thread. If set to "true", the retry is attempted at configurable interval *cassandra.cluster.RetryDownedHostsDelayInSeconds*.

Default = true

#### *cassandra.cluster.RetryDownedHostsQueueSize*

Maximum number of downed Cassandra hosts to maintain in the downed host retry queue at the same time. If multiple Cassandra nodes are down, and if *cassandra.cluster.RetryDownedHosts=true*, then a queue is maintained for retrying downed nodes. The *cassandra.cluster.RetryDownedHostsQueueSize* sets a limit on the number of nodes that can be in the retry queue simultaneously.

Default = -1 (unlimited)

#### *cassandra.cluster.RetryDownedHostsDelayInSeconds*

The number of seconds to wait between retry attempts for downed hosts. Applicable only if

*cassandra.cluster.RetryDownedHosts=true*.

Default = 10

#### *cassandra.cluster.Lifo*

If true, use a "last in, first out" policy for retrieving idle connections from a pool (use the most recently used idle connection). If false, use "first in, first out" retrieval of idle connections (use the oldest idle connection).

Default = true

#### *cassandra.cluster.MinEvictableIdleTimeMillis*

The minimum time in milliseconds that a connection must be idle before it becomes eligible for closing due to maximum idle connection limits.

Default = 100000

#### *cassandra.cluster.TimeBetweenEvictionRunsMillis*

The interval in milliseconds at which to check for idle connections in a pool, for enforcement of maximum idle connection limits.

Default = 100000

#### *cassandra.cluster.UseThriftFramedTransport*

Whether to use framed transport for Thrift. Strongly recommended to leave as true. If set to false, then

*thrift\_framed\_transport\_size\_in\_mb* in *cassandra.yaml* must be set to 0.

Default = true

#### *cassandra.cluster.AutoDiscoverHosts*

Whether or not to periodically check for the presence of new Cassandra hosts within the cluster and to use these same settings to interface with those new hosts. If set to true, a check for new hosts is performed at configurable interval *cassandra.cluster.AutoDiscoveryDelayInSeconds*.

Default = true

*cassandra.cluster.AutoDiscoveryDelayInSeconds*

Number of seconds to wait between checks for new hosts. Applicable only if *cassandra.cluster.AutoDiscoverHosts=true*

Default = 60

*cassandra.cluster.RunAutoDiscoveryAtStartup*

Whether or not to perform auto-discovery at cluster start-up. See *cassandra.cluster.AutoDiscoverHosts*.

Default = false

*cassandra.cluster.HostTimeoutCounter*

If a Cassandra node returns more than this many host timeout exceptions within an interval of *cassandra.cluster.HostTimeoutWindow*, mark the node as temporarily suspended. This setting and the other *HostTimeout\** settings below are applicable only if *cassandra.cluster.UseHostTimeoutTracker=true*.

Default = 3

*cassandra.cluster.HostTimeoutWindow*

If within an interval of this many milliseconds a Cassandra node returns more than *cassandra.cluster.HostTimeoutCounter* host timeout exceptions, mark the node as temporarily suspended.

Default = 1000

*cassandra.cluster.HostTimeoutUnsuspendCheckDelay*

How often to check suspended nodes list to see which nodes should be unsuspended, in seconds.

Default = 10

*cassandra.cluster.HostTimeoutSuspensionDurationInSeconds*

When the periodic check of the suspended nodes list is performed, if a node has been suspended for more than this many seconds, unsuspend the node and place it back in the available pool.

Default = 30

*cassandra.cluster.UseHostTimeoutTracker*

Whether to keep track of how often each Cassandra node replies with a host timeout exception, and to temporarily mark nodes as suspended if their timeout exceptions are too frequent. See the *HostTimeout\** setting descriptions above for details.

Default = true

*cassandra.cluster.UseSocketKeepalive*

Whether to periodically send keep-alive probes to test pooled connections to Cassandra hosts.

Default = true

*cassandra.data.directories*

Takes its value from *common.csv*: "**cassandra\_data\_directory**" (page 535); use that setting instead.

*cassandra.fs.keyspace*

Base name of the Cassandra keyspaces in which object metadata is stored. A storage policy ID is appended to this base name to create a keyspace for a particular storage policy (for example, *UserData\_b06c5f9213ae396de1a80ee264092b56*). There will be one such keyspace for each storage policy that you have configured in your system.

Default = *UserData*

*cassandra.jmx.port*

Cassandra's JMX listening port.

Default = 7199

*cassandra.tombstone\_cleanup\_threshold*

Takes its value from *common.csv*: "**cassandra\_tombstone\_cleanup\_threshold**" (page 536); use that setting instead.

*cassandra.tombstone\_gcgrace*

Takes its value from *common.csv*: "**cassandra\_tombstone\_gcgrace**" (page 536); use that setting instead.

*cloudian.repair.eventqueue.create.interval*

When [proactive repair](#) is run on a node, it reads from a queue of node-specific object write failure events that is stored in Cassandra. The object write failure events are timestamped and are bundled together based on the time interval in which they occurred. The *cloudian.repair.eventqueue.create.interval* setting controls the time span (in minutes) that's used for the bundling. For example with the default of 60 a new bundle starts each 60 minutes, if a node is unavailable for longer than this and write failure events for the node are accumulating.

When the node comes back online, the first automatic run of proactive repair will repair the objects from all the interval-based bundles except for the most current one (the in-progress interval, such as the current hour if the default of 60 is being used). That bundle will be processed at the next automatic run of proactive repair. By default proactive repair runs (if needed) every 60 minutes — this frequency is configurable by "**hyper-store.proactiverepair.poll\_time**" (page 550).

Default = 60

## Consistency Levels for System Metadata

In a distributed storage ring in which data is replicated on more than one node, there is the question of consistency requirements for read and write operations. With the HyperStore system you can configure these requirements:

- Consistency requirements for S3 object data and object metadata are managed through the creation and application of storage policies. For information about creating storage policies, "**Add a Storage Policy**" (page 353).
- Consistency requirements for system metadata stored in Cassandra -- such as usage data and monitoring data -- are configurable in the "Consistency Levels" section of *mts.properties.erb*. For information about system metadata storage, see "**System Metadata Replication**" (page 81).

*cloudian.cassandra.default.ConsistencyLevel.Read*

For the Reports keyspace (service usage data), AccountInfo keyspace (user account information), and Monitoring keyspace (system monitoring data), the consistency level to require for read operations. The reads are requested by other HyperStore components such as the S3 Service and the Admin Service.

For consistency level definitions, see **"Consistency Levels"** (page 367).

You can optionally use a comma-separated list to implement [Dynamic Consistency Levels](#).

Default = LOCAL\_QUORUM,ONE

*cloudian.cassandra.default.ConsistencyLevel.Write*

For the Reports keyspace (service usage data), AccountInfo keyspace (user account information), and Monitoring keyspace (system monitoring data), the consistency level to require for write operations. The writes are requested by other HyperStore components such as the S3 Service and the Admin Service.

For consistency level definitions, see **"Consistency Levels"** (page 367).

You can optionally use a comma-separated list to implement [Dynamic Consistency Levels](#).

Default = QUORUM,LOCAL\_QUORUM

**IMPORTANT !** Since this setting applies to writes, using "ONE" is not recommended.

*cloudian.cassandra.UserData.ConsistencyLevel.HyperStore*

The consistency level to require when [hsstool](#) is reading or writing to the *UserData\_<policyid>* keyspaces in order to implement a repair or cleanup operation. If the configured consistency requirement can't be met, the operation fails and subsequently retries. The operation will abort if two retry attempts fail to achieve the required consistency level.

For consistency level definitions, see **"Consistency Levels"** (page 367).

You can optionally use a comma-separated list to implement [Dynamic Consistency Levels](#).

Default = QUORUM

**IMPORTANT !** Since this setting applies to writes as well as reads, using "ONE" is not recommended.

*cloudian.cassandra.localdatacenter*

Takes its value from *topology.csv*. Typically you should have no reason to edit that file.

## Authorized Services and Namespaces

*cloudian.s3.authorizationV4.singleregioncheck*

In Cloudian deployments that consist of only one service region, this setting impacts the system's handling of incoming S3 requests that are using AWS Signature Version 4 Authentication:

- If this setting is set to "true", the system will validate that the region name that the client used when creating the request signature (as indicated in the scope information that the client specifies in the request) is in fact the region name for your single region. If not the request will be rejected.
- If set to "false" the system will not validate the region name. Instead, the system calculates and validates the signature using whatever region name is specified in the request.

Default = false

**Note** Do not set this property to "true" if the S3 service endpoint (URI) for your one region does not include a region name string. For example, you can set this property to "true" -- thereby enabling region name validation --if your S3 endpoint is *s3-tokyo.enterprise.com* (where *tokyo* is the region name), but not if your S3 endpoint is *s3.enterprise.com* (which lacks a region name string).

*cloudian.s3.authorizationV4.multiregioncheck*

In Cloudian deployments that consist of multiple service regions, this setting impacts the system's handling of incoming S3 requests that are using AWS Signature Version 4 Authentication:

- If this setting is set to "true", the system will validate that the region name that the client used when creating the request signature (as indicated in the scope information that the client specifies in the request) is in fact the region name for the region to which the request has been submitted. If not the request will be rejected.
- If set to "false" the system will not validate the region name. Instead, the system calculates and validates the signature using whatever region name is specified in the request.

Default = false

**IMPORTANT !** Do not set this property to "true" if the S3 service endpoints (URIs) for your regions do not include region name strings. For example, you can set this property to "true" -- thereby enabling region name validation --if your S3 endpoints are *s3-tokyo.enterprise.com* and *s3-osaka.enterprise.com* (where *tokyo* and *osaka* are region names), but not if your S3 endpoints lack region name strings.

*cloudian.s3.serverside.encryption.keylength*

Takes its value from *common.csv*: "**cloudian\_s3\_aes256encryption\_enabled**" (page 533); use that setting instead.

## QoS

The following settings pertain to the implementation of quality of service (QoS) limits for the S3 service. For more detail on the HyperStore QoS feature, see "**Quality of Service (QoS) Feature Overview**" (page 135).

*cloudian.s3.qos.enabled*

Takes its value from "**Enforce Configured QoS Limits for Storage Utilization**" (page 345) in the CMC's [Configuration Settings](#) page; use that setting instead.

*cloudian.s3.qos.rate.enabled*

Takes its value from "**Enforce Configured QoS Limits for Request Rates and Data Transfer Rates**" (page 346) in the CMC's [Configuration Settings](#) page; use that setting instead.

*cloudian.s3.qos.cache.enabled*

When QoS enforcement is enabled, for each S3 request the S3 Service checks current user and group QoS usage against configured QoS limits. The usage counters and configured limits are stored in the Redis QoS DB. This setting if set to "true" enables the S3 Service to cache QoS counter and limits data and to check that cached data first when performing its QoS enforcement role. If there is no cache hit then Redis is checked.

Default = true

Reloadable via JMX (S3 Service's JMX port 19080; MBean attribute = com.gemini.cloudian.s3 → Configuring → Attributes → QosCacheEnabled)

*cloudian.s3.qos.cache.expiryms*

For the S3 Service's QoS data cache (if enabled), the cache entry expiry time in milliseconds.

Default = 10000

Reloadable via JMX (S3 Service's JMX port 19080; MBean attribute = com.gemini.cloudian.s3 → Configuring → Attributes → QosCacheExpiryMs)

*cloudian.s3.qos.storagewrite.batch.enabled*

If set to "true" then the AccountHandler's updates of storage object and storage bytes counters in Redis are batched together.

Default = false

Reloadable via JMX (S3 Service's JMX port 19080; MBean attribute = com.gemini.cloudian.s3 → Configuring → Attributes → QosStorageWriteEnabled)

*cloudian.s3.qos.storagewrite.intervals*

If batching of storage counter updates is enabled, the batch interval in milliseconds.

Default = 60000

Reloadable via JMX (S3 Service's JMX port 19080; MBean attribute = com.gemini.cloudian.s3 → Configuring → Attributes → QosStorageWriteIntervalMs)

*cloudian.s3.qos.maxdelay*

If a PUT Object or PUT Object Copy operation that is overwriting an existing object takes more than this many milliseconds to complete, then before the system updates the user's Storage Bytes (SB) count it re-checks the existing object's metadata to ensure that the calculated size differential between the existing object and the new object is based on fresh metadata. This configurable behavior is intended to reduce the likelihood of erroneous SB counts resulting from race conditions wherein multiple client requests are overwriting the same object at the same time.

The context is that in the case of an existing object being overwritten by a new instance of the object, the system updates the user's SB count by calculating the delta between the original object's size (as indicated by its metadata) and the new object's size and then applying that difference to the user's SB count. It's important therefore that the calculated delta be based on up-to-date metadata for the object that's being overwritten, even in the case where the writing of the new object takes a long time to complete.

Default = 60000

## S3 Service Caching (Bucket Metadata, etc)

*cloudian.s3.metadata.cache.enabled*

If set to *true*, bucket metadata is cached by the S3 Service. When S3 request processing requires bucket metadata, the cache is checked first. If there is no cache hit then the needed metadata is retrieved from Redis, and is cached for subsequent use.

On an ongoing basis, the system detects if the corresponding source metadata in Redis changes and then automatically invalidates the cached metadata.

Default = true

Reloadable via JMX (S3 Service's JMX port 19080; MBean attribute = com.gemini.cloudian.s3 → Configuring → Attributes → MDCacheEnabled)

*cloudian.s3.group.cache.enabled*

If set to *true*, then user group metadata is cached by the S3 Service. When S3 request processing requires group status, the cache is checked first. If there is no cache hit then the needed metadata is retrieved from Redis, and is cached for subsequent use.

On an ongoing basis, the system detects if the corresponding source metadata in Redis changes and then automatically invalidates the cached metadata.

Default = true

Reloadable via JMX (S3 Service's JMX port 19080; MBean attribute = com.gemini.cloudian.s3 → Configuring → Attributes → GroupCacheEnabled)

*cloudian.s3.credentials.cache.enabled*

If set to *true*, users active security credentials are cached by the S3 Service. When S3 request processing requires a user's credentials, the cache is checked first. If there is no cache hit then the needed credentials retrieved from Redis, and are cached for subsequent use.

On an ongoing basis, the system detects if the corresponding source credentials in Redis change and then automatically invalidates the cached credentials.

Default = true

Reloadable via JMX (S3 Service's JMX port 19080; MBean attribute = com.gemini.cloudian.s3 → Configuring → Attributes → GroupCacheEnabled)

*cloudian.s3.user.cache.enabled*

If set to *true*, user metadata is cached by the S3 Service. When S3 request processing requires a user's status and/or display name, the cache is checked first. If there is no cache hit then the needed metadata is retrieved from Redis, and is cached for subsequent use.

On an ongoing basis, the system detects if the corresponding source metadata in Redis changes and then automatically invalidates the cached metadata.

Default = true

Reloadable via JMX (S3 Service's JMX port 19080; MBean attribute = com.gemini.cloudian.s3 → Configuring → Attributes → UserCacheEnabled)

## Domains and Regions

The "region" settings in this section are applicable to multi-region HyperStore deployments. If your HyperStore system does not have multiple service regions, you can leave the settings in this section at their default values.

*cloudian.s3.regions*

Takes its value from *common.csv*: **"regions"** (page 513); use that setting instead.

*cloudian.s3.home\_region*

Takes its value from *region.csv*: *region\_name*; use that setting instead.

*cloudian.s3.default\_region*

Takes its value from *common.csv*: **"default\_region"** (page 513); use that setting instead.

*cassandra.cluster.name.<region>*

Takes its value from *region.csv*: *cassandra\_cluster\_name*. Typically you should have no reason to edit that setting.

*cassandra.cluster.hosts.<region>*

Takes its value from *region.csv*: *cassandra\_cluster\_hosts*. Typically you should have no reason to edit that setting.

*cloudian.s3.domain.<region>*

Takes its value from *region.csv*: *s3\_domain\_and\_port*. Typically you should have no reason to edit that setting.

*cloudian.s3.ssl\_domain.<region>*

Domain and SSL listening port from one of your regions, in format *<FQDN>.<port>*. This is used if your system has multiple service regions and you have configured your S3 service to use SSL. For example, if a GET Object request comes in to the S3 service in region1, and the object is stored in region2, and region2 is configured to use SSL for its S3 service — then the S3 service in region1 needs to know the domain and SSL port for the S3 service in region2, so that it can specify a correct Location header in the Redirect message it returns to the requesting client.

If you have a multi-region HyperStore deployment, the installation script automatically configures the *cloudian.s3.ssl\_domain.<region>* setting for **each** of your regions, including the local region. For example:

```
cloudian.s3.ssl_domain.region1 = s3.region1.mycompany.com:443
cloudian.s3.ssl_domain.region2 = s3.region2.mycompany.com:443
cloudian.s3.ssl_domain.region3 = s3.region3.mycompany.com:443
```

If you have only one region in your HyperStore deployment, then this setting is not relevant to your system.

Default = commented out

*cloudian.s3.website\_endpoint*

Takes its value from *region.csv*: *cloudians3\_website\_endpoint*.

To change this, use the HyperStore **"Installer Advanced Configuration Options"** (page 501).

*cloudian.util.dns.resolver*

Method used to resolve foreign buckets to the correct region, in support of the S3 LocationConstraint feature. Currently only one option is supported:

- *com.gemini.cloudian.util.dns.RedirectResolver* — Never resolve foreign bucket domain names, always reply with a 307 redirect. The Location value to supply in the redirect response is obtained from the global Redis database.

Default = *com.gemini.cloudian.util.dns.RedirectResolver*

*cloudian.publicurl.port*

Takes its value from *common.csv*: *ld\_cloudian\_s3\_port*, which is controlled by the installer. To change S3 listening ports use the installer's Advanced Configuration options.

*cloudian.publicurl.sslport*

Takes its value from *common.csv*: *ld\_cloudian\_s3\_ssl\_port*, which is controlled by the installer. To change S3 listening ports use the installer's Advanced Configuration options.

## S3 Service

*cloudian.s3.server*

HTTP Server header value to return in responses to S3 requests. If you want no Server header value returned in responses to S3 requests, uncomment this setting and set it to empty.

Default = Commented out; uses internal default "CloudianS3"

*cloudian.s3.tmpdir*

The directory in which to temporarily store large files in order to reduce memory usage.

Default = Commented out; uses internal default *java.io.tmpdir*.

*cloudian.fs.read\_buffer\_size*

When interacting with the file storage system in Cassandra, the size of the S3 Service's and Admin Service's read buffer, in bytes. A larger read buffer can enhance performance but will also consume more memory.

Default = 65536

Reloadable via JMX (S3 Service's JMX port 19080; MBean attribute = com.gemini.cloudian.s3 → Configuring → Attributes → ReadBufferSize)

*cloudian.s3.putobject.max\_size*

Takes its value from the **"Put Object Maximum Size (Bytes)"** (page 350) setting on the CMC's [Configuration Settings](#) page; use that setting instead.

*cloudian.s3.getbucket.max\_num\_of\_keys*

When performing an S3 GET Bucket operation (which returns meta-data about the objects in the bucket specified in the request), the maximum number of objects to list in the response. If the client request sets a "max-keys" parameter, then the lower of the client-specified value and the *cloudian.s3.getbucket.max\_num\_of\_keys* value is used.

Default = 1000

Reloadable via JMX (S3 Service's JMX port 19080; MBean attribute = com.gemini.cloudian.s3 → Configuring → Attributes → GetBucketMaxKeys)

*cloudian.s3.max\_user\_buckets*

Takes its value from the **"Maximum Buckets Per User"** (page 350) setting on the CMC's [Configuration Settings](#) page; use that setting instead.

*cloudian.s3.delimiter\_regex*

Regular expression indicating allowed delimiters in getBucketList objects.

Default = .+

Reloadable via JMX (S3 Service's JMX port 19080; MBean attribute = com.gemini.cloudian.s3 → Configuring → Attributes → DelimiterRegex)

*cloudian.s3.multipart.maxparts*

Takes its value from the **"Multipart Upload Maximum Parts"** (page 350) setting on the CMC's [Configuration Settings](#) page; use that setting instead.

*cloudian.s3.multipart.minpartsize*

In an S3 Multipart Upload submitted to HyperStore, the required minimum size per part, excluding the last part. Expressed in number of bytes. The operation will fail if a part other than the last part is smaller than this many bytes. For example, if you set this to 5242880 (5MB, which is the Amazon S3 default for minimum part size) then in a Multipart Upload each part uploaded must be at least 5MB in size, except for the last part which is allowed to be as small as necessary to complete the object.

The HyperStore default of 1 byte essentially places no restriction on minimum part size.

Default = 1

*cloudian.s3.unsupported*

Comma-separated list of Amazon S3 URI parameters that the HyperStore system does not support. This list applies across HTTP methods and across S3 resource types. In response to requests that include an unsupported URI parameter, the HyperStore system will return 501, Not Implemented.

Default = accelerate,requestPayment,analytics,inventory,metrics,select,notification

*cloudian.util.ntp.Path*

Path to *ntpstat*. Setting this value is required only if *ntpstat* is not in the environment PATH.

Default = commented out; internal default = ''

*cloudian.util.ntp.MaximumSynchronizationDistance*

Maximum system clock skew to allow when servicing S3 requests that require the S3 Service to generate an object versionId, in milliseconds.

- If this setting is set to a positive value, a S3 Service node that is processing S3 requests that require generating a versionId will perform an *ntpstat* check. If the node's system clock is skewed by more than *cloudian.util.ntp.MaximumSynchronizationDistance* milliseconds, the S3 Service rejects the S3 request with a "503 Service Unavailable" error response. The frequency of the *ntpstat* check can be limited by using the *cloudian.util.ntp.CheckIntervalMillis* setting.
- If this setting is set to 0, the *ntpstat* check is not performed when processing S3 requests that require generating a versionId.

Default = 0

*cloudian.util.ntp.CheckIntervalMillis*

Minimum time between *ntpstat* checks, in milliseconds.

- If this setting is set to a positive value, then when the S3 Service is processing S3 requests that require generating a versionId, an *ntpstat* check will be performed only if *cloudian.util.ntp.CheckIntervalMillis* milliseconds or more have passed since the last *ntpstat* check was performed.
- If this setting is set to 0, an *ntpstat* check is performed with every S3 request that requires generating a versionId.

Default = 60000

*cloudian.s3.batch.delete.delay*

When executing the batch processing to purge deleted objects from disk (see *cloudian.delete.queue.poll.interval* below for background information), the number of milliseconds to pause in between purges of individual objects.

If `cloudian.s3.batch.delete.delay` is set to 0 (as it is by default), then when the batch processing job is running there is no delay between individual object purges.

If you wish you can use the `cloudian.s3.batch.delete.delay` property to "throttle" the execution of the deleted object batch processing job (to slow it down so as to reduce its resource demands).

Default = 0

#### *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 Cassandra. But the actual purging of object data from disk does not occur until a batch processing job runs on each node, to physically purge the objects that have been marked as deleted. The batch processing execution is distributed across the cluster, with each node being responsible for processing the object deletions that result from S3 requests processed by that node.

The `cloudian.delete.queue.poll.interval` 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.

Default = 60

## S3 Client

The settings in this section are used by an internal S3 Service client that sends requests to the HyperStore S3 Service in other regions of a [multi-region HyperStore deployment](#). These settings are relevant only if you have a multi-region system.

**Note** In the configuration file commenting in this section, the specified "defaults" are internal defaults that would be used if these settings were commented out in the file. In the documentation below, the specified "defaults" are the values that are assigned to the settings in the default version of the configuration file after install.

#### *cloudian.s3.client.ConnectionTimeout*

When interfacing with the S3 Service in a different Cloudian HyperStore service region, the connection establishment timeout in milliseconds.

Default = 2000

Reloadable via JMX (S3 Service's JMX port 19080; MBean attribute = `com.gemini.cloudian.s3` → Configuring → Attributes → `S3ClientConnectionTimeout`)

#### *cloudian.s3.client.SocketTimeout*

When interfacing with the S3 Service in a different Cloudian HyperStore service region, 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

Reloadable via JMX (S3 Service's JMX port 19080; MBean attribute = `com.gemini.cloudian.s3` → Configuring → Attributes → `S3ClientSocketTimeout`)

#### *cloudian.s3.client.MaxErrorRetry*

When interfacing with the S3 Service in a different Cloudfian HyperStore service region, the maximum number of times to retry a retryable failed request (such as when a 5xx response is received). Set to 0 if you want no retries.

Default = 0

Reloadable via JMX (S3 Service's JMX port 19080; MBean attribute = com.gemini.cloudfian.s3 → Configuring → Attributes → S3ClientMaxErrorRetry)

*cloudfian.s3.client.UserAgent*

When interfacing with the S3 Service in a different Cloudfian HyperStore service region, the value of the first part of the HTTP User-Agent header, where the whole header is "<ConfigValue>, <AWS SDK Version> <OS Version> <JDK Version>". For example, if you set this setting to "Agent99", then the resulting User-Agent header will be "Agent99, <AWS SDK Version> <OS Version> <JDK Version>", with the latter three values being populated automatically by the system.

Default = Commented out; uses internal default "Cloudfian/<version> <default-region>"

Reloadable via JMX (S3 Service's JMX port 19080; MBean attribute = com.gemini.cloudfian.s3 → Configuring → Attributes → S3ClientUserAgent)

## Reports

*cloudfian.auditlog.enabled*

Do not edit this setting.

*reports.raw.ttl*

Time-to-live for "raw" S3 usage data in the Reports keyspace in Cassandra, in seconds. Raw service usage data will be automatically deleted this many seconds after its creation. This is set by the system -- **do not manually edit** this setting.

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 =

- 604800 (seven days), if the **"Track/Report Usage for Request Rates and Data Transfer Rates"** (page 344) setting in the CMC's [Configuration Settings](#) page is set to "false" (as it is by default)
- 86400 (one day) if the **"Track/Report Usage for Request Rates and Data Transfer Rates"** (page 344) setting is set to "true"

For an overview of how the HyperStore system tracks service usage by groups, users, and buckets, see **"Usage Reporting and Billing Feature Overview"** (page 138).

*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)

**IMPORTANT !** This hourly rollup data is the basis for generating billing reports. After hourly rollup data is deleted it is no longer available for generating billing reports.

#### *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)

#### *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)

#### *reports.auditdata.ttl*

Time-to-live for audit data in the Reports keyspace in Cassandra, in seconds. Audit data will be automatically deleted this many seconds after its creation.

Default = 5616000 (65 days)

#### *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.

**Note** As soon an alert occurs a record of it is written to the Smart Support log (which by default is uploaded to Cloudfian 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 190).

Default = 86400 (one day)

#### *monitoring.ophistory.ttl*

Time-to-live for per-node data repair and cleanup operation status summaries in Cassandra's Monitoring keyspace, in seconds. Each repair and cleanup status summary will be automatically deleted after it has been stored for *monitoring.ophistory.ttl* seconds.

For as long as repair and cleanup status summaries are stored in the Monitoring keyspace, they can be retrieved on a per-node basis by using the [hsstool opstatus](#) command, with the "-q history" option. This returns the operation history of a specified node. The *monitoring.ophistory.ttl* property controls the maximum

length of that retrievable operation history -- for example with *monitoring.ophistory.ttl* at its default value, for each HyperStore node you can retrieve the history of its repair and cleanup operations from the past 90 days. Repair and cleanup operation status information older than that will be deleted.

Default = 7776000 (90 days)

## Usage

The "Usage" section has settings for tuning the performance of [usage data repair](#) operations.

### *usage.repair.row.size*

When querying Cassandra for the object metadata associated with individual service users, the maximum number of users to retrieve per query.

Default = 1000

### *usage.repair.column.size*

When querying Cassandra for the object metadata associated with individual service users, for each user the maximum number of objects for which to retrieve metadata per query.

Default = 1000

### *usage.repair.maxdirtyusers*

Maximum number of "dirty" users for whom to verify (and if necessary repair) usage data during a single run of the [POST /usage/repair/dirtyusers](#) operation. This operation is triggered every 12 hours by cron job.

Default = 1000

### *usage.rollup.userchunk.size*

When performing rollups of usage data, the maximum number of users to take into memory at a time.

Default = 1000

### *usage.rollup.usagechunk.size*

When performing rollups of usage data, the maximum number of usage records to take into memory at a time.

Default = 1000

### *usage.rollup.hour.maxretry*

Each time the system performs an hourly rollup of usage data for the hour that just ended, it will check whether the hourly rollup data from the **preceding** hours exists (as it should, unless relevant system services have been down or unreachable). If the hourly rollup data from preceding hours is missing, then the system retries processing the hourly rollups for those hours that are missing the hourly rollup data. The *usage.rollup.hour.maxretry* property sets a maximum on the number of preceding hours to check on and (if needed) perform a retry for.

For example, suppose *usage.rollup.hour.maxretry*=6. With this setting, if the system is for example about to perform the hourly rollup from the 10th hour of the day, it will first check that hourly rollup data exists for the 9th, 8th, 7th, 6th, 5th, and 4th hours of the day -- and if any of those hourly rollups are missing, the system will try again to execute those hourly rollups. After doing so the system will then perform the hourly rollup of the usage data from the 10th hour of the day.

Default = 24

*cloudian.s3.usagerates.enabled*

Takes its value from the **"Track/Report Usage for Request Rates and Data Transfer Rates"** (page 344) setting on the CMC's [Configuration Settings](#) page; use that setting instead.

*bucketstats.enabled*

Takes its value from *common.csv*: **"bucketstats\_enabled"** (page 543); use that setting instead.

*cloudian.s3.redis.retry*

The interval in seconds at which the S3 Service will retry a Redis node that has been unresponsive. If the S3 Service finds a Redis node to be unresponsive the S3 Service will temporarily remove that node from the list of Redis nodes that are available to service requests. At an interval of *cloudian.s3.redis.retry* seconds the S3 Service will retry the Redis node. If it's found to be responsive, the node is added back to the S3 Service's list of available Redis nodes.

This setting is applicable to Redis Credentials nodes and Redis QoS nodes.

Default = 30

## Redis Monitor

The HyperStore **"Redis Monitor Service"** (page 27) is installed automatically with the S3 Service and Admin Service. You can run the Redis Monitor in order to monitor Redis cluster health and implement automatic fail-over of the Redis master node role. For Redis Monitor redundancy, it runs on two of your S3 Service / Admin Service nodes, with the Monitor configured as primary on one node and as backup on the other node.

*redis.monitor.subscription.check*

Takes its value from *common.csv*: **"redis\_monitor\_subscription\_check"** (page 533); use that setting instead.

*redis.monitor.primary.pollInterval*

The interval at which the backup Redis Monitor instance should check on the health of the primary instance, in seconds. If the primary Redis Monitor instance is unresponsive, the backup instance takes over the monitoring duties.

Default = 5

*redis.credentials.cluster.pollInterval*

Interval at which the Redis Monitor application should check the health of the Redis Credentials servers, in seconds. At this interval, the Redis Monitor also checks the S3 Service / Admin Service nodes via JMX to ensure that they are configured to point to the current Redis Credentials master, and updates their configuration if necessary.

Default = 5

*redis.credentials.cluster.client.request.waittime*

Maximum time for the Redis Monitor to wait for a JMX connection attempt to a S3 Service / Admin Service node to complete, in seconds. If the connection attempt doesn't complete (with a success or failure result) within this interval, the Redis Monitor marks the S3 Service / Admin Service node as DOWN and writes an INFO level message to *cloudian-redismon.log*. Meanwhile, the connection attempt will continue until completion, and subsequently polling of the S3 Service / Admin Service node will resume at the regular polling interval.

The `redis.credentials.cluster.client.request.waittime` value must be smaller than the `redis.credentials.cluster.pollInterval` value.

Default = 3

#### *redis.monitor.alert.limit*

This setting limits how much the Redis Monitor's monitoring for conditions of DC partition or "split brain" writes to `cloudian-redismon.log`.

Default = 100

#### *redis.monitor.skip.dc.monitoring*

For information about this setting please see "**disable dc partition monitoring**" (page 735) and "**enable dc partition monitoring**" (page 736).

Default = true (monitoring is disabled ["skipped"])

#### *redis.monitor.skip.brain.monitoring*

For information about this setting please see "**disable split brain monitoring**" (page 737) and "**enable split brain monitoring**" (page 738).

Default = false (monitoring is enabled [not "skipped"])

## Credentials

### *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, by the `credentials.iamuser.max` property.

Default = 5

### *credentials.iamuser.max*

Maximum allowed number of S3 credentials per IAM user. Each credential is a key pair consisting of a public key (access key) and a private key (secret key). These credentials enable an IAM user to access the HyperStore S3 storage system through a third party S3 client. IAM users cannot access the HyperStore S3 storage system through the CMC.

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's **IAM User** section (which is accessible only to HyperStore group administrators or regular users -- not system administrators) or the HyperStore implementation of the Amazon IAM API.

Default = 2

*keystore.pass*

Password for the Java keystore file `/opt/cloudian/conf/keystore`. This keystore file stores the Admin Service's pre-generated, self-signed, RSA-based public and private keys for SSL.

Default = adminpass

*secure.transact.alias*

Alias identifying the Admin Service's certificate entry within the keystore.

Default = secure

*secure.transact.pass*

Password to access the certificate entry that's identified by `secure.transact.alias`.

Default = private

*admin.auth.realm*

Takes its value from `common.csv`: **"admin\_auth\_realm"** (page 526); use that setting instead.

*admin.auth.enabled*

Takes its value from `common.csv`: **"admin\_auth\_enabled"** (page 526); use that setting instead.

*admin.secure*

Takes its value from `common.csv`: **"admin\_secure"** (page 526); use that setting instead.

*admin.user.password.length*

Maximum allowed character length for users' Cloudian Management Console login passwords.

Default = 64

*user.password.min.length*

Takes its value from `common.csv`: **"user\_password\_min\_length"** (page 527); use that setting instead.

*user.password.dup.char.ratio.limit*

Takes its value from `common.csv`: **"user\_password\_dup\_char\_ratio\_limit"** (page 527); use that setting instead.

*user.password.unique.generations*

Takes its value from `common.csv`: **"user\_password\_unique\_generations"** (page 527); use that setting instead.

*user.password.rotation.graceperiod*

Takes its value from `common.csv`: **"user\_password\_rotation\_graceperiod"** (page 527); use that setting instead.

*user.password.rotation.expiration*

Takes its value from `common.csv`: **"user\_password\_rotation\_expiration"** (page 527); use that setting instead.

*awsmms.proxy.host*

Takes its value from `common.csv`: **"awsmmsproxy\_host"** (page 542); use that setting instead.

*awsms.proxy.port*

If you are using the AWS Marketplace Metering Service version of Clouidian HyperStore, this is the port that HyperStore connects to on your AWS Proxy Server. HyperStore transmits usage data to the AWS Proxy Server, which in turn relays the data to the AWS Marketplace Metering Service for billing purposes.

Default = 17081

*admin.whitelist.enabled*

Takes its value from *common.csv*: **"admin\_whitelist\_enabled"** (page 537); use that setting instead.

## HyperStore Service

The HyperStore Service section configures S3 Service behavior in its role as a client of the [HyperStore Service](#).

*hyperstore.endpoint*

HyperStore Service listening port to which the S3 Service will submit data operation requests.

Default = Takes its value from elsewhere within the Puppet manifest structure; default is 19090

*hyperstore.maxthreads.read*

Takes its value from *common.csv*: **"hyperstore.maxthreads.read"** (page 519); use that setting instead.

*hyperstore.maxthreads.write*

Takes its value from *common.csv*: **"hyperstore.maxthreads.write"** (page 519); use that setting instead.

*hyperstore.maxthreads.repair*

Takes its value from *common.csv*: **"hyperstore.maxthreads.repair"** (page 518); use that setting instead.

*hyperstore.snd.buffer*

Socket send buffer size from S3 nodes to HyperStore nodes.

Default = 0

*hyperstore.rcv.buffer*

Socket receive buffer size from S3 nodes to HyperStore nodes.

Default = 0

*hyperstore.timeout*

Takes its value from *common.csv*: **"hyperstore\_timeout"** (page 517); use that setting instead.

*hyperstore.connection.timeout*

Takes its value from *common.csv*: **"hyperstore\_connection\_timeout"** (page 518); use that setting instead.

*hyperstore.maxtotalconnections*

Takes its value from *common.csv*: *hyperstore\_maxtotalconnections*, which is controlled by a performance optimization script that runs automatically when you install your cluster or resize your cluster.

*hyperstore.maxperrouteconnections*

Takes its value from *common.csv*: *hyperstore\_maxperrouteconnections*, which is controlled by a performance

optimization script that runs automatically when you install your cluster or resize your cluster

*cassandra.range\_repair.max.waiting.time.in\_sec*

During a Cassandra repair operation each of a node's token ranges are repaired one at a time, sequentially. The system will wait for a maximum of this many seconds for repair of a range to complete. If repair of a range 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

## Process Diagnostics (Phone Home / Smart Support)

*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 Cloudian Support via the S3 protocol once a day, as part of the Smart Support feature. For more information about this feature -- including what data gets sent to Cloudian Support and how they use it for your benefit -- see **"Smart Support and Diagnostics Feature Overview"** (page 190).

- **If you want diagnostics data automatically uploaded to Cloudian Support via S3 each day**, there's nothing you need to do -- just leave *phonehome.enabled* set to "true" and leave the setting *common.csv: phonehome\_uri* set to the Cloudian Support S3 URI. This is the default behavior and is the recommended behavior.
- **If you want diagnostics data automatically uploaded each day to an S3 destination other than Cloudian Support**, leave *phonehome.enabled* set to "true" and set the *common.csv: phonehome\_uri* setting to the desired S3 URI (and also set *common.csv: phonehome\_{bucket, access\_key, secret\_key}*).
- **If you do not want diagnostics data automatically uploaded to an S3 destination each day**, set *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 *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*.

**Note** The deletion of old diagnostics packages is managed by Puppet, as configured by **"cleanup\_directories\_byage\_withmatch\_timelimit"** (page 515) in *common.csv*. By default Puppet deletes the diagnostics packages after they are 15 days old. This presumes that you have left the Puppet daemons running in your HyperStore cluster, which is the default behavior. If you do not leave the Puppet daemons running the diagnostics logs will not be automatically deleted. In that case you should delete the old packages manually, since otherwise they will eventually consume a good deal of storage space.

Default = true

*phonehome.uri*

Takes its value from *common.csv: "phonehome\_uri"* (page 524); use that setting instead.

*phonehome.{bucket, accessKey, secretKey}*

Take their values from *common.csv*: "**phonehome\_bucket**" (page 524) and the subsequent settings; use those settings instead.

Default = empty

*phonehome.proxy.host*

Takes its value from *common.csv*: "**phonehome\_proxy\_host**" (page 523); use that setting instead.

*phonehome.proxy.port*

Takes its value from *common.csv*: "**phonehome\_proxy\_port**" (page 524); use that setting instead.

*phonehome.proxy.username*

Takes its value from *common.csv*: "**phonehome\_proxy\_username**" (page 524); use that setting instead.

*phonehome.proxy.password*

Takes its value from *common.csv*: "**phonehome\_proxy\_password**" (page 524); use that setting instead.

*phonehome.gdpr*

Takes its value from *common.csv*: "**phonehome\_gdpr**" (page 525); use that setting instead.

## System Info Log Upload

*sysinfo.uri*

S3 URI to which to upload on-demand Node Diagnostics packages, when you use the CMC's [Collect Diagnostics](#) function. By default this is the S3 URI for Cloudian Support, but if you prefer you can set this to a different S3 URI. For an overview of this feature see "**Smart Support and Diagnostics Feature Overview**" (page 190).

Include the HTTP or HTTPS protocol part of the URI (*http://* or *https://*).

Default = `https://s3-support.cloudian.com:443`

**Note** If you set *sysinfo.uri* to a URI for your own HyperStore S3 storage system (rather than Cloudian Support), and if your S3 Service is using HTTPS, then your S3 Service's SSL certificate must be a CA-verified, trusted certificate — not a self-signed certificate. By default the Node Diagnostics upload 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 Cloudian Support.

*sysinfo.{bucket, accessKey, secretKey}*

- If you are using the Node Diagnostics upload feature to upload node diagnostic data to **Cloudian Support** via S3, you can leave the *sysinfo.bucket*, *sysinfo.accessKey*, and *sysinfo.secretKey* properties empty. The Node Diagnostics feature will automatically extract the Cloudian Support S3 bucket name and security credentials from your encrypted HyperStore license file. You would only modify these configuration properties if instructed to do so by Cloudian Support.
- If you set *sysinfo.uri* to **your own S3 URI** (rather than the Cloudian Support URI), set the *sysinfo.bucket*, *sysinfo.accessKey*, and *sysinfo.secretKey* properties to the destination bucket name and the applicable S3 access key and secret key.

Default = empty

*sysinfo.proxy.host*

Takes its value from *common.csv*: "**phonehome\_proxy\_host**" (page 523); use that setting instead.

**Note** This property, together with the other *sysinfo.proxy.\** properties below, is for using a local forward proxy when sending Node Diagnostics packages to Cloudian Support (or another external S3 destination). By default these properties will inherit the same *common.csv* values that you set for proxying of the daily Smart Support upload (also known as "phone home"). If you want to use a different proxy for sending Node Diagnostics packages -- not the same proxy settings that you use for the phone home feature -- edit the *sysinfo.proxy.\** settings directly in *mts.properties.erb*. For example you could change *sysinfo.proxy.host=<%= @phonehome\_proxy\_host %>* to *sysinfo.proxy.host=proxy2.enterprise.com*.

*sysinfo.proxy.port*

Takes its value from *common.csv*: "**phonehome\_proxy\_port**" (page 524); use that setting instead.

*sysinfo.proxy.username*

Takes its value from *common.csv*: "**phonehome\_proxy\_username**" (page 524); use that setting instead.

*sysinfo.proxy.password*

Takes its value from *common.csv*: "**phonehome\_proxy\_password**" (page 524); use that setting instead.

*s3.client.timeout*

When uploading a Node Diagnostics package to an S3 destination such as Cloudian Support, the socket timeout in milliseconds.

Default = 1800000

*s3.upload.part.minsize*

When HyperStore uses Multipart Upload to transmit a Node Diagnostics package to an S3 destination such as Cloudian Support, 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 18MB object, and the configured minimum part size is 5MB, the object will be transmitted in four parts of size 5MB, 5MB, 5MB, and 3MB.

Default = 5242880 (5MB)

*s3.upload.part.threshold*

When HyperStore transmits a Node Diagnostics package to an S3 destination such as Cloudian Support, it uses Multipart Upload if the package is larger than this many bytes.

Default = 16777216 (16MB)

## Protection Policies (Storage Policies)

*cloudian.protection.policy.max*

Maximum number of bucket protection policies (storage policies) that the system will support. Policies with status "Active", "Pending", or "Disabled" count toward this system limit.

If the policy maximum has been reached, you will not be able to create new policies until you either delete existing policies or increase the value of *cloudian.protection.policy.max*.

Default = 25

Reloadable via JMX (S3 Service's JMX port 19080; MBean attribute = com.gemini.cloudian.s3 → Configuring → Attributes → MaxProtectionPolicies)

For more information about storage policies, see **"Storage Policies Feature Overview"** (page 76).

## User Agent for Tiering

*cloudian.tiering.useragent*

Takes its value from *common.csv*: **"cloudian\_tiering\_useragent"** (page 530); use that setting instead.

## SSL for SSEC

*cloudian.s3.ssec.usessl*

This setting controls whether the S3 servers 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 servers require 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 your S3 servers -- and the load balancer, when receiving an incoming HTTPS request from clients, terminates the SSL and uses regular HTTP to connect to an S3 server over your internal network.
- You trust your internal network to safely transport users' encryption keys from the load balancer to the S3 servers over regular HTTP.

For background information about HyperStore support for server-side encryption (SSE and SSE-C), see **"Server-Side Encryption"** (page 105).

Default = true

## AWS Region for AWS-KMS

*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 110).

Default = us-east-1

## Tracker for Torrent

*cloudian.s3.torrent.tracker*

If you want your service users to be able to use BitTorrent for object retrieval, use this property to specify the URL of a BitTorrent "tracker" (a server that keeps track of the clients that have retrieved a particular object and makes this information available to other clients retrieving the object). This can be a tracker that you implement yourself or one of the many public BitTorrent trackers. HyperStore itself does not provide a tracker.

This must be a single URL, not a list of URLs.

The tracker URL that you specify here will be included in the torrent file that the HyperStore S3 Server returns to clients when they submit a **"GetObjectTorrent"** (page 956) request.

Default = commented out

## Elasticsearch Integration

*cloudian.elasticsearch.\**

For information about the *cloudian.elasticsearch.process.type* and *cloudian.elasticsearch.http.url* settings, see **"Option to Send Metadata to an HTTP Server Rather than to Elasticsearch"** (page 175).

For information about all the other *cloudian.elasticsearch.\** settings, see **"Elasticsearch Integration for Object Metadata"** (page 171).

## Node Status Configuration

In your storage cluster the [S3 Service](#) runs on each node and so too does the [HyperStore Service](#). S3 requests incoming to your cluster are distributed among the S3 Service instances, and in processing an S3 request an S3 Service instance sends write or read requests to the HyperStore Service instances on multiple nodes (such as when storing a new object in accordance with a 3X replication storage policy, for example).

The settings in the "Node Status Configuration" section of *mts.properties.erb* configure a feature whereby the S3 Service on any node will mark the HyperStore Service on any node as being "Down" if recent requests to that HyperStore Service node have failed at a rate in excess of defined thresholds. When an S3 Service node marks a HyperStore Service node as Down, that S3 Service node will temporarily stop sending requests to that HyperStore Service node. This prevents a proliferation of log error messages that would likely have resulted if requests continued to be sent to that HyperStore Service node, and also allows for the implementation of fall-back consistency levels in the case of storage policies configured with **"Dynamic Consistency Levels"** (page 39).

The configurable thresholds in this section are applied by each individual S3 Service node -- so that each individual S3 Service node makes its own determination of when a problematic HyperStore Service node should be marked as Down.

An S3 Service node will mark a HyperStore Service node as Down in either of these conditions:

- The number of timeout error responses from a HyperStore Service node has exceeded ***hss.-timeout.count.threshold*** (default = 10) over a period of ***hss.timeout.time.threshold*** number of seconds (default = 300) and also the percentage of error responses of any type from that HyperStore Service node has exceeded ***hss.fail.percentage.threshold*** (default = 50) over the past ***hss.-fail.percentage.period*** number of seconds (default = 300).
- The number of other types of error responses from a HyperStore Service node has exceeded ***hss.-fail.count.threshold*** (default = 10) over a period of ***hss.fail.time.threshold*** number of seconds (default = 300) and also the percentage of error responses of any type from that HyperStore Service node has exceeded ***hss.fail.percentage.threshold*** (default = 50) over the past ***hss.fail.percentage.period*** number of seconds (default = 300).

So by default an S3 Service node will mark a HyperStore Service node as Down if during a five minute period the HyperStore Service node has returned either more than 10 timeout responses or more than 10 error responses of other types, while during that same five minute period more than half the requests that the S3 Service node has sent to that HyperStore Service node have failed.

Note that these triggering conditions are based on a combination of number of error responses and percentage of error responses from the problematic HyperStore Service node. This approach avoids marking a HyperStore Service node as down in circumstances when a high percentage of a very small number of requests fail, or when the number of failed requests is sizable but constitutes only a small percentage of the total requests.

Once an S3 Service node has marked a HyperStore Service node as Down, that Down status will persist for ***hss.bring.back.wait*** number of seconds (default = 300) before the Down status is cleared and that S3 Service node resumes sending requests to that HyperStore Service node.

**Note** If the S3 Service node is restarted during this interval, then the Down status for that HyperStore Service node will be lost and the S3 Service node upon restarting will resume sending requests to that HyperStore Service node.

**Note** There is **special handling in the event that a HyperStore Service node returns a "Connection Refused" error** to an S3 Service node (such as would happen if the HyperStore Service was stopped on the target node). In this case the S3 Service node immediately marks that HyperStore Service node as being down, and will then resume sending requests to that HyperStore Service node after a wait period of 15 seconds. This behavior is not configurable.

## Proactive repair queue restriction

*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 writes 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 operation intervention. For more information about proactive repair see "**Proactive Repair**" (page 151).

The proactive repair feature's queueing mechanism entails writing metadata to Cassandra, which is subsequently removed when the endpoint writes are executed by proactive repair. To avoid over-burdening Cassandra with proactive queueing data it's best if a cap be placed on how long the queueing can go on for in a given instance of a write endpoint being unavailable. The *hyperstore.proactiverepair.queue.max.time* property sets this cap, in minutes.

If a node has been unavailable for more than *hyperstore.proactiverepair.queue.max.time* minutes, the system stops writing to the proactive repair queue for that node, 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 after the node comes back online you need to wait for proactive 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 timer is reset to 0 in terms of counting against the *hyperstore.proactiverepair.queue.max.time* limit. So if that subsequently node goes down again, proactive repair queueing would again occur for that node for up to *hyperstore.proactiverepair.queue.max.time* 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.proactiverepair.queue.max.time* to 0.

## Bucket Share API

*cloudian.s3.enablesharedbucket*

To enable the HyperStore S3 API extension that allows an S3 user to list all the buckets that have been shared

with him or her, set this property to *true*.

Default = false

**Note** For information about the relevant S3 API call and how to use the extension, see **"ListBuckets"** (page 958).

## Max User ID Length

*cloudian.userid.length*

Takes its value from *common.csv*: **"cloudian\_userid\_length"** (page 514); use that setting instead.

### 9.5.4. mts-ui.properties.erb

The *mts-ui.properties* file configures the Cloudian Management Console server (CMC). On each of your HyperStore nodes, the file is located at the following path by default:

```
/opt/tomcat/webapps/Cloudian/WEB-INF/classes/mts-ui.properties
```

**Do not directly edit the *mts-ui.properties* file on individual HyperStore nodes.** Instead, if you want to make changes to the settings in this file, edit the configuration template file *mts-ui.properties.erb* on the Puppet master node:

```
/etc/cloudian-<version>-puppet/modules/cmc/templates/mts-ui.properties.erb
```

Certain *mts-ui.properties.erb* properties take their values from settings in [common.csv](#) or from settings that you can control through the CMC's [Configuration Settings](#) page. In the *mts-ui.properties.erb* file these properties' values are formatted as bracket-enclosed variables, like `<%= ... %>`. In the property documentation below, the descriptions of such properties indicate "Takes its value from `<location>`: `<setting>`; use that setting instead." The remaining properties in the *mts-ui.properties.erb* file -- those that are "hard-coded" with specific values -- are settings that in typical circumstances you should have no need to edit. Therefore **in typical circumstances you should not need to manually edit the *mts-ui.properties.erb* file.**

*If you are using the HyperStore Shell*

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), from any directory on the Puppet master node you can edit this configuration file with this command:

```
$ hspkg config -e mts-ui.properties.erb
```

Specify just the configuration file name, not the full path to the file.

In the background this invokes the Linux text editor *vi* to display and modify the configuration file. Therefore you can use the [standard keystrokes supported by vi](#) to make and save changes to the file.

**IMPORTANT !** If you do make edits to *mts-ui.properties.erb*, be sure to push your edits to the cluster and restart the CMC to apply your changes. For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

The *mts-ui.properties.erb* file has the settings below.

*admin.host*

Takes its value from *common.csv*: **"cmc\_admin\_host\_ip"** (page 537); use that setting instead.

*admin.port*

Takes its value from *common.csv:ld\_cloudian\_s3\_admin\_port*, which is controlled by the installer.

*admin.secure*

Takes its value from *common.csv:"admin\_secure"* (page 526); use that setting instead.

*admin.secure.port*

Takes its value from *common.csv: "cmc\_admin\_secure\_port"* (page 527); use that setting instead.

*admin.secure.ssl*

Takes its value from *common.csv: "cmc\_admin\_secure\_ssl"* (page 538); use that setting instead.

*admin.conn.timeout*

The connection timeout for the CMC to use as a client to the Admin Service, in milliseconds. If the CMC cannot connect to the Admin Service within this many milliseconds, the connection attempt times out and the CMC interface displays an error message.

**Note** To provide any of its functions for any type of user, the CMC must successfully connect to the Admin Service.

Default = 10000 (10 seconds)

*iam.enabled*

Takes its value from *common.csv: "iam\_service\_enabled"* (page 528); use that setting instead.

*iam.host*

Takes its value from *common.csv: "iam\_service\_endpoint"* (page 528); use that setting instead.

*iam.port*

Takes its value from *common.csv: "iam\_port"* (page 528); use that setting instead.

*iam.secure.port*

Takes its value from *common.csv: "iam\_secure\_port"* (page 528); use that setting instead.

*iam.secure*

Takes its value from *common.csv: "iam\_secure"* (page 528); use that setting instead.

*iam.socket.timeout*

If during an HTTP/S connection with the IAM Service this many milliseconds pass without any data being passed back by the IAM Service, the CMC will drop the connection.

Default = 30000

*iam.max.retry*

If when trying to initiate an IAM request the CMC fails in an attempt to connect to the IAM Service, the CMC will retry this many times before giving up.

Default = 3

*web.secure*

Takes its value from *common.csv*: "**cmc\_web\_secure**" (page 538); use that setting instead.

*web.secure.port*

Takes its value from *common.csv*: "**cmc\_https\_port**" (page 538); use that setting instead.

*web.nonsecure.port*

Takes its value from *common.csv*: "**cmc\_http\_port**" (page 538); use that setting instead.

*storageuri.ssl.enabled*

Takes its value from *common.csv*: "**cmc\_storageuri\_ssl\_enabled**" (page 539); use that setting instead.

*path.style.access*

Takes its value from *common.csv*: "**path\_style\_access**" (page 516); use that setting instead.

*application.name*

Takes its value from *common.csv*: "**cmc\_application\_name**" (page 539); use that setting instead.

*s3.client.timeout*

Socket timeout on requests from the CMC to the S3 Service, in milliseconds.

Default = 1800000

*s3.upload.part.minsize*

When the CMC uses Multipart Upload to transmit an object to the S3 Service, 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 18MB object, and the configured minimum part size is 5MB, the object will be transmitted in four parts of size 5MB, 5MB, 5MB, and 3MB.

Default = 5242880 (5MB)

*s3.upload.part.threshold*

When a CMC user uploads an object larger than this many bytes, the CMC uses Multipart Upload to transmit the object to the S3 Service, rather than PUT Object.

Default = 16777216 (16MB)

*query.maxrows*

For the CMC's **Usage By Users & Groups** page, the maximum number of data rows to retrieve when processing a usage report request.

Default = 100000

*page.size.default*

For the CMC's **Usage By Users & Groups** page, for usage report pagination, the default number of table rows to display on each page of a tabular report.

Default = 10

*page.size.max*

For the CMC's **Usage By Users & Groups** page, for usage report pagination, the maximum number of table

rows that users can select to display on each page of a tabular report.

Default = 100

*list.multipart.upload.max*

In the CMC's **Objects** page, when a user is uploading multiple objects each of which is large enough to trigger the use of the S3 multipart upload method, the maximum number of multipart upload objects for which to simultaneously display upload progress.

For example, with the default value of 1000, if a user is concurrently uploading 1005 objects that require the use of the multipart upload method, the CMC's **Objects** page will display uploading progress for 1000 of those objects.

Default = 1000

*graph.datapoints.max*

For the CMC's **Usage By Users & Groups** page, the maximum number of datapoints to include within a graphical report.

Default = 1000

*csv.rows.max*

For the CMC's **Usage By Users & Groups** page, the maximum number of rows to include within a comma-separated value report.

Default = 1000

*fileupload.abort.max.hours*

When a very large file is being uploaded through the CMC, the maximum number of hours for the CMC to wait for the S3 file upload operation to complete. If this maximum is reached, the upload operation is aborted.

Default = 3

*license.request.email*

Email address to which to send Cloudian license requests. This address is used in a request license information link in the CMC interface.

Default = cloudian-license@cloudian.com

*admin.auth.user*

Takes its value from *common.csv*: "**admin\_auth\_user**" (page 525); use that setting instead.

*admin.auth.pass*

Takes its value from *common.csv*: "**admin\_auth\_pass**" (page 525); use that setting instead.

*admin.auth.realm*

Takes its value from *common.csv*: "**admin\_auth\_realm**" (page 526); use that setting instead.

*user.password.min.length*

Takes its value from *common.csv*: "**user\_password\_min\_length**" (page 527); use that setting instead.

*user.password.dup.char.ratio.limit*

Takes its value from *common.csv*: "**user\_password\_dup\_char\_ratio\_limit**" (page 527); use that setting

instead.

*user.password.unique.generations*

Takes its value from *common.csv*: "**user\_password\_unique\_generations**" (page 527); use that setting instead.

*user.password.rotation.graceperiod*

Takes its value from *common.csv*: "**user\_password\_rotation\_graceperiod**" (page 527); use that setting instead.

*user.password.rotation.expiration*

Takes its value from *common.csv*: "**user\_password\_rotation\_expiration**" (page 527); use that setting instead.

*acl.grantee.public*

In the CMC UI dialogs that let CMC users specify permissions on S3 buckets, folders, or files, the label to use for the ACL grantee "public". If the *acl.grantee.public* property is not set in *mts-ui.properties*, then the system instead uses the *acl.grantee.public* value from your *resources\_xx\_XX.properties* files (for example, for the U.S. English version of the UI, the value is in *resources\_en\_US.properties*).

Default = commented out (value is taken from resource file[s])

*acl.grantee.cloudianUser*

In the CMC UI dialogs that let CMC users specify permissions on S3 buckets, folders, or files, the label to use for the ACL grantee "all Cloudian HyperStore service users". If the *acl.grantee.cloudianUser* property is not set in *mts-ui.properties*, then the system instead uses the *acl.grantee.cloudianUser* value from your *resources\_xx\_XX.properties* files (for example, for the U.S. English version of the UI, the value is in *resources\_en\_US.properties*).

Default = commented out (value is taken from resource file[s])

*session.timeout.url*

URL of page to display if a CMC user's login session times out.

If this value is not set in *mts-ui.properties*, the behavior defaults to displaying the CMC Login screen if the user's session times out.

Default = commented out (CMC Login screen displays)

*admin.manage\_users.enabled*

This setting controls whether the **Manage Users** function will be enabled in the CMC GUI. For a screen shot and description of this function, see "**Manage Users**" (page 262).

Options are:

- **true** — This function will display for users logged in as a system administrator or group administrator. For group admins this function is restricted to their own group.
- **false** — This function will not display for any users. If you set this to "false" then the **Manage Users** functionality as a whole is disabled and the more granular *admin.manage\_users.\*.enabled* properties below are ignored.
- **SystemAdmin** — This function will display only for users logged in as a system administrator.
- **GroupAdmin** — This function will display only for users logged in as a group administrator.

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

**Note** If you want to enable some aspects of the **Manage Users** function and not others, you can have *admin.manage\_users.enabled* set so that the function is enabled for your desired user types, and then use the granular *admin.manage\_users.\*.enabled* properties below to enable/disable specific capabilities.

#### *admin.manage\_users.create.enabled*

Within the **Manage Users** function in the CMC GUI, this setting enables or disables the capability to create new users.

Options are:

- true — This capability will display for users logged in as a system administrator or group administrator. For group admins this capability is restricted to their own group.
- false — This capability will not display for any users.
- SystemAdmin — This capability will display only for users logged in as a system administrator.
- GroupAdmin — This capability will display only for users logged in as a group administrator.

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

#### *admin.manage\_users.edit.enabled*

Within the **Manage Users** function in the CMC GUI, this setting enables or disables the capability to edit existing users' profiles and service attributes.

Options are:

- true — This capability will display for users logged in as a system administrator or group administrator. For group admins this capability is restricted to their own group.
- false — This capability will not display for any users.
- SystemAdmin — This capability will display only for users logged in as a system administrator.
- GroupAdmin — This capability will display only for users logged in as a group administrator.

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

**Note** Regardless of how you configure the *admin.manage\_users.edit.enabled* 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.

#### *admin.manage\_users.delete.enabled*

Within the **Manage Users** function in the CMC GUI, this setting enables or disables the capability to delete users.

Options are:

- `true` — This capability will display for users logged in as a system administrator or group administrator. For group admins this capability is restricted to their own group.
- `false` — This capability will not display for any users.
- `SystemAdmin` — This capability will display only for users logged in as a system administrator.
- `GroupAdmin` — This capability will display only for users logged in as a group administrator.

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

*admin.manage\_users.viewuserdata.enabled*

Takes its value from *common.csv*: "**cmc\_view\_user\_data**" (page 541); use that setting instead.

*admin.manage\_users.edit.user\_credentials.enabled*

Within the **Manage Users** function in the CMC GUI, this setting enables or disables the capability to change users' CMC login passwords and to view and manage user's S3 access credentials.

Options are:

- `true` — This capability will display for users logged in as a system administrator or group administrator. For group admins this capability is restricted to their own group.
- `false` — This capability will not display for any users.
- `SystemAdmin` — This capability will display only for users logged in as a system administrator.
- `GroupAdmin` — This capability will display only for users logged in as a group administrator.

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

**Note** Regardless of how you configure the *admin.manage\_users.edit.user\_credentials.enabled* setting, regular users can manage their own CMC login password and S3 access credentials, in the **Security Credentials** page of the CMC.

*admin.manage\_users.edit.user\_qos.enabled*

Within the **Manage Users** function in the CMC GUI, this setting enables or disables the capability to set Quality of Service (QoS) controls for specific users.

Options are:

- `true` — This capability will display for users logged in as a system administrator or group administrator. For group admins this capability is restricted to their own group.
- `false` — This capability will not display for any users.
- `SystemAdmin` — This capability will display only for users logged in as a system administrator.
- `GroupAdmin` — This capability will display only for users logged in as a group administrator.

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

**Note** This setting is relevant only if the *admin.manage\_users.enabled* and *admin.manage\_users.edit.enabled* settings are enabled.

*admin.manage\_groups.enabled*

This setting controls whether the **Manage Groups** function will be enabled in the CMC GUI. For a screen shot and description of this function, see "**Manage Groups**" (page 270).

Options are:

- true — This function will display for users logged in as a system administrator or group administrator. For group admins this function is restricted to their own group.
- false — This function will not display for any users. If you set this to "false" then the **Manage Groups** functionality as a whole is disabled and the more granular *admin.manage\_groups.\*.enabled* properties below are ignored.
- SystemAdmin — This function will display only for users logged in as a system administrator.
- GroupAdmin — This function will display only for users logged in as a group administrator.

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

**Note** If you want to enable some aspects of the **Manage Groups** function and not others, you can have *admin.manage\_groups.enabled* set so that the function is enabled for your desired user types, and then use the granular *admin.manage\_groups.\*.enabled* properties below to enable/disable specific capabilities.

*admin.manage\_groups.create.enabled*

Within the **Manage Groups** function in the CMC GUI, this setting enables or disables the capability to create new groups.

Options are:

- true — This capability will display for users logged in as a system administrator.
- false — This capability will not display for any users.

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

*admin.manage\_groups.edit.enabled*

Within the **Manage Groups** function in the CMC GUI, this setting enables or disables the capability to edit an existing group's profile and service attributes.

Options are:

- true — This capability will display for users logged in as a system administrator or group administrator. For group admins this capability is restricted to their own group.

**Note** Even when this 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 changing the default user QoS settings for the group. The latter capability is controlled by a more granular configuration property *admin.-manage\_groups.user\_qos\_groups\_default.enabled* (below), which defaults to "true".

- false — This capability will not display for any users.
- SystemAdmin — This capability will display only for users logged in as a system administrator.
- GroupAdmin — This capability will display only for users logged in as a group administrator.

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

#### *admin.manage\_groups.delete.enabled*

Within the **Manage Groups** function in the CMC GUI, this setting enables or disables the capability to delete a group.

Options are:

- true — This capability will display for users logged in as a system administrator.
- false — This capability will not display for any users.

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

#### *admin.manage\_groups.user\_qos\_groups\_default.enabled*

Within the **Manage Groups** function in the CMC GUI, this setting enables or disables the capability to set default Quality of Service (QoS) controls for users within a specific group.

Options are:

- true — This capability will display for users logged in as a system administrator or group administrator. For group admins this capability is restricted to their own group.
- false — This capability will not display for any users.
- SystemAdmin — This capability will display only for users logged in as a system administrator.
- GroupAdmin — This capability will display only for users logged in as a group administrator.

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

**Note** This setting is relevant only if the *admin.manage\_groups.enabled* and *admin.manage\_groups.edit.enabled* settings are enabled.

#### *account.profile.writeable.enabled*

This setting controls whether a user can edit his or her own account profile information in the **Account** section of the CMC GUI. For user types for which this editing capability is not enabled, account profile information will be read-only. For a screen shot and description of this function, see **"Profile"** (page 399).

Options are:

- true — This capability will be enabled for all user types (system administrator, group administrator, and regular user).
- false — This capability will be disabled for all user types.
- SystemAdmin — This capability will be enabled only for users logged in as a system administrator.
- GroupAdmin — This capability will be enabled only for users logged in as a group administrator.
- User — This capability will be enabled only for users logged in as a regular user.

- You can also specify a comma-separated list of multiple user types — for example, "SystemAdmin,GroupAdmin".

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

#### *account.credentials.enabled*

This setting controls whether the **Security Credentials** function will be enabled in the CMC GUI. For a screen shot and description of this function, see Security Credentials in the CMC Help document.

Options are:

- true — This function will display for all user types (system administrator, group administrator, and regular user).
- false — This function will not display for any user types. If you set this to "false" then the Security Credentials functionality as a whole is disabled and the more granular *account.credentials.\*.enabled* properties below are ignored.
- SystemAdmin — This function will display only for users logged in as a system administrator.
- GroupAdmin — This function will display only for users logged in as a group administrator.
- User — This function will display only for users logged in as a regular user.
- You can also specify a comma-separated list of multiple user types for which to enable this function — for example, "SystemAdmin,GroupAdmin".

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

**Note** If you want to enable some aspects of the **Security Credentials** function and not others, you can have *account.credentials.enabled* set so that the function is enabled for your desired user types, and then use the granular *account.credentials.\*.enabled* properties below to enable/disable specific capabilities.

#### *account.credentials.access.enabled*

Within the **Security Credentials** function in the CMC GUI, this setting enables or disables the capability of CMC users to view and change their own S3 storage access keys.

Options are:

- true — This capability will display for all user types (system administrator, group administrator, and regular user).
- false — This capability will not display for any user types.
- SystemAdmin — This capability will display only for users logged in as a system administrator.
- GroupAdmin — This capability will display only for users logged in as a group administrator.
- User — This capability will display only for users logged in as a regular user.
- You can also specify a comma-separated list of multiple user types for which to enable this capability — for example, "SystemAdmin,GroupAdmin".

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

#### *account.credentials.signin.enabled*

Within the **Security Credentials** function in the CMC GUI, this setting enables or disables the capability CMC users to change their own CMC login password.

Options are:

- true — This capability will display for all user types (system administrator, group administrator, and regular user).
- false — This capability will not display for any user types.
- SystemAdmin — This capability will display only for users logged in as a system administrator.
- GroupAdmin — This capability will display only for users logged in as a group administrator.
- User — This capability will display only for users logged in as a regular user.
- You can also specify a comma-separated list of multiple user types for which to enable this capability — for example, "SystemAdmin,GroupAdmin".

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

#### *account.activity.enabled*

This setting controls whether the **Account Activity** function will be enabled in the CMC GUI. For a screen shot and description of this function, see "**Account Activity**" (page 282).

Options are:

- true — This capability will display for users logged in as a system administrator.
- false — This capability will not display for any users.

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

#### *usage.enabled*

This setting controls whether the **Usage By Users and Groups** function will be enabled in the CMC GUI. For a screen shot and description of this function, see "**Usage By Users & Groups**" (page 211).

Options are:

- true — This function will display for all user types (system administrator, group administrator, and regular user).
- false — This function will not display for any user types.
- SystemAdmin — This function will display only for users logged in as a system administrator.
- GroupAdmin — This function will display only for users logged in as a group administrator.
- User — This function will display only for users logged in as a regular user.
- You can also specify a comma-separated list of multiple user types for which to enable this function — for example, "SystemAdmin,GroupAdmin".

Default = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

#### *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. For a screen shot and description of this function, see "**Security Credentials**" (page 400).

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 = Commented out and uses internal default of "true". To assign a different value, uncomment the setting and edit its value.

**Note** For HyperStore system admins, the S3 service endpoint information displays in the [Cluster Information](#) page. The `security.serviceinfo.enabled` property has no effect on this display.

*login.languageselection.enabled*

Takes its value from *common.csv*: "**cmc\_login\_languageselection\_enabled**" (page 539); use that setting instead.

*login.grouplist.enabled*

Takes its value from *common.csv*: "**cmc\_login\_grouplist\_enabled**" (page 540); use that setting instead.

*login.banner.\**

These settings take their values from the *cmc\_login\_banner\_\** settings in *common.csv*; use those settings instead. For information about using those settings see "**Configuring a Login Page Acknowledgment Gate**" (page 408).

*grouplist.enabled*

Takes its value from *common.csv*: "**cmc\_grouplist\_enabled**" (page 539); use that setting instead.

*grouplist.size.max*

Takes its value from *common.csv*: "**cmc\_grouplist\_size\_max**" (page 540); use that setting instead.

*error.stacktrace.enabled*

Throughout the CMC UI, when an exception occurs and an error page is displayed to the user, include on the error page a link to a stack trace.

Options are:

- true — Stack trace links will display for all user types (system administrator, group administrator, and regular user).
- false — Stack trace links will not display for any user types.
- SystemAdmin — Stack trace links will display only for users logged in as a system administrator.
- GroupAdmin — Stack trace links will display only for users logged in as a group administrator.
- User — Stack trace links will display only for users logged in as a regular user.
- You can also specify a comma-separated list of multiple user types for which to enable this feature — for example, "SystemAdmin,GroupAdmin".

Default = false

*admin.whitelist.enabled*

Takes its value from *common.csv*: "**admin\_whitelist\_enabled**" (page 537); use that setting instead.

*sso.enabled*

Whether to enable CMC support for single sign-on 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*.

Default = false

**Note** For more information on CMC SSO, see "**Implementing Single Sign-On for the CMC**" (page 410).

*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.

Default = ss0sh5r3dk3y

**IMPORTANT !** If you enable CMC SSO functionality (using the *sso.enabled* setting), then for security reasons you should **set a custom value for *sso.shared.key***. Do not leave it at its default value.

*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. This effectively serves as a request expiry mechanism.

Default = 3600000

*sso.cookie.cipher.key*

Triple DES key used for cookie encryption.

Default = 123456789012345678901234

**IMPORTANT !** If you enable CMC SSO functionality (using the *sso.enabled* setting), then for security reasons you should **set a custom value for *sso.cookie.cipher.key***. Do not leave it at its default value.

**Note** If you change this value after CMC SSO has already been in service, end users who had used SSO previously will have on their browser a Cloudian SSO cookie that is no longer valid. If such users access the CMC after your *sso.cookie.cipher.key* change, the CMC detects the invalid cookie, deletes it, and drops a new, valid one.

*bucket.storagepolicy.showdetail.enabled*

Within the **Bucket Properties** function in the CMC GUI, this setting enables or disables the display of a "Storage Policy" tab that provides information about the storage policy being used by the bucket. This information includes the storage policies replication factor or erasure coding k+m configuration..

Options are:

- `true` — This tab will display for all user types (system administrator, group administrator, and regular user).
- `false` — This tab will not display for any user types.
- `SystemAdmin` — This tab will display only for users logged in as a system administrator.
- `GroupAdmin` — This tab will display only for users logged in as a group administrator.
- `User` — This tab will display only for users logged in as a regular user.
- You can also specify a comma-separated list of multiple user types for which to display this tab — for example, `"SystemAdmin,GroupAdmin"`.

Default = Commented out and uses internal default of `"true"`. To assign a different value, uncomment the setting and edit its value.

*bucket.tiering.enabled*

Takes its value from **"Enable Auto-Tiering"** (page 347) in the CMC's [Configuration Settings](#) page; use that setting instead.

*tiering.perbucketcredentials.enabled*

Takes its value from **"Enable Per Bucket Credentials"** (page 347) in the CMC's [Configuration Settings](#) page; use that setting instead.

*tiering.customendpoint.enabled*

Takes its value from **"Enable Custom Endpoint"** (page 348) in the CMC's [Configuration Settings](#) page; use that setting instead.

*bucket.tiering.default.destination.list*

Takes its value from *common.csv*: **"cmc\_bucket\_tiering\_default\_destination\_list"** (page 541); use that setting instead.

*bucket.tiering.custom.url*

Takes its value from [Default Tiering URL](#) in the CMC's [Configuration Settings](#) page; use that setting instead.

*puppet.master.licensefile*

This setting supports the feature whereby an updated HyperStore license file can be uploaded via the CMC. Do not edit.

*puppet.master.fileupdate.location*

This setting supports the feature whereby an updated HyperStore license file can be uploaded via the CMC. Do not edit.

*local.ssh.privateKey*

Private key to use when the CMC connects via SSH to the Puppet master node or to other HyperStore nodes when implementing node management functions. The Cloudian installation script automatically populates this setting.

*local.ssh.passphrase*

Pass phrase to use when the CMC connects via SSH to the Puppet master node or to other HyperStore nodes when implementing node management functions. The Cloudian installation script automatically populates this setting.

*local.ssh.applianceKey*

Private key to use when the CMC connects via SSH to a new HyperStore Appliance node when the appliance node is being added to an existing system. The Cloudian installation script automatically populates this setting.

*local.temp.dir*

This setting supports the feature whereby an updated HyperStore license file can be uploaded via the CMC. Do not edit.

*remote.ssh.user*

The user as which to connect via SSH to the Puppet master node or HyperStore nodes.

Default = root

*remote.ssh.port*

The port to which to connect via SSH to the Puppet master node or HyperStore nodes.

Default = 22

*offload.services.node.options*

Used internally by the CMC when invoking the installer script for certain operations. Do not edit.

Default = -m

*uninstall.node.options*

Used internally by the CMC when invoking the installer script for certain operations. Do not edit.

Default = -u -r

*elasticsearch.enabled*

For information about this setting, see **"Elasticsearch Integration for Object Metadata"** (page 171).

*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 Cassandra, each time the Dashboard page is loaded in your browser.

The *proactive.repair.queue.warning.enabled* property enables and disables this Dashboard feature. If this property is set to "false", then the Dashboard does not retrieve proactive repair queue length data from Cassandra and does not display any warnings in regard to proactive repair queue length.

Default = true

*cloudian.userid.length*

Takes its value from *common.csv*: **"cloudian\_userid\_length"** (page 514); use that setting instead.

### 9.5.5. Other Configuration Files

The tables below briefly describe additional HyperStore configuration files that reside in the configuration directories on your Puppet master node. Under typical circumstances you should not need to manually edit these files.

All of these files are in sub-directories under the `/etc/cloudian-<version>-puppet` directory. For brevity, in the section headings that follow `/etc/cloudian-<version>-puppet` is replaced with `"..."` and only the sub-directory is specified.

- **"Files under `.../manifests/extdata/`"** (page 595)
- **"Files under `.../modules/cloudians3/templates/`"** (page 596)
- **"Files under `.../modules/cmc/templates/`"** (page 596)
- **"Files under `.../modules/salt/`"** (page 597)

If you are using the **HyperStore Shell**

If you are using the **HyperStore Shell (HSH)** as a **Trusted user**, from any directory on the Puppet master node you can edit any of these configuration files with this command:

```
$ hspkg config -e <filename>
```

Specify just the configuration file name, not the full path to the file.

In the background this invokes the Linux text editor `vi` to display and modify the configuration file. Therefore you can use the [standard keystrokes supported by vi](#) to make and save changes to the file.

#### 9.5.5.1. Files under `.../manifests/extdata/`

File	Purpose
<code>adminsslconfigs.csv</code>	Configures TLS/SSL implementation for the Admin Service's HTTPS listener.
<code>dynsettings.csv</code>	Used by the CMC's <a href="#">Configuration Settings</a> page. Do not edit this file.
<code>fileupdates.csv</code>	There may be rare circumstances in which Cloudian Support asks you to use this file, in which case you will be provided instructions for using it.
<code>iamsslconfigs.csv</code>	Configures TLS/SSL implementation for the IAM Service. This file is updated automatically if you use the installer's "Advanced Configuration Options" to manage HTTPS for the IAM Service. For more information see <b>"HTTPS Support (TLS/SSL)"</b> (page 114)
<code>&lt;node&gt;.csv</code>	These files are for settings that are tailored to individual nodes, as identified by the <code>&lt;node&gt;</code> segment of the file names (for example, <code>host1.csv</code> , <code>host2.csv</code> , and so on). There will be one such file for each node in your system. These files are created and pre-configured by the HyperStore install script, based on information that you provided during installation. Settings in a <code>&lt;node&gt;.csv</code> file override default settings from <code>common.csv</code> , for the specified node.
<code>&lt;regionName&gt;_region.csv</code>	Configures settings that are particular to a service region. Settings that would have different values in different regions are in this file. If you have multiple HyperStore service regions, you will have multiple instances of this file. These files are created and pre-configured by the HyperStore install script, based on information that you provided during installation.

File	Purpose
<i>s3sslconfigs.csv</i>	Configures TLS/SSL implementation for the S3 Service. This file is updated automatically if you use the installer's "Advanced Configuration Options" to manage HTTPS for the S3 Service. For more information see <b>"HTTPS Support (TLS/SSL)"</b> (page 114)
<i>&lt;regionName&gt;_topology.csv</i>	Specifies a topology of nodes, racks, and data centers in a service region. If you have multiple HyperStore service regions, you will have multiple instances of this file. These files are created and pre-configured by the HyperStore install script, based on information that you provided during installation.

#### 9.5.5.2. Files under .../modules/cloudians3/templates/

File	Purpose
<i>admin.xml.erb</i>	This file configures the Admin Service's underlying Jetty server functionality, for processing incoming HTTP requests.
<i>admin_realm.properties.erb</i>	This file configures HTTP Basic Authentication for the Admin Service.
<i>cloudian-cron.tab.erb</i>	This file configures HyperStore system maintenance cron jobs.
<i>log4j-*.xml.erb</i>	These files configure logging behavior for various services such as the S3 Service, Admin Service, HyperStore Service, and so on. For information about settings in these files, see <b>"Log Configuration Settings"</b> (page 626).
<i>s3.xml.erb</i>	This file configures the S3 Service's underlying Jetty server functionality, for processing incoming HTTP requests.
<i>storage.xml.erb</i>	This file configures the HyperStore Service's underlying Jetty server functionality, for processing incoming HTTP requests.
<i>tiering-map.txt.erb</i>	This file is not used currently.
<i>tiering-regions.xml.erb</i>	In support of the HyperStore auto-tiering feature, this file lists S3 endpoints that objects can be auto-tiered to. By default this file is pre-configured with all the Amazon S3 regional service endpoints. If you have a multi-region system, the file is also pre-configured with the S3 endpoints for each of your service regions, and the file is used in support of the <a href="#">cross-region replication</a> feature.

#### 9.5.5.3. Files under .../modules/cmc/templates/

File	Purpose
<i>server.xml.erb</i>	This file configures the HyperStore Service's underlying Tomcat server functionality, for processing incoming HTTP requests.

#### 9.5.5.4. Files under ../modules/salt

Starting with HyperStore version 7.2, HyperStore uses the open source version of [Salt](#) for certain configuration management functions (such as configuration management of the HyperStore firewall -- which from a user perspective is controlled through the installer's Advanced Configuration Options menu). **Do not edit any of the configuration files under `/etc/cloudian-<version>-puppet/modules/salt`.**

**Note** Salt configuration management activity is recorded in the log file `/var/log/cloudian/salt.log`.

#### 9.5.6. Using JMX to Dynamically Change Configuration Settings

Certain HyperStore configuration settings (a minority of them) can be dynamically reloaded via JMX. You can use a JMX client such as [JConsole](#) or [Jmxterm](#) to connect to the JMX listening port of the particular service that you're configuring (S3 Service JMX port = 19080; Admin Service JMX port = 19081; HyperStore Service JMX port = 19082; Redis Monitor JMX port = 19083). *JConsole* is a graphical user interface that's part of the HyperStore system's Java installation and can be found under `JAVA_HOME/bin`. *Jmxterm* is a command line tool that comes bundled with the HyperStore system and can be found under `opt/cloudian/tools`.

When you use JMX to make a configuration setting change, the change is applied to the service dynamically — you do not need to restart the service for the change to take effect. However, the **setting change persists only for the current running session of the affected service**. If you restart the service it will use whatever setting is in the configuration file. Consequently, if you want to change a setting dynamically and also have your change persist through a service restart, you should change the setting in the configuration file as well as changing it via JMX.

In the documentation of HyperStore configuration files, if a setting supports being dynamically changed via JMX, the setting description indicates "Reloadable via JMX". It also indicates the name of the dynamically changeable MBean attribute that corresponds to the configuration file setting. For example, the documentation of the HyperStore Service configuration property `repair.session.threadpool.corepoolsize` includes a note that says:

"Reloadable via JMX (HyperStore Service's JMX port 19082; MBean attribute = com.gemini.cloudian.hybrid.server → FileRepairService → Attributes → RepairSessionThreadPoolCorePoolSize)"

**Note** Some settings in HyperStore configuration files can be set through the CMC's [Configuration Settings](#) page. The CMC uses JMX to apply setting changes dynamically, and the CMC also automatically makes the corresponding configuration file change and triggers a Puppet push so that your changes will persist across service restarts. For these settings it's therefore best to use the CMC to make any desired edits rather than directly using JMX. For such configuration file settings, the descriptions in the HyperStore configuration file documentation indicate that you should use the CMC to edit the setting. Consequently these settings are not flagged in the documentation as JMX reloadable.

## 9.6. Configuration Special Topics

### 9.6.1. 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\_data\_directory* in *common.csv*)
- */var/lib/{cassandra,cassandra\_commit,redis}*
- */var/log/{cloudian,cassandra,redis}*
- */opt/{cassandra,cloudian,cloudian-packages,cloudianagent,dnsmasq,redis,tomcat}*
- */etc/cloudian-<version>-puppet\**

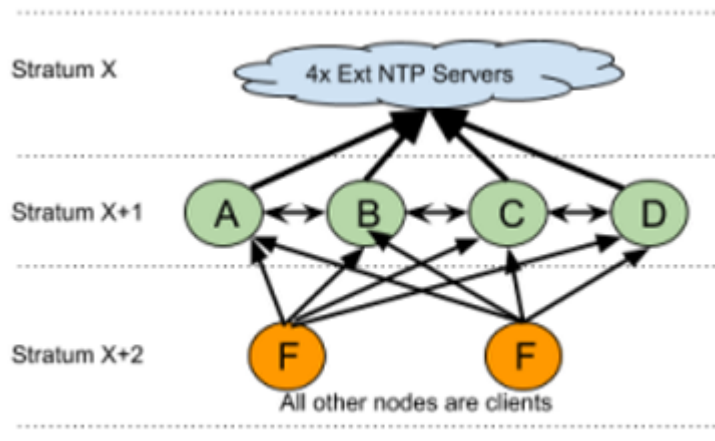
### 9.6.2. NTP Automatic Set-Up

**Note** This topic describes the NTP configuration that HyperStore automatically implements. If instead you are using your own custom NTP set-up, Cloudian 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 *ntpd*, as follows:

- In each of your HyperStore data centers, four HyperStore nodes are configured as internal NTP servers. These internal NTP server 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 HyperStore data center has only four or fewer nodes, then all the nodes in the data center are configured as internal NTP servers.)
- All other nodes in the data center 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 that page you can also view the list of external NTP servers to which the internal NTP servers will synchronize.

**IMPORTANT !** In order to connect to the external NTP servers the internal NTP servers must be allowed outbound internet access.

**Note** *ntpd* is configured to automatically start on host boot-up. However, it's recommended that after booting a HyperStore host, you verify that *ntpd* is running (which you can do with the *ntpq -p* command — if *ntpd* is running this command will return a list of connected time servers).

#### 9.6.2.1. 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 471).

#### 9.6.3. 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 Puppet master node, change to 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 Puppet

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.

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. The final prompt will ask whether you want to save your changes -- type **yes** to do so.
4. Go back to the installer's main menu again and enter **2** for "Cluster Management". Then enter **a** for "Review Cluster Configuration", and when prompted as to whether you want to update the Puppet master with your changes type **yes**.
5. After returning to the Cluster Management menu, enter **b** for "Push Configuration Settings to Cluster", and follow the prompts.
6. After returning to the Cluster Management menu again, enter **c** for "Manage Services", and 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

Do not exit the installer until you complete Step 7 below, if applicable.

7. **If you have the HyperStore firewall enabled** (as described in "**HyperStore Firewall**" (page 100)), 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 taking these steps with the installer:
  - a. At the installer's main menu, enter **4** for "Advanced Configuration Options". Then at the Advanced Configuration Options menu enter **s** for "Configure Firewall".
  - b. At the Firewall Configuration menu, enter **x** for "Apply configuration changes and return to previous menu". When prompted, enter **yes** to confirm that you want to apply your configuration changes.

After your changes are successfully applied to the cluster you can exit the installer.

### 9.6.4. Changing S3, Admin, CMC, or IAM 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**.)

**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 `sqs_endpoint` in [common.csv](#) and then restarting the SQS service.

1. On the Puppet master node, change to 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 Puppet 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.

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).

**Note** The only instance of the string "s3" should be the leading prefix (as in the examples above). Do not also include "s3" in the `<regionname>` value, the `<dcname>` value, or the `<domain>` value because having two instances of "s3" in the service endpoint will cause S3 service requests to fail. For example, do not have a service endpoint such as `"s3-tokyo.s3.enterprise.com"`.

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, and same for the CMC service and the IAM service.

The final prompt will ask whether you want to save your changes -- type *yes* to do so.

4. Go to the main menu again and choose "Cluster Management" → "Push Configuration Settings to Cluster" and follow the prompts.
5. Go to the "Cluster Management" menu again, choose "Manage Services", and restart the S3 Service and the CMC. If you are using DNSMASQ for HyperStore service endpoint resolution, then also restart DNSMASQ.

**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" in the *HyperStore Installation Guide*.

**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 log into the CMC 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.

### 9.6.5. 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 Puppet master node, change to the [installation staging directory](#) 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 Puppet 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.

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 optim-

ization for all nodes in your cluster. When the script run is done the installer interface will prompt you to continue to the next steps.

4. Go to the installer's main menu again and choose "Cluster Management" → "Push Configuration Settings to Cluster" and follow the prompts.
5. Go to the "Cluster Management" menu again, choose "Manage Services", and restart the S3 Service, the HyperStore Service, and the Cassandra Service.

### 9.6.6. Vanity Domains for S3 Buckets

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 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 service domain is *s3.enterprise.com* then you would map vanity domain *baseball-stats.com* to CNAME *baseball-stats.com.s3.enterprise.com*.
- In your DNS set-up, use CNAME to map a wildcard of sub-domains of your S3 service domain to the S3 service domain itself. For example, if your HyperStore S3 service domain is *s3.enterprise.com* then you would map wildcard *\*.s3.enterprise.com* to CNAME *s3.enterprise.com*.

This page left intentionally blank

# Chapter 10. Logging

## 10.1. HyperStore Logs

Subjects covered in this section:

- *Introduction (below)*
- **"Admin Service Logs"** (page 605)
- **"Cassandra Logs"** (page 606)
- **"CMC Log"** (page 608)
- **"HyperStore Firewall Log"** (page 609)
- **"HyperStore Service Logs"** (page 609)
- **"HyperStore Shell Log"** (page 613)
- **"IAM Service Logs"** (page 613)
- **"Monitoring Agent and Collector Logs"** (page 615)
- **"Phone Home (Smart Support) Log"** (page 616)
- **"Redis and Redis Monitor Logs"** (page 617)
- **"S3 Service Logs (including Auto-Tiering, CRR, and WORM)"** (page 619)
- **"SQS Service Logs"** (page 625)

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 below indicate each log's default location, logging level, rotation and retention policy, log entry format, and where to modify the log's configuration.

**Note** With the exception of Cassandra and Redis logs, all HyperStore logs are located in `/var/log/cloudian/`.

**Note** For information on viewing logs from within the HyperStore Shell, see **"Using the HSH to View Logs"** (page 640).

### 10.1.1. 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	<code>yyyy-mm-dd HH:mm:ss,SSS PriorityLevel[ThreadId]MessageCode ClassName:MESSAGE</code> 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 HyperStore online Help.
Log Entry Example	<code>2017-05-04 15:48:03,158 ERROR[main]HS081003 CassandraProtectionPolicy:Caught: me.prettyprint.hector.api.exceptions.</code>

	<code>HectorException: [10.50.10.21(10.50.10.21):9160] All host pools marked down. Retry burden pushed out to client.</code>
Default Logging Level	INFO
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB <b>or</b> if oldest rotated file age reaches 180 days.</p>
Configuration	In the Puppet template file <code>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-admin.xml.erb</code> , this log is configurable in the block that starts with <code>&lt;RollingRandomAccessFile name="ADMINAPP"&gt;</code> . For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).

#### Admin Service request log (*cloudian-admin-request-info.log*)

Location	On every node, <code>/var/log/cloudian/cloudian-admin-request-info.log</code>
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS ClientIpAddress HttpMethod Uri QueryParams  DurationMicrosecs HttpStatus</pre> <div style="background-color: #d4edda; padding: 10px; margin-top: 10px;"> <p><b>Note</b> Query parameters are not logged for requests that involve user credentials.</p> </div>
Log Entry Example	<code>2016-10-27 14:54:01,170 10.20.2.57 GET /group/list limit:100 188212 200</code>
Logging Level	Not applicable
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 100MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 2GB <b>or</b> if oldest rotated file age reaches 180 days.</p>
Configuration	In the Puppet template file <code>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-admin.xml.erb</code> , this log is configurable in the block that starts with <code>&lt;RollingRandomAccessFile name="ADMINREQ"&gt;</code> . For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).

### 10.1.2. Cassandra Logs

#### Cassandra application log (*system.log*)

Location	On every node, <i>var/log/cassandra/system.log</i>
Log Entry Format	PriorityLevel [ThreadId] Date(ISO8601) CallerFile:Line# - MESSAGE
Log Entry Example	INFO [FlushWriter:3] 2016-11-10 21:09:59,487 Memtable.java:237 - Writing Memtable-Migrations @445036697(12771/15963 serialized/live bytes, 1 ops)
Default Logging Level	INFO
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 20MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 200MB <b>or</b> if oldest rotated file age reaches 30 days.</p>
Configuration	<p>Configuration: <i>/etc/cloudian-&lt;version&gt;-puppet/modules/cassandra/templates/logback.xml.erb</i>. For setting descriptions see the online documentation for Logback:</p> <ul style="list-style-type: none"> <li>• <a href="#">FixedWindowRollingPolicy</a></li> <li>• <a href="#">SizeBasedTriggeringPolicy</a></li> </ul>

#### 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 <b>client side</b> as the S3 Service sends requests to Cassandra.
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS LogEntryType S3RequestId MESSAGE</pre> <p>The LogEntryType is one of NORMAL, ERROR, or SLOW.</p>
Log Entry Example	<pre>2016-11-22 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>
Default Logging Level	<p>DEBUG</p> <div style="background-color: #d4edda; padding: 10px; border: 1px solid #c3e6cb;"> <p><b>Note</b> In <i>log4j-s3.xml.erb</i> on your Puppet master node there are three different <i>AsyncLogger</i> instances for Cassandra request logging. The ERROR logger logs entries when a Cassandra request results in an error; the SLOW logger logs entries when a Cassandra request takes more than 5 seconds to process; and the NORMAL logger logs all Cassandra requests. The three loggers all write to <i>/var/log/cloudian/cassandra-s3-tx.log</i>, and the implementation prevents duplicate entries across the three loggers. All three loggers are set to DEBUG level by default; and each logger works only if set to DEBUG or TRACE. To disable a logger, set its level to INFO or higher. For example to disable the NORMAL logger so that only error and slow requests are recorded, set the NORMAL logger's level to INFO. Then do a Puppet push and restart the S3 Service.</p> </div>

Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB <b>or</b> if oldest rotated file age reaches 180 days.</p>
Configuration	<p>In the Puppet template file <i>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-s3.xml.erb</i>, this log is configurable in the block that starts with <i>&lt;RollingRandomAccessFile name="CASSANDRATX"</i>. For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).</p>

**Note** Currently only a small fraction 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). Currently the only types of requests that use the DataStax driver are requests to clean up tombstones. As new Cassandra request types are implemented in HyperStore they will use the DataStax driver, and thus over time a growing portion of Cassandra requests will support request logging.

### 10.1.3. CMC Log

*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> <p>In the case of log entries for user logins to the CMC, the MESSAGE value will be formatted as follows:</p> <p>Normal login</p> <pre>Login &lt;groupId&gt; &lt;userId&gt; from: &lt;ipAddress&gt; Success</pre> <pre>Login &lt;groupId&gt; &lt;userId&gt; from: &lt;ipAddress&gt; Failed [&lt;reason&gt;]</pre> <p>SSO login</p> <pre>SSOLogin &lt;groupId&gt; &lt;userId&gt; from: &lt;ipAddress&gt; Success</pre> <pre>SSOLogin &lt;groupId&gt; &lt;userId&gt; from: &lt;ipAddress&gt; Failed [&lt;reason&gt;]</pre>
Log Entry Example	<pre>2017-05-04 11:56:48,475 INFO [localhost-startStop-1] ServiceMapUtil&gt;Loading service map info from: cloudianservicemap.json</pre>
Default Logging Level	INFO
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB <b>or</b> if oldest rotated file age reaches 180 days.</p>

Configuration	In the Puppet template file <code>/etc/cloudian-&lt;version&gt;-puppet/modules/cmc/templates/log4j.xml.erb</code> , this log is configurable in the block that starts with <code>&lt;RollingRandomAccessFile name="APP"</code> . For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).
---------------	---

### 10.1.4. 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 100).

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 URG=0</pre>
Default Rotation and Retention Policy	<p>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).</p> <p>Rotated files are named as <i>firewall.log-YYYYMMDDHH.gz</i>. Rotated files are compressed with <i>gzip</i>.</p> <p>Rotated files are retained for 180 days and then automatically deleted.</p>
Configuration	Rotation of this log is managed by the Linux <i>logrotate</i> utility. In the current version of HyperStore, the rotation settings for the HyperStore firewall log are not configurable.

### 10.1.5. 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> <ul style="list-style-type: none"> <li>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 HyperStore online Help.</li> <li>The S3RequestId value is present only in messages associated with implementing S3 requests.</li> </ul>
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 Retention Policy	<p>Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB <b>or</b> if oldest rotated file age reaches 180 days.</p>
Configuration	In the Puppet template file <i>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-hyperstore.xml.erb</i> , this log is configurable in the block that starts with <i>&lt;RollingRandomAccessFile name="APP"</i> . For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).

#### 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</pre> <ul style="list-style-type: none"> <li>The <i>IpAddressOfClientS3Server</i> is the IP address of the S3 Server node that submits the request to the HyperStore Service.</li> <li>The <i>HttpOperation</i> 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 on the secure delete feature see the description of the configuration property <b>"secure.delete"</b> (page 544).</li> <li>The <i>OriginalUri</i> field shows the group ID, bucket name, and object name from the originating S3 API request, in URI-encoded form ("CloudianTest1", "buser1", and "514kbtcs", respectively, in the example above).</li> <li>The <i>Etag</i> field will be "0" for operations other than PUT.</li> </ul>
Log Entry Example	<pre>2016-10-27 15:18:18,031 10.20.2.52 6e4c6884-a4a2-1238-a908-525400c5e557  200 PUT /file/CloudianTest1%2Fbuser1%2Ftest100b  /ccloudian2/hsfs/1IjBeBudSCVmsYbKdPV8Ns/4a3ceb36ee344e1ebd43ed413b310bc8/046/ 075/56017837606746367338485930470043970723.1477552697400  100 854411 7b2a7abdfdaa1a01c33432b5c41e0939</pre>
Logging Level	Not applicable
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 300MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p>

	Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 3GB <b>or</b> if oldest rotated file age reaches 180 days.
Configuration	In the Puppet template file <code>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-hyperstore.xml.erb</code> , this log is configurable in the block that starts with <code>&lt;RollingRandomAccessFile name="REQ"&gt;</code> . For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).

#### 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 Command# ObjectKey ObjectFilePath</pre> <p>This log has entries when an <a href="#">hsstool cleanup</a> or <a href="#">hsstool cleanupec</a> 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.</p>
Log Entry Example	<pre>2018-02-28 05:57:25,743 1 buser1/obj1 /var/lib/cloudian/hsfs/Sa1A110Vu6oCThSSRafvH/7cf10597b0360421d7564e7c248b2445/165/206/16398559635448146388914806157301167971.1476861996241</pre>
Default Logging Level	INFO
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>Rotated files are named as <code>cloudian-hyperstore-cleanup.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></p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB <b>or</b> if oldest rotated file age reaches 180 days.</p>
Configuration	In the Puppet template file <code>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-hyperstore.xml.erb</code> , this log is configurable in the block that starts with <code>&lt;RollingRandomAccessFile name="CLEANUP"&gt;</code> . For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).

#### HyperStore Service repair log (*cloudian-hyperstore-repair.log*)

Location	On every node, <code>/var/log/cloudian/cloudian-hyperstore-repair.log</code>
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS Type Command# Coordinator RepairEndpoint StreamFromEndpoint ObjectKey ObjectFilePath ObjectSize Md5Hash RepairLatencyMillisecs Status</pre> <p>This log has entries when a repair operation results in an attempt to repair files on the node. A repair operation that determines that no repairs are needed on the node will not cause any entries to this log.</p> <p>The Type field value is one of:</p> <ul style="list-style-type: none"> <li>RR -- regular repair for replica</li> <li>PRR -- proactive repair for replicas</li> </ul>

	<ul style="list-style-type: none"> <li>• REC -- regular repair for erasure coded data</li> <li>• PREC -- proactive repair for erasure coded data</li> <li>• UECD -- update of EC digest fields</li> </ul> <p>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.</p> <p>The Status field indicates the status of the repair attempt for the object and is one of:</p> <ul style="list-style-type: none"> <li>• OK -- Successfully repaired</li> <li>• PRQUEUE -- Failed to repair, but successfully added to the proactive repair queue</li> <li>• ERROR / STREAMERROR / STREAMTIMEOUT -- Failed to repair and did not add to the proactive repair queue</li> </ul> <div style="background-color: #d4edda; padding: 10px; border: 1px solid #c3e6cb;"> <p><b>Note</b> ERROR / STREAMERROR / STREAMTIMEOUT status will be recorded in this log only for erasure coded objects. For information about failed replicated object repairs see the <i>hss-error.log</i> file.</p> </div>
Log Entry Example	<pre>2018-02-13 00:58:55,816 RR 2 10.20.2.34 10.20.2.35 10.20.2.34 newb1/efile586 /home/disk2/hsfs/ eFq94PDeBSa5RV1sZMbh2/2479a0125408240ff77bd1b0a8ea28b0/082/083/ 13762854026742787754176464820359359646.1484021921915 10000  fe912baa5d49737c70624b5b82328838 22 OK</pre>
Default Logging Level	INFO
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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></p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB <b>or</b> if oldest rotated file age reaches 180 days.</p>
Con-figuration	<p>In the Puppet template file <i>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-hyperstore.xml.erb</i>, this log is configurable in the block that starts with <i>&lt;RollingRandomAccessFile name="REPAIR"</i>. For setting descriptions see "<b>Log Configuration Settings</b>" (page 626).</p>

#### HyperStore Service repair error and heal logs (*hss-error.log* and *hss-heal.log*)

The *hss-error.log* and *hss-heal.log* are for use by Cloudian Support if they are helping you to troubleshoot repair failures.

#### 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](#).

### 10.1.6. HyperStore Shell 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 **"Security Features"** (page 89).

Location	On every node, <code>/var/log/hsh/hsh.log</code>
Log Entry Examples	<p>HSH user login:</p> <pre>time="2019-09-04T20:26:21-07:00" level=info msg="New Session" session=15C16CFDA4A46863 user=sa_admin</pre> <p>HSH user running a command:</p> <pre>time="2019-09-04T21:22:30-07:00" level=info msg="Running command." args="&lt;s:setup&gt;" command=hspkg session=15C16CFDA4A46863 type=interactive  time="2019-09-04T21:22:30-07:00" 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  time="2019-09-04T21:24:38-07:00" level=info msg="Command complete." args="&lt;s:setup&gt;" command=hspkg session=15C16CFDA4A46863 status=0 type=interactive</pre>
Default Rotation and Retention Policy	<p>Rotation occurs monthly or if the live file size reaches 500MB.</p> <p>Rotated files are named as <code>hsh.log.&lt;n&gt;</code>, with <code>hsh.log.1</code> being the most recently rotated file. The most recently rotated file is not compressed; all other rotated files are compressed with <code>gzip</code> and the file names will have a <code>.gz</code> suffix.</p> <p>12 rotated files are retained. Older files are automatically deleted.</p>
Configuration	Rotation of this log is managed by the Linux <code>logrotate</code> utility. In the current version of HyperStore, the rotation settings for the HyperStore shell log are not configurable.

### 10.1.7. IAM Service Logs

*IAM application log (cloudian-iam.log)*

Location	On every node, <code>/var/log/cloudian/cloudian-iam.log</code>
Log Entry Format	<p><b>Note</b> For an IAM overview see <b>"HyperStore Support for the AWS IAM API"</b> (page 991).</p> <pre>yyyy-mm-dd HH:mm:ss,SSS PriorityLevel[ThreadId]MessageCode ClassName:MESSAGE</pre> <ul style="list-style-type: none"> <li>The MessageCode uniquely identifies the log message, for messages of level WARN</li> </ul>

	or higher. For documentation of specific message codes including recommended operator action, see the HyperStore online Help.
Log Entry Example	2018-02-16 10:16:36,246 INFO[qtp214187874-41] CloudianHFactory:Creating DynamicKS for: AccountInfo
Default Logging Level	INFO
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p> <p>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.</p>
Configuration	In the Puppet template file <i>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-iam.xml.erb</i> , this log is configurable in the block that starts with <i>&lt;RollingRandomAccessFile name="IAMAPP"</i> . For setting descriptions see " <b>Log Configuration Settings</b> " (page 626).

*IAM request log (cloudian-iam-request-info.log)*

Location	<p>On every node, <i>/var/log/cloudian/cloudian-iam-request-info.log</i></p> <div style="background-color: #d9ead3; padding: 10px; border: 1px solid #ccc;"> <p><b>Note</b> This log records Security Token Service (STS) requests as well as IAM requests.</p> </div>
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> <ul style="list-style-type: none"> <li>For the <i>IamUserId</i> field: <ul style="list-style-type: none"> <li>If request is by IAM user, this is the <i>UserId</i> of the IAM user</li> <li>If request is by a non-SAML role session, this is the <i>UserId</i> of the IAM user who assumed the role</li> <li>If request is by a SAML role session, this is the <i>Subject</i> field value from the SAML Assertion (the user identifier)</li> <li>If request is by an account root user, this field is empty</li> </ul> </li> <li>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.</li> <li>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&amp;RoleSessionId&amp;TempCredentialsAccessKey</i>). Otherwise this field is empty.</li> </ul>
Log Entry	2020-08-10 18:58:59,607 10.20.2.34 679d95846fb0f0047f5926ba16546552 testu159986

Examples	<pre>myGroup8732 iam:PutRolePolicy    200   6  2020-08-10 18:58:59,619 10.20.2.34 679d95846fb0f0047f5926ba16546552 testu159986  myGroup8732 sts:AssumeRole aidcc54d3e60de2a74e89ad639561df0    200   arn:aws:iam::679d95846fb0f0047f5926ba16546552:role/iammypath/rolen134094&amp; rolesn54301&amp;asicbd8ef4bdd6e7e03c 118</pre>
Default Logging Level	INFO
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 100MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 2GB <b>or</b> if oldest rotated file age reaches 180 days.</p>
Configuration	<p>In the Puppet template file <i>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-iam.xml.erb</i>, this log is configurable in the block that starts with <i>&lt;RollingRandomAccessFile name="IAMREQ"</i>. For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).</p>

### 10.1.8. 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	<pre>yyyy-mm-dd HH:mm:ss,SSS PriorityLevel [ThreadId] ClassName:MESSAGE</pre>
Log Entry Example	<pre>2017-05-04 11:57:03,698 WARN [pool-2-thread-7] LogFileTailerListener:Log file not found for service:cron</pre>
Default Logging Level	WARN
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB <b>or</b> if oldest rotated file age reaches 180 days.</p>
Configuration	<p>In the Puppet template file <i>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudianagent/templates/log4j-agent.xml.erb</i>, this log is configurable in the block that starts with <i>RollingRandomAccessFile name="APP"</i>. For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).</p>

*Monitoring Data Collector application log (cloudian-datacollector.log)*

Location	On Monitoring Data Collector node, <i>/var/log/cloudian/cloudian-datacollector.log</i>
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS PriorityLevel[ThreadId]MessageCode ClassName:MESSAGE</pre> <p>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 HyperStore online Help.</p>
Log Entry Example	<pre>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</pre>
Default Logging Level	WARN
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>Rotated files are named as <i>cloudian-datacollector.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>.</p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB <b>or</b> if oldest rotated file age reaches 180 days.</p>
Configuration	In the Puppet template file <i>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-datacollector.xml.erb</i> , this log is configurable in the block that starts with <i>&lt;RollingRandomAccessFile name="APP"</i> . For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).

## 10.1.9. Phone Home (Smart Support) Log

*Phone Home application log (cloudian-phonehome.log)*

Location	On 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> <p>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 HyperStore online Help..</p>
Log Entry Example	<pre>2017-05-04 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
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB <b>or</b> if oldest rotated file age reaches 180 days.</p>

Configuration	In the Puppet template file <code>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-phonehome.xml.erb</code> , this log is configurable in the block that starts with <code>&lt;RollingRandomAccessFile name="APP"&gt;</code> . For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).
---------------	--

### 10.1.10. Redis and Redis Monitor Logs

#### *Redis Credentials application log (redis-credentials.log)*

Location	On Redis Credentials nodes, <code>/var/log/redis/redis-credentials.log</code>
Log Entry Format	<pre>PID.role dd MMM hh:mm:ss.ms loglevel MESSAGE</pre> <p>The role will usually be "M" for master or "S" for slave.</p> <p>The loglevel will be "*" for NOTICE or "#" for ERROR.</p>
Log Entry Example	<pre>18290:S 28 Jul 01:23:42.416 # Connection with master lost.</pre>
Default Logging Level	NOTICE
Default Rotation Policy	Not rotated by default. You can set up rotation by using <a href="#">logrotate</a> .
Configuration	<p>Redis Credentials application logging is configured in the main Redis configuration file. The file name depends on the Redis node type -- master or slave. These templates are on the Puppet master node, under <code>/etc/cloudian-&lt;version&gt;-puppet/modules/redis/templates/</code>:</p> <ul style="list-style-type: none"> <li>Redis Credentials master node: <code>redis-credentials.conf.erb</code></li> <li>Redis Credentials slave node: <code>redis-credentials-slave.conf.erb</code></li> </ul> <p>The only configurable logging settings are the log file name and the logging level. See the commenting in the configuration file for more detail.</p>

#### *Redis QoS application log (redis-qos.log)*

Location	On Redis QoS nodes, <code>/var/log/redis/redis-qos.log</code>
Log Entry Format	<pre>PID.role dd MMM hh:mm:ss.ms loglevel MESSAGE</pre> <p>The role will usually be "M" for master or "S" for slave.</p> <p>The loglevel will be "*" for NOTICE or "#" for ERROR.</p>
Log Entry Example	<pre>24401:M 28 Jul 01:41:46.963 * Calling fsync() on the AOF file.</pre>
Default Logging Level	NOTICE
Default Rotation Policy	Not rotated by default. You can set up rotation by using <a href="#">logrotate</a> .
Configuration	Redis QoS application logging is configured in the main Redis configuration file. The file

	<p>name depends on the Redis node type -- master or slave. These templates are on the Puppet master node, under <code>/etc/cloudian-&lt;version&gt;-puppet/modules/redis/templates/</code>:</p> <ul style="list-style-type: none"> <li>Redis QoS master node: <code>redis-qos.conf.erb</code></li> <li>Redis QoS slave node: <code>redis-qos-slave.conf.erb</code></li> </ul> <p>The only configurable logging settings are the log file name and the logging level. See the commenting in the configuration file for more detail.</p>
--	--

#### Redis request logs (`redis-{s3,admin,hss}-tx.log`)

Location	If enabled, these logs are written to <code>/var/log/cloudian/redis-{s3,admin,hss}-tx.log</code> on the S3 Service, Admin Service, and/or HyperStore nodes (that is, these request logs are written on the <b>client side</b> as S3, Admin, and HyperStore service instances send requests to Redis).
Log Entry Format	<code>yyyy-mm-dd HH:mm:ss,SSS S3RequestId MESSAGE</code>
Log Entry Example	<code>2016-11-17 23:54:01,695 CLIENT setname ACCOUNT_GROUPS_M</code>
Default Logging Level	<p>INFO</p> <div style="background-color: #e6f2e6; padding: 10px; border: 1px solid #ccc;"> <p><b>Note</b> The default logging level of INFO disables these logs. If you want these logs to be written, you must edit the Puppet template files <code>log4j-s3.xml.erb</code> (for logging S3 Service access to Redis), <code>log4j-admin.xml.erb</code> (for logging Admin Service access to Redis), and/or <code>log4j-hyperstore.xml.erb</code> (for logging HyperStore Service access to Redis). Find the <code>AsyncLogger name="redis.clients.jedis"</code> block and change the <code>level</code> from "INFO" to "TRACE". Then do a Puppet push, and then restart the relevant services (S3-Admin and/or HyperStore).</p> </div>
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>Rotated files are named as <code>redis-{s3,admin,hss}-tx.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 <code>gzip</code>.</p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB <b>or</b> if oldest rotated file age reaches 180 days.</p>
Configuration	In the Puppet template files <code>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-{s3,admin,hss}.xml.erb</code> , this log is configurable in the block that starts with <code>&lt;RollingRandomAccessFile name="REDISTX"</code> . For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).

#### Redis Monitor application log (`cloudian-redismon.log`)

Location	On Redis Monitor nodes, <code>/var/log/cloudian/cloudian-redismon.log</code>
Log Entry Format	<p><code>yyyy-mm-dd HH:mm:ss,SSS PriorityLevel[ThreadId]MessageCode ClassName:MESSAGE</code></p> <p>The MessageCode uniquely identifies the log message, for messages of level WARN or</p>

	higher. For documentation of specific message codes including recommended operator action, see the HyperStore 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	<p>Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size reaches 100MB or if oldest rotated file age reaches 180 days.</p>
Configuration	In the Puppet template file <i>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-redismon.xml.erb</i> , this log is configurable in the block that starts with <i>&lt;RollingRandomAccessFile name="APP"&gt;</i> . For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).

### 10.1.11. 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> <p>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 HyperStore online Help.</p>
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	<p>Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p> <p>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.</p>
Configuration	In the Puppet template file <i>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-s3.xml.erb</i> , this log is configurable in the block that starts with <i>&lt;RollingRandomAccessFile name="S3APP"&gt;</i> . For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).

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  Group CanonicalUserId IamUserId RoleSessionArn RoleSessionId  TempCredentialsAccessKey</pre> <ul style="list-style-type: none"> <li>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 <a href="#">health check request</a>, the Operation field indicates "healthCheck". In the case of requests submitted to the S3 Service by a <a href="#">system cron job</a>, the Operation field indicates the name of the cron job action, such as "systemBatchDelete").</li> <li>The Etag field is the Etag value from the response, if applicable to the request type. For information about Etag see for example <a href="#">Common Response Headers</a> 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.</li> <li>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 <a href="#">Error Responses</a> from the Amazon S3 REST API spec.</li> </ul> <div style="background-color: #d9ead3; padding: 10px; margin: 10px 0;"> <p><b>Note</b> In the case where the Operation field value is <i>deleteObjects</i>, the ErrorCode field will be formatted as <i>objectname1-errorcode1,objectname2-errorcode2,objectname3-errorcode3...</i>, and the object names will be URL-encoded. If there are no errors the field is formatted as <i>objectname1-0,objectname2-0,objectname3-0...</i></p> </div> <ul style="list-style-type: none"> <li>The SourceBucketName/UrlEncodedSourceObjectName field is populated only for copyObject and uploadPartCopy operations and is empty for other operation types.</li> <li>If you want the ClientIpAddress, BucketOwnerUserID, and RequestorUserId to be removed from the copies of the S3 request log that get uploaded to Cloudian Support as part of the Smart Support feature and on-demand Node Diagnostics feature, set <b>"phonehome_gdpr"</b> (page 525) to true in <a href="#">common.csv</a>. Doing so will not remove these fields from the original S3 request logs on your HyperStore nodes -- just from the log file copies that get sent to Cloudian. For more information on these support feature see <b>"Smart Support and Diagnostics Feature Overview"</b> (page 190).</li> <li>For the iamUserId field: <ul style="list-style-type: none"> <li>If request is by IAM user, this is the UserId of the IAM user</li> <li>If request is by a non-SAML role session, this is the UserId of the IAM user who assumed the role</li> <li>If request is by a SAML role session, this is the Subject field value from the SAML Assertion (the user identifier)</li> </ul> </li> </ul>

	<ul style="list-style-type: none"> <li>◦ If request is by an account root user, this field is empty</li> <li>• If request is by a role session, the RoleSessionArn, RoleSessionId, and TempCredentialsAccessKey fields provide information about the role session making the request. Otherwise these fields are empty.</li> </ul>
Log Entry Example	<pre>2020-10-13 06:01:25,655 10.50.10.15 PubsUser1 putObject pubs.bucket1  PubsUser1 972 1638336 135 0 1639443 145360 IAM+Roles+and+SAML+Design.docx  200 a12d2f8e-fdd2-1bcd-92ba-0026b95d08b9 83043a9f55a8a9880fb7c63d6130d473  0  Pubs edfaf89260798e4c9d49064e7ef4d0c3   </pre>
Logging Level	Not applicable
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 100MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 2GB <b>or</b> if oldest rotated file age reaches 180 days.</p>
Configuration	<p>In the Puppet template file <code>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-s3.xml.erb</code>, this log is configurable in the block that starts with <code>&lt;RollingRandomAccessFile name="S3REQ"</code>. For setting descriptions see "<b>Log Configuration Settings</b>" (page 626).</p>

**Note** For information on setting up an Elastic Stack node and streaming *cloudian-request-info.log* data to that node to support integrated analysis of your S3 request traffic, see "**Setting Up Elastic Stack for S3 Request Traffic Analysis**" (page 631).

#### 10.1.11.0.1. Logging the True Originating Client IP Address

If you use a load balancer in front of your S3 Service (as would typically be the case in a production environment), then the *ClientIpAddress* in your S3 request logs will by default 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, your options depend on what load balancer you're using.

If your load balancer is HAProxy or a different load balancer that supports the PROXY Protocol, enable S3 support for the PROXY Protocol (see "**s3\_proxy\_protocol\_enabled**" (page 530) in [common.csv](#)) and configure your load balancer to use the PROXY Protocol for relaying S3 requests to the S3 Service. Consult with your Cloudian Sales Engineering or Support representative for guidance on load balancer configuration.

If your load balancer does not support the PROXY Protocol:

1. Configure your load balancers so that they pass the HTTP *X-Forwarded-For* header to the S3 Service. This is an option only if you run your load balancers in "HTTP mode" rather than "TCP mode". Consult with your Cloudian Sales Engineering or Support representative for guidance on load balancer configuration.
2. Configure your S3 Service to support the *X-Forwarded-For* header. You can enable S3 Service support for this header by editing the configuration file *s3.xml.erb* on your Puppet master node. The needed configuration lines are already in that file; you only need to uncomment them.

Before uncommenting:

```
<!-- Uncomment the block below to enable handling of X-Forwarded- style headers -->
<!--
<Call name="addCustomizer">
  <Arg><New class="org.eclipse.jetty.server.ForwardedRequestCustomizer"/></Arg>
</Call>
-->
```

After uncommenting:

```
<!-- Uncomment the block below to enable handling of X-Forwarded- style headers -->
<Call name="addCustomizer">
<Arg><New class="org.eclipse.jetty.server.ForwardedRequestCustomizer"/></Arg>
</Call>
```

After making this configuration edit, [do a Puppet push and restart the S3 Service](#) to apply your change.

#### Auto-Tiering request log (*cloudian-tiering-request-info.log*)

Location	<p>On every node, <i>/var/log/cloudian/cloudian-tiering-request-info.log</i></p> <div style="border: 1px solid black; padding: 10px; background-color: #f0f0f0;"> <p><b>Note</b> For regular auto-tiering that occurs on a defined schedule this request logging occurs on the same node that is running the HyperStore system cron jobs. In the special case of "bridge mode" auto-tiering, the request logging is distributed across all your S3 nodes -- 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.</p> </div>
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS Command Protocol SourceBucket/Object  SourceObjectVersion TargetBucket TargetObjectVersion ObjectSize  TotalRequestSize Status DurationMicrosecs</pre>
Log Entry Example	<pre>2017-10-31 20:39:37,282 MOVEOBJECT AZURE cbucket/a.c null  testbucket null 112 115 COMPLETED 83000</pre>
Logging Level	Not applicable
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB <b>or</b> if oldest rotated file age reaches 180 days.</p>
Configuration	<p>In the Puppet template file <i>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-s3.xml.erb</i>, this log is configurable in the block that starts with <i>&lt;RollingRandomAccessFile name="TIER"</i>. For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).</p>

#### Cross-Region Replication request log (*cloudian-crr-request-info.log*)

Location	<p>On every node, <i>/var/log/cloudian/cloudian-crr-request-info.log</i></p> <div> <p><b>Note</b> 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 "<b>Cross-Region Replication Feature Overview</b>" (page 186).</p> </div>
Log Entry Format	<pre>yyyy-mm-dd HH:mm:ss,SSS SourceBucket/Object ObjectVersionId DestinationBucket CrrOperation Status DurationMillisecs Size</pre> <p>The Status will be one of:</p> <ul style="list-style-type: none"> <li>COMPLETED -- The object was successfully replicated to the destination bucket.</li> <li>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 <a href="#">Alerts</a> page.</li> <li>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>.</li> </ul>
Log Entry Example	<pre>2020-03-04 15:04:58,423 prod-2020-01-16/0299-0166180000362773.pdf fe15a1de-de25-f8af-9a28-b4a9fc08ca7e prod-backup-2020-01-16 REPLICATEOBJECT COMPLETED 61 328471</pre>
Logging Level	Not applicable
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB <b>or</b> if oldest rotated file age reaches 180 days.</p>
Configuration	<p>In the Puppet template file <i>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-s3.xml.erb</i>, this log is configurable in the block that starts with <i>&lt;RollingRandomAccessFile name="CRR"</i>. For setting descriptions see "<b>Log Configuration Settings</b>" (page 626).</p>

*WORM audit log (s3-worm.log)*

Location	<p>On every node, <code>/var/log/cloudian/s3-worm.log</code></p> <div> <b>Note</b> For information about the WORM feature see <b>"WORM (Object Lock)"</b> (page 121). </div>
Log Entry Format	<pre>DateTime Hostname S3RequestId S3Operation Headers Bucket Object  ObjectVersionId CanonicalUserId CanonicalIamUserId StatusCode  StatusMessage</pre> <p>The S3Operation will be any of:</p> <ul style="list-style-type: none"> <li>• 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>)</li> <li>• A regular S3 operation that includes object lock related headers (such as a <i>PUT Bucket</i> request that includes an <code>x-amz-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>.</li> <li>• A <i>DELETE Object Version</i> request in regard to a locked object version.</li> </ul> <p>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:</p> <ul style="list-style-type: none"> <li>• First character: "T" for object lock is enabled on bucket</li> <li>• Second character: "G" for Governance mode or "C" for Compliance mode</li> <li>• Third character: "D" or "Y" for retention period unit of measurement (days or years)</li> <li>• Remaining characters: Integer for number of days or years</li> </ul> <p>Example 1: <i>TGD30</i> for a Governance mode configuration with 30 day retention period</p> <p>Example 2: <i>T</i> for a bucket with object lock enabled, but no default object lock configuration</p> <p>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.</p>
Log Entry Example	<pre>2019-10-06 09:59:30,767 arcturus a9d7e5b1-e85a-11e9-8519-52540014b047  s3:GetBucketObjectLockConfiguration TGD90 newbucket    b584fb57480af5108e32d17f10c5cb7b  200 OK</pre>
Default Logging Level	INFO
Default Rotation and Retention Policy	<p>Rotation occurs if live file size reaches 10MB. Rotation also occurs at end of each day, regardless of live file size.</p> <p>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>.</p> <p>Deletion of oldest rotated log file occurs if aggregate rotated files size (after compression) reaches 100MB <b>or</b> if oldest rotated file age reaches 180 days.</p>

Configuration	In the Puppet template file <code>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-s3.xml.erb</code> , this log is configurable in the block that starts with <code>&lt;RollingRandomAccessFile name="S3WORM"</code> . For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).
---------------	--

## 10.1.12. SQS Service Logs

### *SQS application log (cloudian-sqs.log)*

Location	On every node, <code>/var/log/cloudian/cloudian-sqs.log</code>  <b>Note</b> For an SQS overview (including information about how to enable the SQS Service, which is disabled by default) see <b>"HyperStore Support for the AWS SQS API"</b> (page 1041).
Log Entry Format	<code>yyyy-mm-dd HH:mm:ss,SSS PriorityLevel[ThreadId]MessageCode ClassName:MESSAGE</code>
Log Entry Example	<code>2019-11-05 02:32:44,453 INFO[SQSRetentionCheckerThread - pool-5-thread-1] SQSMessageRetentionChecker:Another thread processing. Do nothing...</code>
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 <code>cloudian-sqs-req.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.
Configuration	In the Puppet template file <code>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-sqs.xml.erb</code> , this log is configurable in the block that starts with <code>&lt;RollingRandomAccessFile name="SQSAPP"</code> . For setting descriptions see <b>"Log Configuration Settings"</b> (page 626).

### *SQS request log (cloudian-sqs-request.log)*

Location	On every node, <code>/var/log/cloudian/cloudian-sqs-request.log</code>  <b>Note</b> For an SQS overview (including information about how to enable the SQS Service, which is disabled by default) see <b>"HyperStore Support for the AWS SQS API"</b> (page 1041).
Log Entry Format	<code>yyyy-mm-dd HH:mm:ss,SSS PriorityLevel[ThreadId]MessageCode ClassName:MESSAGE</code>  The MESSAGE includes the client IP address, the user ID, and the request type (the SQS

	"action").
Log Entry Example	2019-07-11 13:50:21,736 INFO[qtp1548962651-122] RequestLogger:2019-07-11 13:50:21,732 10.20.2.61 user1 CreateQueue  testQueue 200 4
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 <b>or</b> if oldest rotated file age reaches 180 days.
Configuration	In the Puppet template file <i>/etc/cloudian-&lt;version&gt;-puppet/modules/cloudians3/templates/log4j-sqs.xml.erb</i> , this log is configurable in the block that starts with <i>&lt;RollingRandomAccessFile name="SQSREQ"</i> . For setting descriptions see " <b>Log Configuration Settings</b> " (page 626).

## 10.2. Log Configuration Settings

The S3 Service, HyperStore Service, Redis Monitor, Admin Service, Monitoring Data Collector, Monitoring Agent, and CMC each have their own XML-formatted *log4j-\*.xml.erb* configuration template in which you can adjust logging settings. Within a *log4j-\*.xml.erb* file, specific logs -- such as the S3 application log and the S3 request log -- are configured by named instances of *RollingRandomAccessFile*. The "**HyperStore Logs**" (page 605) overview topic indicates the specific *log4j-\*.xml.erb* file and the specific *RollingRandomAccessFile* name by which each log is configured (for example the S3 application log is configured by the *RollingRandomAccessFile* instance named "S3APP" in the *log4j-s3.xml.erb* file).

**Note** After making any configuration file edits, be sure to [trigger a Puppet sync-up and then restart the affected service](#) (for example, the S3 Service if you've edited the *log4j-s3.xml.erb* file).

Within a particular log's *RollingRandomAccessFile* instance there are these editable settings:

- **PatternLayout pattern="<pattern>"** — The log entry format. This flexible formatting configuration is similar to the *printf* function in C. For detail see [PatternLayout](#) from the online Apache Log4j2 documentation.
- **TimeBasedTriggeringPolicy interval="<integer>"** — Roll the log after this many days pass. (More precisely, the log rolls after *interval* number of time units pass, where the time unit is the most granular unit of the date pattern specified within the *filePattern* element — which in the case of all HyperStore logs' configuration is a day). Defaults to rolling once a day if *interval* is not specified. All HyperStore logs use the default of one day.
- **SizeBasedTriggeringPolicy size="<size>"** — Roll the log when it reaches this size (for example "10 MB"). Note that this trigger and the *TimeBasedTriggeringPolicy* operate together: the log will be rolled if either the time based trigger **or** the size based trigger occur.

- **IfLastModified age="<interval>"** — When a rolled log file reaches this age the system automatically deletes it (for example "180d").
- **IfAccumulatedFileSize exceeds="<size>"** — When the aggregate size (after compression) of rolled log files for this log reaches this size, the system automatically deletes the oldest rolled log file (for example "100 MB"). Note that this setting works together with the *IfLastModified* setting -- old rolled log files will be deleted if either the age based trigger or the aggregate size based trigger occur.

**Note** Each *RollingRandomAccessFile* instance also includes a *DefaultRolloverStrategy* *max-x="<integer>"* parameter which specifies the maximum number of rolled files to retain from a single day's logging. However, by default this parameter is not relevant for HyperStore because HyperStore logs are configured such that the *IfAccumulatedFileSize* trigger will be reached before the *DefaultRolloverStrategy* trigger.

For each log's default value for the settings above, see the **"HyperStore Logs"** (page 605) overview topic.

In the *log4j-\*.xml.erb* files, in addition to *RollingRandomAccessFile* instances there are also **Logger** instances. Each *Logger* instance contains an *AppenderRef* element that indicates which log that *Logger* instance applies to, by referencing the log's *RollingRandomAccessFile* name (for example *AppenderRef ref="S3APP"* means that the *Logger* instance is associated with the S3 application log). Note that multiple *Logger* instances may be associated with the same log — this just means that multiple core components of a service (for example, multiple components within the S3 Service) have separately configurable loggers. If you're uncertain about which *Logger* instance to edit to achieve your objectives, consult with Cloudbian Support.

The *Logger* instances are where you can configure a logging level, using the *level* attribute:

- **level="<level>"** — Logging level. The following levels are supported (only events at the configured level and above will be logged):
  - OFF = Turn logging off.
  - ERROR = Typically a fail-safe for programming errors or a server running outside the normal operating conditions. Any error which is fatal to the service or application. These errors will cause administrator alerts and typically force administrator intervention.
  - WARN = Anything that can potentially cause application oddities, but where the server can continue to operate or recover automatically. Exceptions caught in "catch" blocks are commonly at this level.
  - INFO = Generally useful information to log (service start/stop, configuration assumptions, etc). Info to always have available. Normal error handling, like a user not existing, is an example.
  - DEBUG = Information that is diagnostically helpful.
  - TRACE = Very detailed information to "trace" the execution of a request or process through the code.
  - ALL = Log all levels.

For each log's default log level, see the **"HyperStore Logs"** (page 605) overview topic.

## 10.3. Aggregating Logs to a Central Server

If you wish you can have application logs and request logs from the S3 Service, Admin Service, and HyperStore Service — as well as system logs from the host machines in your cluster — aggregated to a central logging server using *rsyslog*. This is not enabled by default, but you can enable it by editing configuration files.

This procedure sets up your HyperStore logging so that logs are written 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.

**IMPORTANT !** The central logging server must **not** be one of your HyperStore nodes.

**Note** *rsyslog* is included in the HyperStore Appliance and also in standard RHEL/CentOS distributions. This procedure has been tested using *rsyslog* v5.8.10.

**Note** For information on setting up an [Elastic Stack](#) node and streaming S3 request log data to that node to support analysis and visualization of your S3 request traffic, see **"Setting Up Elastic Stack for S3 Request Traffic Analysis"** (page 631).

To aggregate HyperStore application logs, request logs, and system logs to a central logging server follow the instructions below.

**1. On the central logging server, do the following:**

**a)** In the configuration file */etc/rsyslog.conf*, enable UDP by uncommenting these lines:

```
# BEFORE EDITING

#ModLoad imudp
#UDPServerRun 514

# AFTER EDITING

$ModLoad imudp
$UDPServerRun 514
```

**b)** Still on the central logging server, create a file */etc/rsyslog.d/cloudian.conf* and enter these configuration lines in the file:

```
$template CloudianTpl,"%HOSTNAME% %TIMESTAMP:::date-rfc3339% %syslogseverity-text:::uppercase%
%msg%\n"
$template CloudianReqTpl,"%HOSTNAME% %msg%\n"
:programname, isequal, "ADMINAPP" /var/log/cloudian-admin.log;CloudianTpl
:programname, isequal, "ADMINREQ" /var/log/cloudian-admin-request-info.log;CloudianReqTpl
:programname, isequal, "HSAPP" /var/log/cloudian-hyperstore.log;CloudianTpl
:programname, isequal, "HSREQ" /var/log/cloudian-hyperstore-request-info.log;CloudianReqTpl
:programname, isequal, "S3APP" /var/log/cloudian-s3.log;CloudianTpl
:programname, isequal, "S3REQ" /var/log/cloudian-request-info.log;CloudianReqTpl
:programname, isequal, "s3-worm" /var/log/s3-worm.log;CloudianReqTpl
```

**c)** Still on the central logging server, restart *rsyslog* by "service rsyslog restart".

**d)** Still on the central logging server, enable rotation on the centralized HyperStore logs. For example, to use *logrotate* for rotating the HyperStore logs, create a file */etc/logrotate.d/cloudian* and enter the

following configuration lines in the file. (Optionally adjust the rotated file retention scheme — 14 rotations before deletion in the example below — to match your retention policy.)

```
/var/log/cloudian-*.log
{
    daily
    rotate 14
    create
    missingok
    compress
    delaycompress
    sharedscripts
    postrotate
        /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true
    endscript
}
```

## 2. On your HyperStore Puppet Master node, do the following:

**a)** Go to the `/etc/cloudian-7.2.3-puppet/modules/cloudians3/templates/` directory. Here you will edit three configuration files:

- `log4j-admin.xml.erb`
- `log4j-hyperstore.xml.erb`
- `log4j-s3.xml.erb`

In each of these three files **search for and uncomment all sections marked with a "#syslog" tag**. When you are done uncommenting, there should be no remaining "#syslog" tags.

**Also**, in the first #syslog section at the top of each file — nested within the "Properties" block — in addition to uncommenting the #syslog section set the `sysloghost` property to the hostname or IP address or your central syslog server and set the `syslogport` property to the central syslog server's UDP port (which by default is port number is 514).

Here is a before and after example for uncommenting and editing the #syslog section within a "Properties" block. In the BEFORE EDITING text, the commenting boundaries (which need to be removed) are highlighted in **red**. In the AFTER EDITING text, the commenting boundaries have been removed and the central logging server hostname has been set to "regulus".

```
# BEFORE EDITING

<Properties>
  <!-- #syslog
  <Property name="sysloghost">localhost</Property>
  <Property name="syslogport">514</Property>
  -->
</Properties>

# AFTER EDITING

<Properties>
  <Property name="sysloghost">regulus</Property>
  <Property name="syslogport">514</Property>
</Properties>
```

Be sure to uncomment all of the "#syslog" tagged sections in each of the three files. Remember that

only the `#syslog` section within the "Properties" block at the top of each file requires editing an attribute value. The rest of the `#syslog` sections only require uncommenting.

In this example of a `#syslog` section in `log4j-s3.xml.erb` you would remove the commenting boundaries that here are highlighted in **red**.

```
<!-- #syslog
<Syslog name="SYSLOG-S3APP" format="RFC5424" host="${sysloghost}" port="${syslogport}"
  protocol="UDP" appName="S3APP" mdcId="mdc" includeMDC="true" facility="USER" newLine="true">
</Syslog>
<Syslog name="SYSLOG-S3REQ" format="RFC5424" host="${sysloghost}" port="${syslogport}"
  protocol="UDP" appName="S3REQ" mdcId="mdc" includeMDC="true" facility="USER" newLine="true">
</Syslog>
<Syslog name="SYSLOG-S3WORM" format="RFC5424" host="${sysloghost}" port="${syslogport}"
  protocol="UDP" appName="s3-worm" mdcId="mdc" includeMDC="true" facility="USER" newLine="true">
</Syslog>
-->
```

Here is a second example of a section that needs uncommenting, from that same file:

```
<!-- Request Logger -->
<Logger name="com.gemini.cloudian.RequestLogger" additivity="false" level="INFO">
  <AppenderRef ref="S3REQ" />
  <!-- #syslog
  <AppenderRef ref="SYSLOG-S3REQ" />
  -->
</Logger>
```

**b)** Still on your HyperStore Puppet Master node, go to the `/etc/cloudian-<version>-puppet/modules/rsyslog/templates/` directory. Then edit `loghost.conf.erb` to uncomment the following line and replace "`<loghost>`" with the hostname (or IP address) of your central syslog server host:

```
# BEFORE EDITING
#*. * @<loghost>:514

# AFTER EDITING

*. * @regulus:514
```

**IMPORTANT !** The central log host must **not** be one of your HyperStore hosts. (The reason is that if you have one of your HyperStore hosts acting as the central log host, then that host is sending logs to itself which results in a loop and rapid proliferation of log messages.)

**c)** Still on your HyperStore Puppet Master node, use the installer to push your changes to the cluster and to restart the HyperStore Service and the S3 Service. For instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

### 3. Back on the central logging server:

- a)** Confirm that logs are being written in `/var/log/cloudian-*` files.
- b)** Confirm that system messages from HyperStore nodes appear on the log host (for example in `/var/log/messages`). If you want you can proactively test this by running the command "logger test 1" on any HyperStore node.

## 10.4. Setting Up Elastic Stack for S3 Request Traffic Analysis

Subjects covered in this section:

- *Introduction (immediately below)*
- **"Installing Elasticsearch, Kibana, and Logstash"** (page 631)
- **"Installing Filebeat"** (page 634)
- **"Configuring Kibana for Custom Metrics Visualizations"** (page 635)

These instructions describe how to install an Elasticsearch-Logstash-Kibana (ELK) stack on a **single node** that will process S3 request logs (*cloudian-request-info.log* files) from HyperStore nodes.

**Note** These instructions involve acquiring and using technologies from the open source [Elastic Stack](#).

**Note** You may also want to integrate with Elasticsearch for searching HyperStore object metadata (as described in **"Elasticsearch Integration for Object Metadata"** (page 171)). Note that for object metadata search, a minimum of three Elasticsearch nodes are recommended and so the single-node set-up instructions below are not sufficient to that use case.

ELK Cluster Components:

- Elasticsearch: Stores the S3 request log files from HyperStore nodes in the filtered form specified in Logstash
- Logstash: Filters S3 request logs into a format that is searchable by Elasticsearch
- Kibana: Provides a web interface to access Elasticsearch data

HyperStore Cluster Components:

- Filebeat: Forwards S3 request logs from HyperStore nodes to ELK cluster

Prerequisites:

- An existing HyperStore cluster
- A CentOS 7 machine that has Java 8 already installed and Internet access

**Note** To work with HyperStore your **Elasticsearch version must be a 6.x version, 6.6 or newer**. In the installation instructions that follow the version is 6.6.0.

### 10.4.1. Installing Elasticsearch, Kibana, and Logstash

On the host that you want to set up as an ELK node, take the following steps.

#### 10.4.1.1. Installing Elasticsearch

1. Run the following command to import the ES public GPG key into RPM:

```
# sudo rpm --import http://packages.elastic.co/GPG-KEY-elasticsearch
```

2. Run the following command to download the ES RPM file:

```
# wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.6.0.rpm
```

3. Install the RPM by running the following command:

```
# sudo rpm --install elasticsearch-6.6.0.rpm
```

4. Open `/etc/elasticsearch/elasticsearch.yml` in a text editor and uncomment the "network.host" entry and set it to:

```
# network.host: "localhost"
```

5. Also uncomment "cluster.name" and set it to the desired name for the ES cluster so that a cluster can be formed.
6. Save the configuration changes that you made in `elasticsearch.yml`.

To start Elasticsearch, run the following command:

```
# sudo systemctl start elasticsearch
```

To have Elasticsearch start automatically on boot:

```
# sudo systemctl enable elasticsearch
```

### 10.4.1.2. Installing Kibana

1. Run the following command to download the Kibana RPM:

```
# wget https://artifacts.elastic.co/downloads/kibana/kibana-6.6.0-x86_64.rpm
```

2. Install the RPM by running the following command:

```
# sudo rpm --install kibana-6.6.0-x86_64.rpm
```

3. If you want to be able to access Kibana from other machines besides the ELK node itself, open `/etc/kibana/kibana.yml`, find and uncomment the "server.host:" line, and set it to the IP address of the ELK node:

```
# server.host: <ELK_NODE_IP_ADDR>
```

To start Kibana, run the following command:

```
# sudo systemctl start kibana
```

To have Kibana start automatically on boot:

```
# sudo systemctl enable kibana
```

### 10.4.1.3. Installing Logstash

1. Run the following command to download the Logstash RPM:

```
# wget https://artifacts.elastic.co/downloads/logstash/logstash-6.6.0.rpm
```

2. Install the RPM by running the following command:

```
# sudo rpm --install logstash-6.6.0.rpm
```

### 10.4.1.4. Configuring Logstash

Under the `/etc/logstash/conf.d` directory create a file `logstash.conf` and paste the following lines into it:

```
input {  
  beats {  
    port => 5044  
  }  
}
```

```

}

filter {
  if ([document_type] == "cloudian-request-info") {
    urldecode {
      field => "message"
    }
    csv {
      id => "cloudian-request-info"
      autogenerate_column_names => false
      separator => "|"
      columns => [
        "timestamp",
        "ipAddress",
        "bucketOwnerUserId",
        "operation",
        "bucketName",
        "contentAccessorUserID",
        "requestHeaderSize",
        "requestBodySize",
        "responseHeaderSize",
        "responseBodySize",
        "totalRequestResponseSize",
        "durationMsec",
        "objectName",
        "httpStatus",
        "s3RequestID",
        "eTag",
        "errorCode",
        "copySource"
      ]
      convert => { "requestHeaderSize"      => "integer" }
      convert => { "requestBodySize"        => "integer" }
      convert => { "responseHeaderSize"     => "integer" }
      convert => { "responseBodySize"       => "integer" }
      convert => { "totalRequestResponseSize" => "integer" }
      convert => { "durationMsec"           => "integer" }
      remove_field => "message"
    }
    date {
      match => ["timestamp", "ISO8601"]
      remove_field => [ "timestamp" ]
    }
    geoip {
      source => "ipAddress"
    }
  }
}

output {
  if "_csvparsefailure" not in [tags] and "_dateparsefailure" not in [tags] {
    elasticsearch {
      hosts => [ "localhost:9200" ]

```

```
document_type => "%{[document_type]}"
index => "logstash-%{+YYYY.MM.DD}"
}
}
}
```

Save the file and exit the text editor.

To start Logstash, run the following command:

```
# sudo systemctl start logstash
```

To have Logstash start automatically on boot:

```
# sudo systemctl enable logstash
```

## 10.4.2. Installing Filebeat

Take the following steps on **each of your HyperStore nodes**.

1. Get the Elastic GPG key:

```
# sudo rpm --import http://packages.elastic.co/GPG-KEY-elasticsearch
```

2. Run the following command to download the Filebeat RPM:

```
# wget https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-6.6.0-x86_64.rpm
```

3. Install the RPM by running the following command:

```
# sudo rpm --install filebeat-6.6.0-x86_64.rpm
```

4. Open `/etc/filebeat/filebeat.yml` and find the “paths:” section. Replace the “`- /var/log/*.log`” line with “`- /var/log/cloudian/cloudian-request-info.log`”, while maintaining the level of indentation.
5. Find the “fields: “ section, uncomment it, and under it add the indented line “`document_type: cloudian-request-info`”, so that the section looks like this

```
fields:
  document_type: cloudian-request-info
```

6. Just above the “fields: “ section that you uncommented add the line “`fields_under_root: true`”, so that the result looks like this:

```
fields_under_root: true
fields:
  document_type: cloudian-request-info
```

7. Find the “`output.elasticsearch:`” section and comment it out along with any entries in that section. You don’t need this section since you will have the log files sent to Logstash instead.
8. Find the “`output.logstash:` “ section and uncomment it. Also uncomment the line “`#hosts: [“localhost:5044”]`” under the “`output.logstash:`” section and replace localhost with the IP address of the ELK node:

```
hosts: [“<ELK_NODE_IP_ADDR>:5044”]
```

9. Save the configuration changes that you made in `filebeat.yml`.

To start Filebeat, run the following command:

```
# sudo systemctl start filebeat
```

To have Filebeat start automatically on boot:

```
# sudo systemctl enable filebeat
```

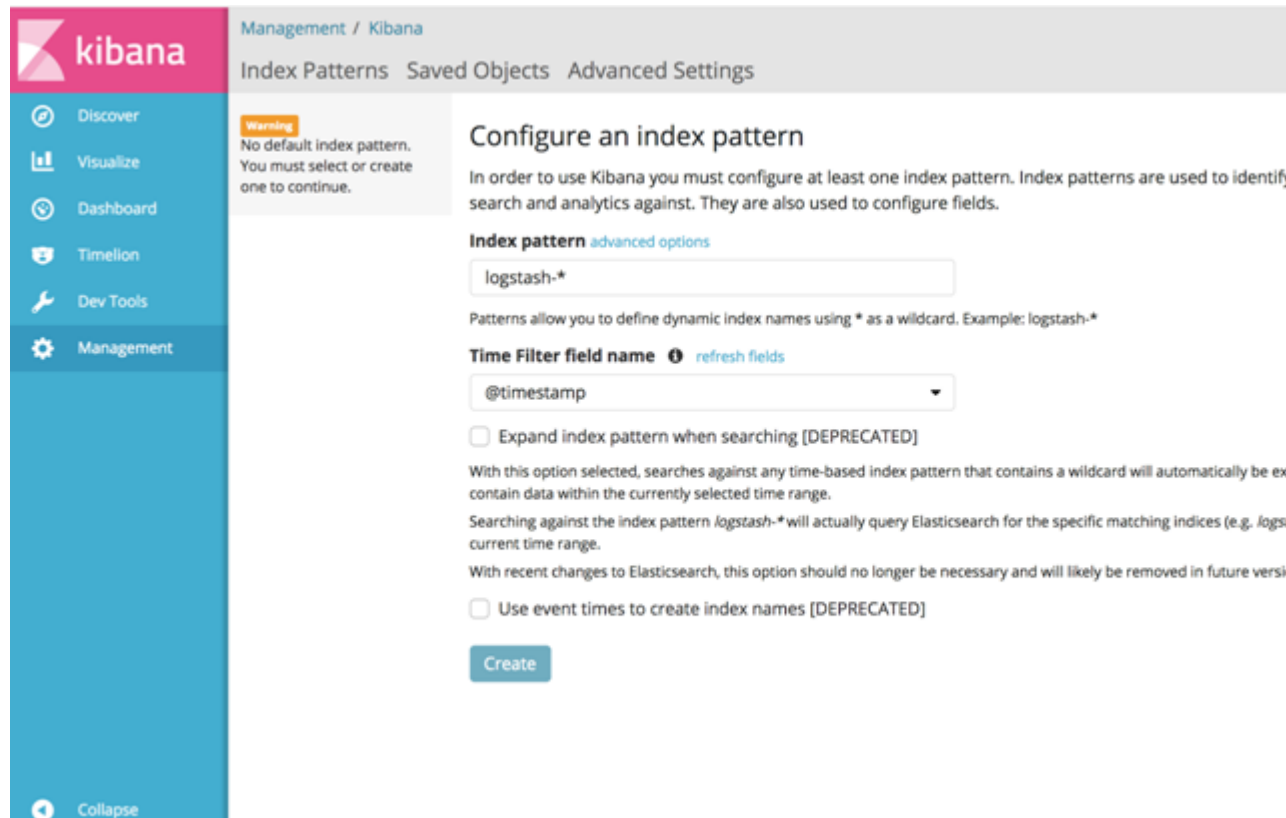
### 10.4.3. Configuring Kibana for Custom Metrics Visualizations

**Note** The following instructions are intended only as an example, and the screen images that you see here may not exactly match what you see in your version of Kibana.

1. Open up a browser that can access the ELK node's IP address and enter the following:

```
http://<ELK_NODE_IP_ADDR>:5601/
```

You should see a page similar to this:

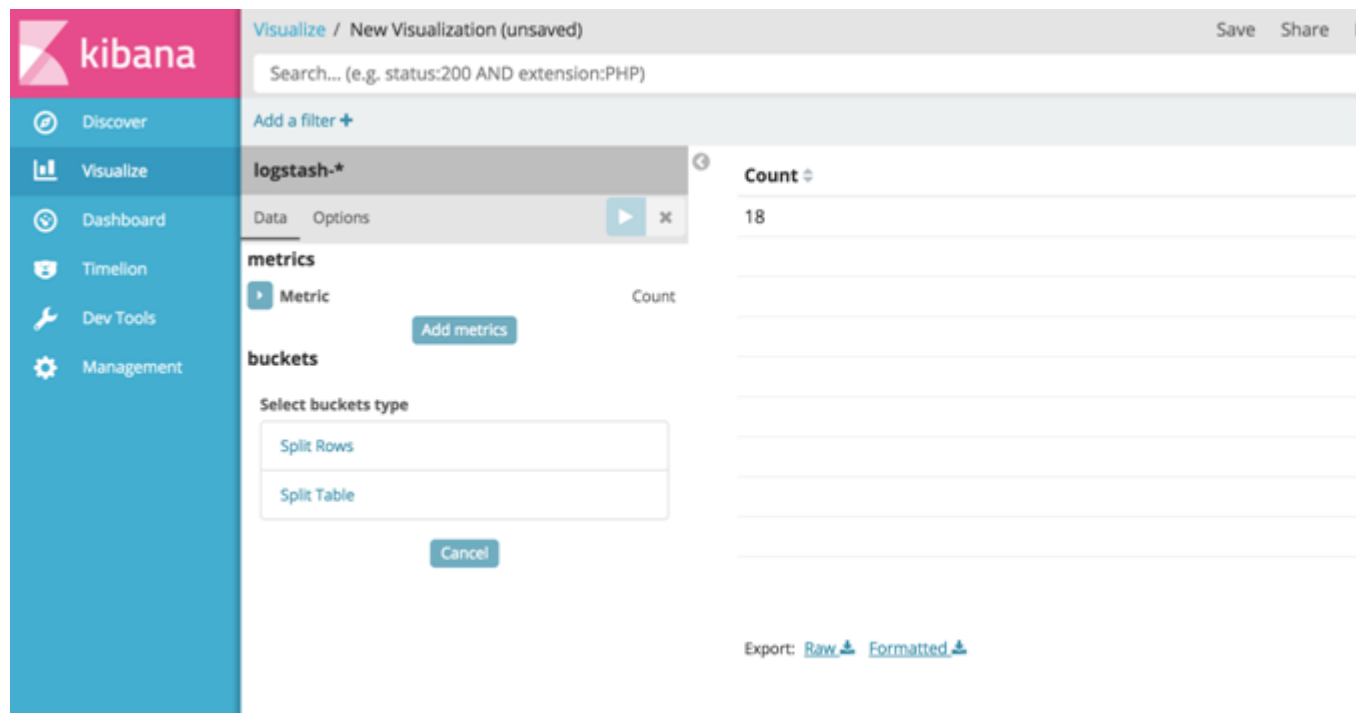


If you don't see a page like this, then Logstash has not yet pushed an index to Elasticsearch, either because of an installation error or because no requests have been made to your cluster yet.

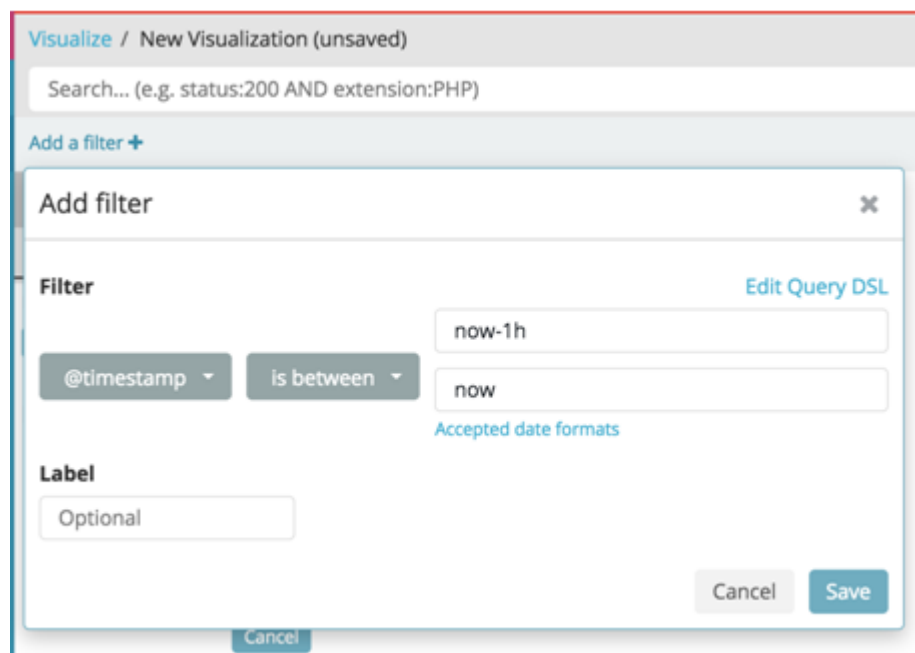
2. Under the "Time Filter field name" drop-down select "@timestamp" and click **Create**.

#### 10.4.3.1. Top 5 users in # of requests the last hour

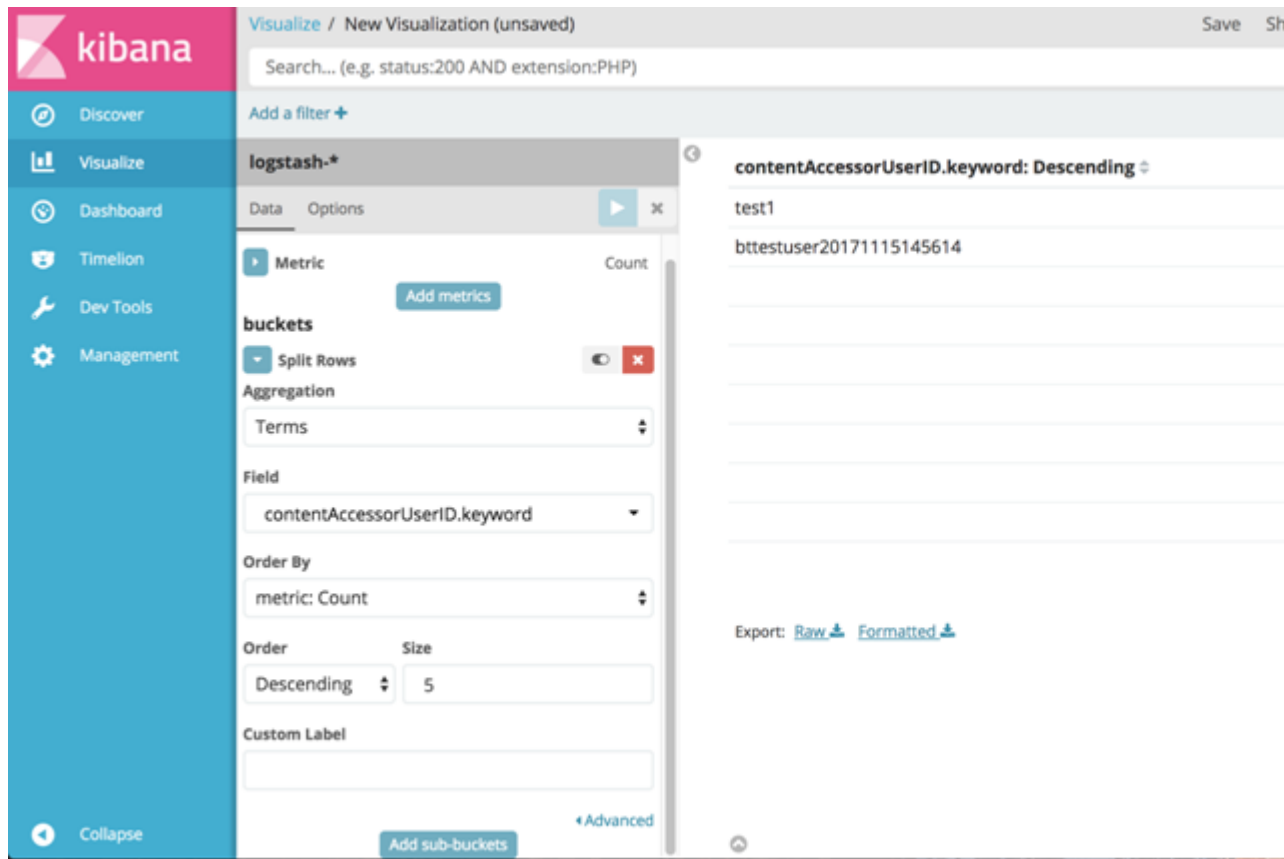
1. To create custom metrics, go to "Visualize" in the sidebar, click **Create a visualization**, select "Data Table", and then select the "logstash-\*" index. You should then be at a screen like this:



- At the top right of the screen, make sure the time setting is set to “Last 1 hour”. If not, click on it, then select “Quick”, and then select “Last 1 hour” from the list of options.
- Click **Add a filter** under the search bar, select the “@timestamp” field, select the “is between” operator, enter “now-1h” in the “from” field, enter “now” in the “to” field, and then click **Save**:



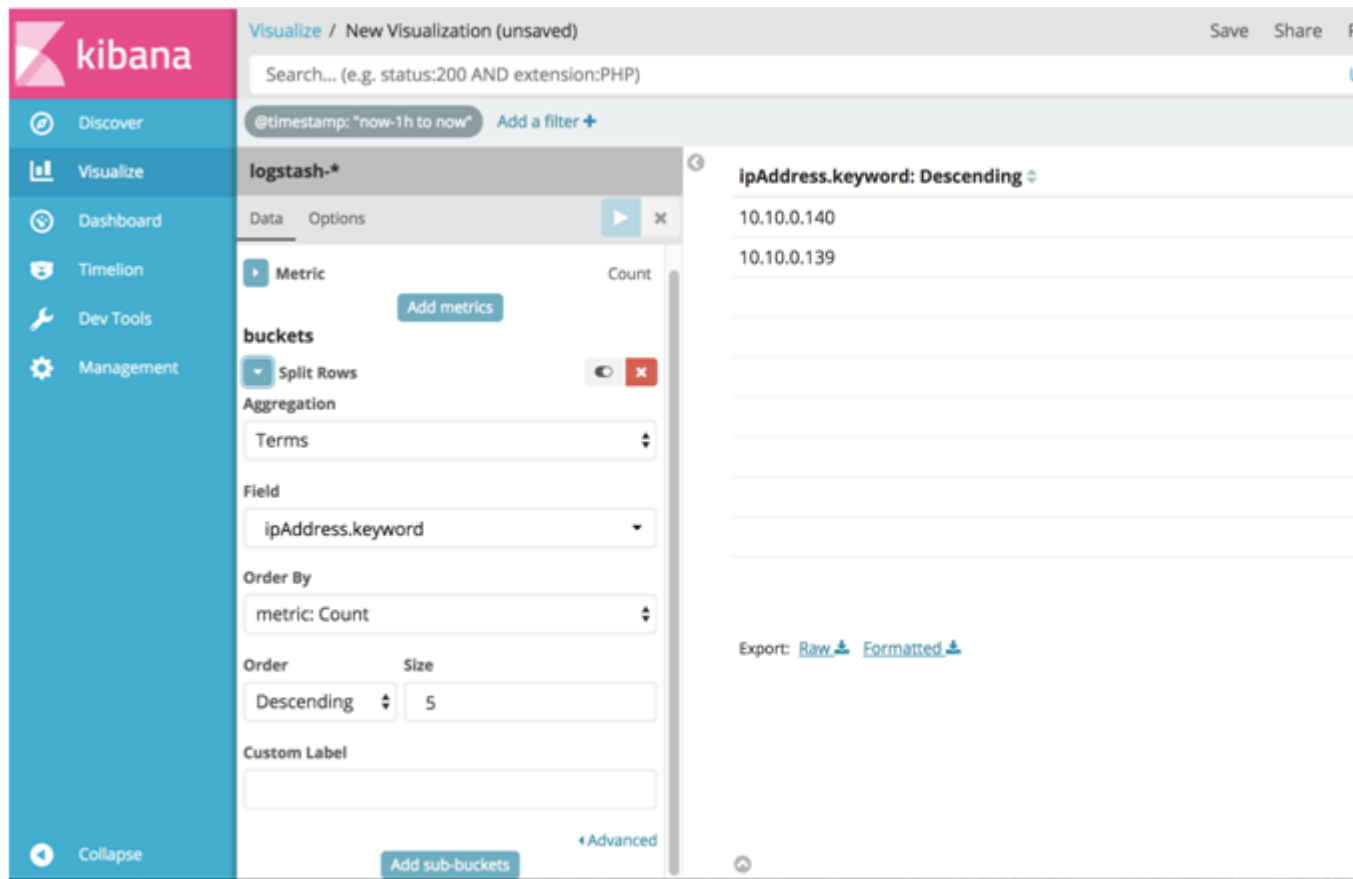
- Next, in the “buckets” section, click **Split Rows**, select “Terms” from the “Aggregation” drop-down, select “contentAccessoruserID.keyword” from the “Field” drop-down, select “metric: Count” from the “Order By” drop-down, select “Descending” from the “Order” drop-down, and finally set “Size” to “5”. Click the play button at the top of the options and you should see a screen like this:



5. Click **Save** in the top right of the page to save the visualization.

#### 10.4.3.2. Top 5 client IPs in # of requests the last hour

1. Similarly to the last metric, go to "Visualize" in the sidebar, click **Create a visualization**, select "Data Table", and then select the "logstash-\*" index. Again confirm the time setting is "Last 1 hour", and again create the "@timestamp" filter as explained in the previous section.
2. Next, in the "buckets" section, click "Split Rows", select "Terms" from the "Aggregation" drop-down, select "ipAddress.keyword" from the "Field" drop-down, select "metric: Count" from the "Order By" drop-down, select "Descending" from the "Order" drop-down, and finally set "Size" to 5. Click the play button at the top of the options and you should see a screen like this:



3. Click **Save** in the top right of the page to save the visualization.

#### 10.4.3.3. Top 5 PUT object sizes in the last hour

1. Similarly to the last metric, go to “Visualize” in the sidebar, click **Create a visualization**, select “Data Table”, and then select the “logstash-\*” index. Again confirm the time setting is “Last 1 hour”, and again create the “@timestamp” filter as explained in the previous section.
2. We also have to add another filter for this visualization. Click **Add a filter** under the search bar, select the “operation.keyword” field, select the “is one of” operator, enter “putObject” and “uploadPart” in “Values” and then click Save:

**Edit filter** ✕

**Filter** Edit Query DSL

operation.keyword ▼ is one of ▼ putObject ✕ uploadPart ✕

**Label**

Optional

✕ Cancel Save

- Next, in the “buckets” section, click “Split Rows”, select “Terms” from the “Aggregation” drop-down, select “requestBodySize” from the “Field” drop-down, select “Custom Metric” from the “Order By” drop-down, select “Max” from the new “Aggregation” drop-down, select “requestBodySize” from the new “Field” drop-down, select “Descending” from the “Order” drop-down, and finally set “Size” to 5. Click the play button at the top of the options and you should see a screen like this:

**kibana** Visualize / New Visualization (unsaved) Save Share

Search... (e.g. status:200 AND extension:PHP)

@timestamp: "now-1h to now" operation.keyword: "putObject, uploadPart" Add a filter +

**logstash-\***

Data Options ▶ ✕

**buckets**

☒ Split Rows ✕

Aggregation

Terms

Field

requestBodySize

Order By

Custom Metric

Aggregation

Max

Field

requestBodySize Advanced

Order Descending ▼ Size 5

**requestBodySize: Descending**

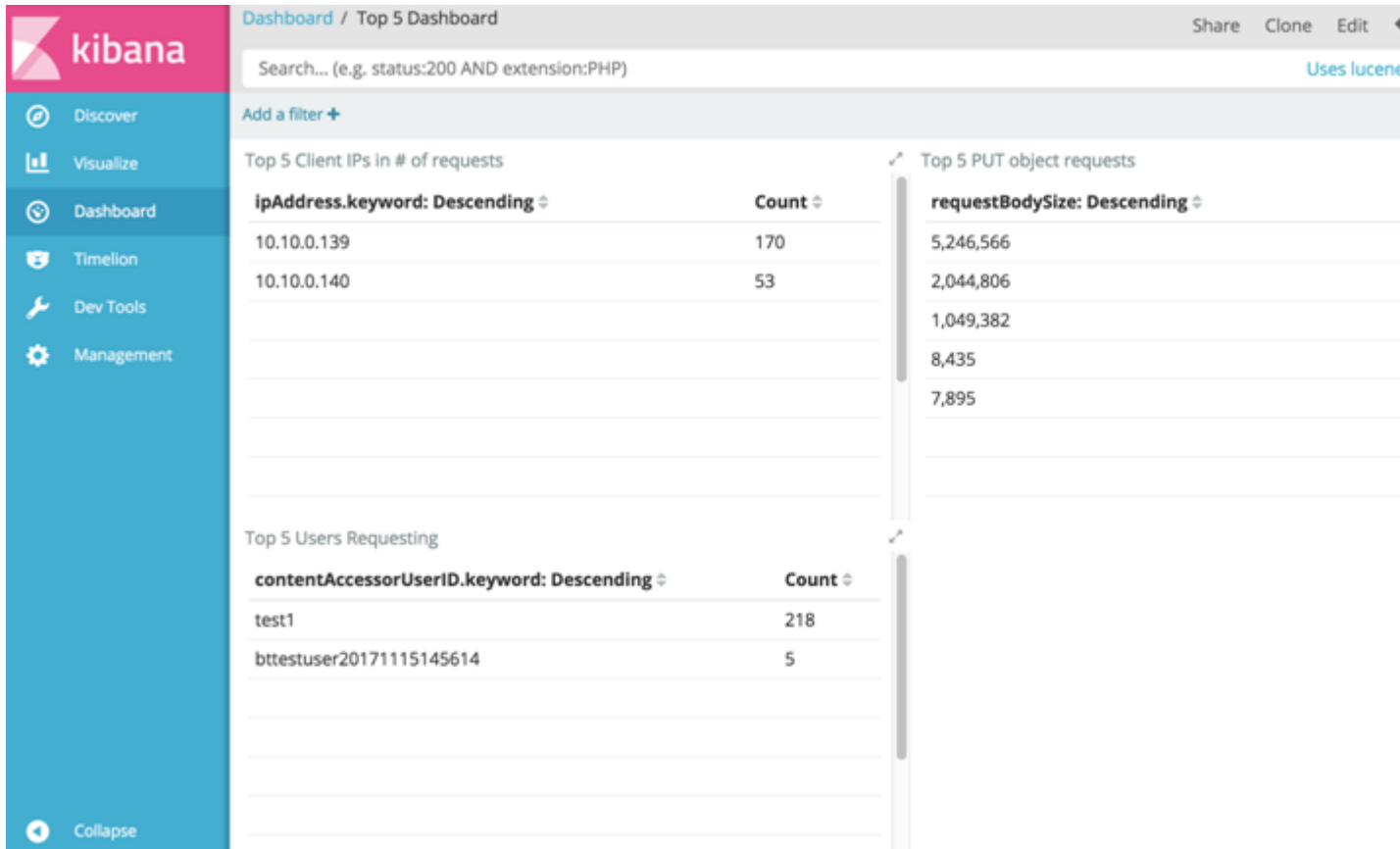
5,246,566
2,044,806
1,049,382
8,435
7,895

Export: [Raw](#) [Formatted](#)

- Click **Save** in the top right of the page to save the visualization.

### 10.4.3.4. Dashboard

To view all of your saved visualizations in one place go to “Dashboard” and click **Create a Dashboard**. Click the **Add** button at the top of the page, then click on each visualization you just created to add it to the dashboard. Click **Save** at the top of the page. You should now see a screen like this:



## 10.5. 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 Puppet master node (via SSH) 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 name>
```

**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
```

In the background this invokes the Linux command `/less` to display the log file. Therefore you can use the standard keystrokes supported by `/less` to navigate the display; for example:

- `f` key or Space bar -- Page down
- `b` key -- Page up
- Down arrow key or Enter -- Go down one line
- Up arrow key -- Go up one line
- `<n>f` key or `<n>Space bar` -- Go down `<n>` number of lines
- `<n>b` key -- Go up `<n>` number of lines
- `/string` Enter-- Search down for the specified string
- `?string` Enter -- Search up for the specified string
- `q` key -- Quit the file display and return to the HSH prompt

For details about the logs you can view, see **"HyperStore Logs"** (page 605).

This page left intentionally blank

# Chapter 11. Commands

## 11.1. `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 directly 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. For detail see the descriptions of the individual commands.
- 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** All *hsstool* commands can be executed either on the command line or through the CMC's [Node Advanced](#) page. The documentation in this section shows the CMC interface for each command as well as the command line syntax.

Command Type	Command	Purpose
Information Commands	<a href="#">ring</a>	View vNode info for whole cluster
	<a href="#">info</a>	View vNode and data load info for a physical node
	<a href="#">status</a>	View summary status for whole cluster
	<a href="#">opstatus</a>	View status of operations such as cleanup, repair, and rebalance
	<a href="#">proactiverepairq</a>	View status of proactive repair queues
	<a href="#">repairqueue</a>	View auto-repair schedule or enable/disable auto-repair
	<a href="#">ls</a>	View vNode and data load info per mount point
	<a href="#">whereis</a>	View storage location information for an S3 object
	<a href="#">metadata</a>	View metadata for an S3 object
	<a href="#">trmap</a>	View token range snapshot IDs or snapshot content
	madd	This is for internal use by the install script, or for use as directed by Clodian Support.
Maintenance Commands	<a href="#">cleanup</a>	Clean a node of replicated data that doesn't belong to it
	<a href="#">cleanupec</a>	Clean a node of erasure coded data that doesn't belong to it
	autorepair	In the CMC interface this invokes the <a href="#">repairqueue</a> command
	<a href="#">repair</a>	Repair the replicated data on a node
	<a href="#">repairec</a>	Repair the erasure coded data in a data center
	<a href="#">repaircassandra</a>	Repair only the Cassandra metadata on a node, not the object data
	<a href="#">rebalance</a>	Shift data load from your existing nodes to a newly added node
	<a href="#">opctl</a>	List or stop all repair and cleanup operations currently running in the region

### 11.1.1. *hsstool* cleanup

Subjects covered in this section:

- *Introduction (immediately below)*
- **"Command Syntax"** (page 645)
- **"Command/Response Example"** (page 646)
- **"When to Use *hsstool* cleanup"** (page 647)

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.

**Note** By default you can only run *hsstool cleanup* on one node at a time per data center. This limit is configurable by the "**max.cleanup.operations.perdc**" (page 550) setting in [hyperstore-server.properties.erb](#). If you want to raise this limit, consult with Clodian Support first.

**Note** 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 whose Last Modified timestamp is older than the interval set by the system configuration property *hyperstore-server.properties: cleanup.session.delete.graceperiod*. 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*.

#### 11.1.1.1. Command Syntax

The *hsstool cleanup* command line syntax is as follows. For description of a particular parameter, click on the parameter; or for the full list of parameter descriptions see "**hsstool cleanup Parameters**" (page 647).

```
# hsstool -h <host> cleanup [allkeyspaces|nokeyspaces] [-n] [-l <true|false>]
[-b] [-x] [-no] [-a] [-c <true|false>] [-d <mountpoint>] [-vnode <token>] [-policy]
[-stop]
```

You can also run the *hsstool cleanup* 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 '1'), 'Alerts', 'Admin', and 'Help'. Below this, the 'Nodes' sub-tab is selected (highlighted with a red box and a red '2'). The 'Advanced' tab is selected in the sub-navigation bar (highlighted with a red box and a red '3'). The 'Command Type' dropdown is set to 'Maintenance' (circled in red). The 'hstool Command' dropdown is set to 'cleanup' (circled in red). The 'Target Node' dropdown is set to 'jarvis'. The 'Options' section shows 'default' selected for 'allkeyspaces' and 'nokeyspaces', and 'n', 'l', 'b', 'x', 'a', 'c', and 'stop' are checked. The 'Description' is 'Cleanup HyperStore node data'. An 'EXECUTE' button is 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 [hstool opstatus](#) command.

### 11.1.1.2. Command/Response Example

The example below shows a default run of *cleanup*, using no options. For description of a particular response item, click on the response item; or for the full list of response item descriptions see **"hstool cleanup and hstool opstatus cleanup Response Items "** (page 650).

```
# hstool -h cloudian-node1 cleanup
Executing cleanup. keyspaces=UserData deletedata=true logging=true deleteobject-without-
bucketinfo=false
check-protection=true deleteobject-without-bucketinfo=false
optype: CLEANUP cmdno#: 1 status: COMPLETED
arguments: deleteobject-without-bucketinfo=false check-protection=true deletedata=true
deleteobject-without-policy=false keyspaces=UserData logging=true delete-only-outofrange-
objects=false
no-delete-out-of-range=false
start: Thu May 18 05:13:50 PDT 2017
end: Thu May 18 05:13:50 PDT 2017
duration: 0.145 sec
progress percentage: 100%
cassandra cleanup time: 0.015 sec
task count: 2
completed count: 2
Number of files deleted count: 0
failed count: 0
skipped count: 2
```

**Note** In the example above there is very little data in the system and so the operation completes almost instantly. In a real-world environment this is a long-running operation and the command response will not return until the operation completes. In the meanwhile you can track operation progress as described in the Note in the [Command Format](#) section above.

**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" rather than "Number of files deleted count".

### 11.1.1.3. When to Use *hsstool cleanup*

The operational procedure during which you would use *hsstool cleanup* are:

- "Restoring a Node That Has Been Offline" (page 453)
- "Delete a Storage Policy" (page 379)

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 420). 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 Parameters"** (page 682).

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.

### 11.1.1.4. *hsstool cleanup* Parameters

`-h <host>`

(Mandatory) Hostname or IP address of the node to clean.

**Note** In the CMC UI for this command this parameter is called "Target Node".

*allkeyspaces* | *nokeyspaces*

(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*

(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>*

(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 *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 *cleanup* command run is marked with a unique command number.

*-b*

(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*

(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*

(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*

(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>*

(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 in 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>*

(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>*

(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*

(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 hourly run of the [object deletion cron job](#). 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 deletion cron job. 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*

(Optional) Use *hsstool -h <host> cleanup -stop* to terminate an in-progress cleanup operation on the specified node.

You can subsequently use the **"hsstool opstatus"** (page 666) 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. Instead, when you want to clean the node you can run a regular full cleanup operation.

#### 11.1.1.5. *hsstool cleanup* and *hsstool opstatus cleanup* Response Items

*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*.

*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.

*start*

Start time of the operation.

*end, duration*

End time and duration of a completed operation.

*estimated completion time, time remaining*

Estimated completion time and estimated time remaining for an in-progress 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.

*cassandra cleanup time*

The time spent cleaning metadata in Cassandra.

*task count*

The number of object replicas on the node, which the cleanup operation must evaluate to determine whether they correctly belong on the node.

*completed count*

The number of object replicas that the cleanup operation evaluated to determine whether they correctly belong on the node.

*Number of files deleted count*

The number of object replicas that the cleanup 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 that the cleanup operation tried to delete (because they don't belong on the node), but failed.

*skipped count*

The number of object replicas that the cleanup operation left on the node because they belong on the node.

## 11.1.2. hsstool cleanupec

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Command Syntax"** (page 652)

- **"Command/Response Example"** (page 653)
- **"When to Use *hsstool cleanupec*"** (page 654)

**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.

**Note** By default you can only run *hsstool cleanupec* on one node at a time per data center. This limit is configurable by the **"max.cleanup.operations.perdc"** (page 550) setting in [hyperstore-server.properties.erb](#). If you want to raise this limit, consult with Cloudbian Support first.

**Note** The system will not allow you to run *hsstool cleanupec* on a node on which [hsstool repairec](#) is currently running.

**Note** The *hsstool cleanupec* operation will only clean objects whose Last Modified timestamp is older than the interval set by the system configuration property *hyperstore-server.properties: cleanup.session.delete.graceperiod*. 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*.

### 11.1.2.1. Command Syntax

The *hsstool cleanupec* command line syntax is as follows. For description of a particular parameter, click on the parameter; or for the full list of parameter descriptions see **"hsstool cleanupec Parameters"** (page 654).

```
# hsstool -h <host> cleanupec \[-n\] \[-l <true|false>\] \[-b\] \[-x\] \[-no\]
\[-c <true|false>\] \[-d <mountpoint>\] \[-vnode <token>\] \[-policy\] \[-stop\]
```

You can also run the *hsstool cleanupec* 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.

### 11.1.2.2. Command/Response Example

The example below shows a default run of *cleanupec*, using no options. For description of a particular response item, click on the response item; or for the full list of response item descriptions see "**hsstool cleanupec and hsstool opstatus cleanupec Response Items**" (page 656).

```
# hsstool -h cloudian-nodel cleanupec
Executing cleanupec. deleteobject-without-bucketinfo=false check-protection=true deletedata=true
logging=true
delete-only-outofrange-objects=false
optype: CLEANUPEC cmdno#: 1 status: COMPLETED
arguments: deleteobject-without-bucketinfo=false check-protection=true deletedata=true logging=true
delete-only-outofrange-objects=false no-delete-out-of-range=false
start: Thu May 18 05:14:18 PDT 2017
end: Thu May 18 05:14:18 PDT 2017
duration: 0.011 sec
progress percentage: 100%
task count: 0
completed count: 0
Number of files deleted count: 0
failed count: 0
skipped count: 0
```

**Note** In the example above there is very little data in the system and so the operation completes almost instantly. In a real-world environment this is a long-running operation and the command

response will not return until the operation completes. In the meanwhile you can track operation progress as described in the Note in the [Command Format](#) section above.

**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" rather than "Number of files deleted count".

### 11.1.2.3. When to Use `hsstool cleanupec`

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 453)
- **"Delete a Storage Policy"** (page 379)

Please refer to those procedures for step-by-step instructions, including the proper use of `hsstool cleanupec` within the context of the procedure.

**Note** It's OK for `hsstool cleanupec` to be running on multiple nodes in parallel. To do so, you need to initiate the cleanups one node at a time, but you don't need to wait for `hsstool cleanupec` to complete on one node before starting it on another node.

**Note** The `hsstool cleanupec` operation will only clean objects whose Last Modified timestamp is older than the interval set by the system configuration property `hyperstore-server.properties: cleanup.session.delete.graceperiod`. 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`.

### 11.1.2.4. `hsstool cleanupec` Parameters

`-h <host>`

(Mandatory) Hostname or IP address of the node to clean.

**Note** In the CMC UI for this command this parameter is called "Target Node".

`-n`

(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>`

(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 Clouidian 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`

(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`

(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`

(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>`

(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>`

(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>`

(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`

(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 [object deletion cron job](#). 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 deletion cron job. 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`

(Optional) Use `hsstool -h <host> cleanupec -stop` to terminate an in-progress *cleanupec* operation on the specified node.

You can subsequently use the **"hsstool opstatus"** (page 666) 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. Instead, when you want to clean the node you can run a regular full *cleanupec* operation.

#### 11.1.2.5. *hsstool cleanupec* and *hsstool opstatus cleanupec* Response Items

*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-hyper-store.log* for error messages from the period during which the operation was running.

A TERMINATED status means that the clean up run was terminated by an operator, using *cleanupec -stop*.

*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.

*start*

Start time of the operation.

*end, duration*

End time and duration of a completed operation.

*estimated completion time, time remaining*

For an INPROGRESS operation: the estimated completion time of the operation, and estimated time remaining to complete 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.

*cassandra cleanup time*

The time spent cleaning metadata in Cassandra.

*task count*

The number of erasure coded fragments on the node, which the cleanup operation must evaluate to determine whether they correctly belong on the node.

*completed count*

The number of erasure coded fragments that the cleanup operation evaluated to determine whether they correctly belong on the node.

*Number of files deleted count*

The number of erasure coded fragments that the cleanup 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 `cleanupec` 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 erasure coded fragments that the cleanup operation tried to delete (because they don't belong on the node), but failed.

#### *skipped count*

The number of erasure coded fragments that the cleanup operation left on the node because they belong on the node.

### 11.1.3. `hsstool info`

Subjects covered in this section:

- Introduction (immediately below)
- **"Command Syntax"** (page 658)
- **"Command/Response Example"** (page 659)

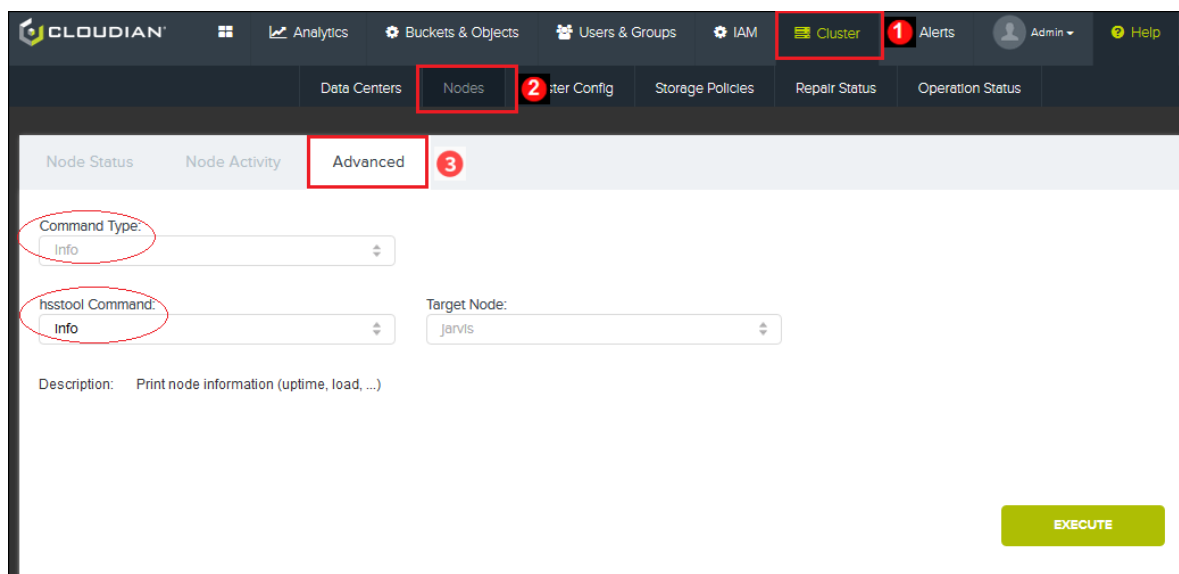
This `hsstool` command returns [virtual node](#) (token range) information and data load information for a specified physical node within a storage cluster. The return includes a list of virtual nodes (tokens) assigned to the physical node.

#### 11.1.3.1. Command Syntax

The `hsstool info` command line syntax is as follows.

```
# hsstool [-h <host>] info
```

You can also run the `hsstool info` command through the CMC UI:



### 11.1.3.2. 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. For description of a particular response item, click on the response item; or for the full list of response item descriptions see "**hsstool info Response Items**" (page 659).

```
# hsstool -h cloudian-node1 info
Cloudian : 6.1
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](#).

### 11.1.3.3. *hsstool info* Parameters

*-h <host>*

(Optional) Hostname or IP address of the node for which to retrieve token and load information. If not supplied, this defaults to the hostname of the host on which you are executing *hsstool*.

**Note** In the CMC UI for this command this parameter is called "Target Node".

### 11.1.3.4. *hsstool info* 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 42).

#### *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.

### 11.1.4. `hsstool ls`

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Command Syntax"** (page 660)
- **"Command/Response Example "** (page 661)

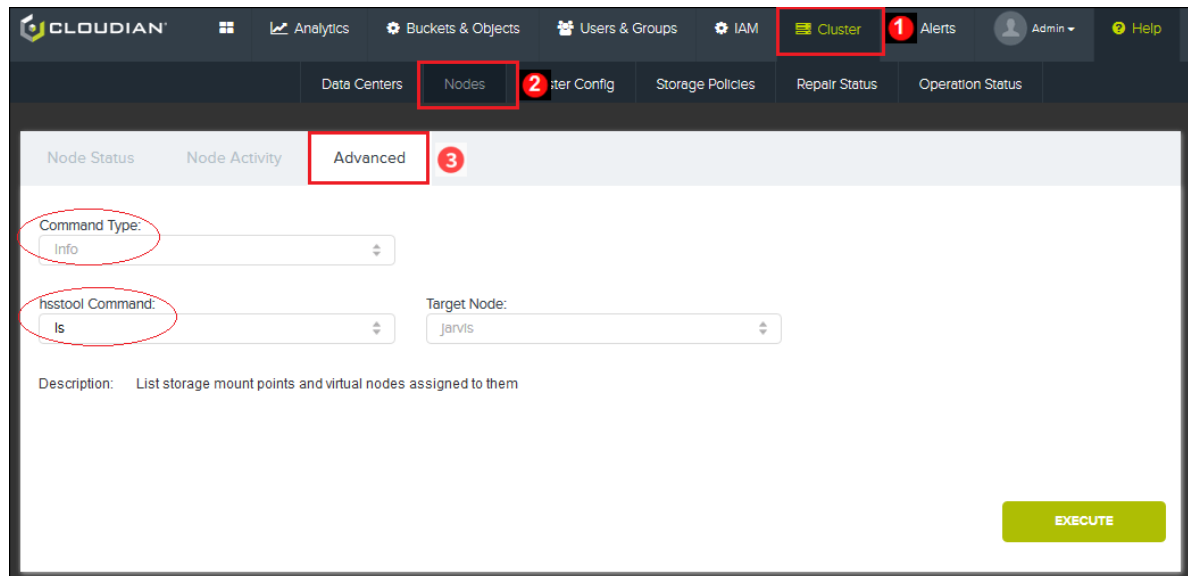
This [hsstool](#) command returns a node's list of HyperStore data mount points, the list of storage tokens currently assigned to each mount point on the node, and the current disk usage per mount point.

#### 11.1.4.1. Command Syntax

The `hsstool ls` command line syntax is as follows.

```
# hsstool [-h <host>] ls
```

You can also run the `hsstool ls` command through the CMC UI:



### 11.1.4.2. 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. For description of a particular response item, click on the response item; or for the full list of response item descriptions see "**hsstool ls Response Items**" (page 662).

```
# hsstool -h cloudian-node1 ls
```

device	mount-point	total (KB)	available (KB)	status
/dev/sdf1	/cassandra	226955412	225396540	OK
/dev/sda1	/cloudian1	3784528904	2992343200	OK
/dev/sdb1	/cloudian2	3784528904	2864827364	OK
/dev/sdc1	/cloudian3	3784528904	2932836832	OK
/dev/sdd1	/cloudian4	3784528904	3244435388	OK

```
/cloudian1/hsfs:
119513460404589000549564154532759249912 2jf00jEMI1ffVOT7No4LU0
166983878496718254895534451073709499214 3p37rkVSXWvrnGkGrwNh5a
14669198764159070178516665850483502746 Kp70oWCXwTDVyCuWhy18U
...
```

### 11.1.4.3. *hsstool ls* Parameters

*-h <host>*

(Optional) Hostname or IP address of the node for which to retrieve token and load information. If not supplied, this defaults to the hostname of the host on which you are executing *hsstool*.

**Note** In the CMC UI for this command this parameter is called "Target Node".

#### 11.1.4.4. *hsstool ls* Response Items

##### *device*

Device name of the disk drive

##### *mount-point*

Mount point of the device

##### *total (KB)*

Total capacity of the disk, in KBs

##### *available (KB)*

Remaining available capacity of the disk, in KBs

##### *status*

Disk status: either OK or ERROR or DISABLED. For more information on the disk's status see the CMC's **Node Status** page, [Disk Detail Info](#) section.

##### *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 23).

#### 11.1.5. *hsstool* metadata

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Command Syntax"** (page 662)
- **"Command/Response Example"** (page 663)

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.

##### 11.1.5.1. Command Syntax

The *hsstool metadata* command line syntax is as follows. For description of a particular parameter, click on the parameter; or for the full list of parameter descriptions see **"hsstool metadata Parameters"** (page 664).

```
# hsstool \[-h <host>\] metadata <bucket>/<object> \[-v <version>\]
```

You can also run the *hsstool metadata* command through the CMC UI:

CLLOUDIAN

Analytics Buckets & Objects Users & Groups IAM Cluster Alerts Admin Help

Data Centers Nodes Cluster Config Storage Policies Repair Status Operation Status

Node Status Node Activity **Advanced**

Command Type: Info

hsstool Command: metadata Target Node: jarvis

Bucket: Object: Version(optional)

Description: Display metadata information of a given object

EXECUTE

### 11.1.5.2. Command/Response Example

The *metadata* command example below returns the metadata for the specified object. For description of a particular response item, click on the response item; or for the full list of response item descriptions see **"hsstool metadata Response Items"** (page 664).

```
# hsstool -h cloudian-node1 metadata hsfsbn/so
Key: hsfsbn/so
Policy ID: 43c3e277945403c98e5b8f9441d85e16
Version: null
Compression: NONE
Create Time: 2020-07-02T06:16:54.681Z
Last Modified: 2020-07-02T06:16:54.681Z
Last Access Time: 2020-07-02T06:16:54.681Z
Digest: 7bcec86b667ff2be2982f637b67e4942
Size: 1048576
Region: region1

CLOUDIAN_METADATA metadata: {"Path": "hsfsbn/so", "Type": "FILE", "Etag":
"7bcec86b667ff2be2982f637b67e4942", "Locale": null, "GroupId": "CloudianTest1", "UserId":
"77600d5e7dfb7e0f46a198fbded86fe3", "CreatorId": "77600d5e7dfb7e0f46a198fbded86fe3",
"Ttl": null, "CreateTime": "2020-07-02T06:16:54.681Z", "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": "2020-07-02T06:16:54.681Z", "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": "{îÈkfu007Fò%)\u0082ö7¿~IB}}", "AccumulateSize":
1048576, "LockInfo": null}
```

### 11.1.5.3. *hsstool metadata* Parameters

*-h <host>*

(Optional) Hostname or IP address of the node for which to retrieve token and load information. If not supplied, this defaults to the hostname of the host on which you are executing *hsstool*.

**Note** In the CMC UI for this command this parameter is called "Target Node".

*<bucket>/<object>*

(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\_2016-06-27.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\_2016-06-27.mpg*.

*-v <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.

### 11.1.5.4. *hsstool metadata* Response Items

*Key*

Key that uniquely identifies the S3 object, in format *<bucketname>/<objectname>*. For example, *bucket1/Documents/Meetings\_2016-06-27.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*

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. An object's 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 Clouidian 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.

### 11.1.6. hsstool opctl

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Command Syntax"** (page 666)
- **"Command/Response Examples "** (page 666)

This [hsstool](#) command returns a list of all [hsstool repair](#), [hsstool repairec](#), [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.

### 11.1.6.1. Command Syntax

The *hsstool opctl* command line syntax is as follows. For description of a particular parameter, click on the parameter; or for the full list of parameter descriptions see "**hsstool opctl Parameters**" (page 666).

```
# hsstool -h <host> opctl [-l] [-stop]
```

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.

### 11.1.6.2. 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,merkletree=true,
computedigest=false

# hsstool -h cloudian-node1 opctl -stop
Aborted REPAIR on 10.10.0.184
```

### 11.1.6.3. *hsstool opctl* Parameters

*-h <host>*

(Mandatory) Hostname or IP address of the node to which to submit the command. This can be any node in the service region. The command will apply to all nodes in the service region.

*-l*

(Optional) List all in-progress *repair*, *repairec*, *cleanup*, and *cleanupec* operations in the service region.

*-stop*

(Optional) Terminate all in-progress *repair*, *repairec*, *cleanup*, and *cleanupec* operations in the service region.

### 11.1.7. *hsstool opstatus*

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Command Syntax"** (page 667)

- **"Command/Response Examples"** (page 668)

This [hsstool](#) command returns the status of the most recent runs of repair, cleanup, rebalance, or decommission 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.

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. This page does not report on operations that you've launched on the command line -- for such operations *hsstool opstatus* is your only option for checking status.

### 11.1.7.1. Command Syntax

The *hsstool opstatus* command line syntax is as follows. For description of a particular parameter, click on the parameter; or for the full list of parameter descriptions see **"hsstool opstatus Parameters"** (page 671).

```
# hsstool [-h <host>] opstatus [<op-type>] [-a] [-q history]
```

You can also run the *hsstool opstatus* command through the CMC UI:

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'). In the 'Advanced' section, the 'Command Type' dropdown is set to 'Info' (circled in red), and the 'hsstool Command' dropdown is set to 'opstatus' (circled in red). The 'Target Node' dropdown is set to 'jarvis'. A description below reads: 'Display status of operations in progress'. An 'EXECUTE' button is located at the bottom right of the form.

### 11.1.7.2. Command/Response Examples

#### *opstatus for cleanup and cleanupec*

For description of a particular response item, click on the response item; or for the full list of response item descriptions see **"hsstool cleanup and hsstool opstatus cleanup Response Items"** (page 650) and **"hsstool cleanupec and hsstool opstatus cleanupec Response Items"** (page 656).

```
# hsstool -h cloudian-node1 opstatus cleanup
```

```
optype: CLEANUP cmdno#: 1 status: COMPLETED
arguments: deleteobject-without-bucketinfo=false check-protection=true deletedata=true
deleteobject-without-policy=false keyspaces=UserData logging=true delete-only-outofrange-
objects=false
start: Thu May 18 05:13:50 PDT 2017
end: Thu May 18 05:13:50 PDT 2017
duration: 0.145 sec
progress percentage: 100%
cassandra cleanup time: 0.015 sec
task count: 2
completed count: 2
deleted count: 0
failed count: 0
skipped count: 2
```

```
# hsstool -h cloudian-node1 opstatus cleanupec
```

```
optype: CLEANUPEC cmdno#: 1 status: COMPLETED
arguments: deleteobject-without-bucketinfo=false check-protection=true deletedata=true logging=true
delete-only-outofrange-objects=false
start: Thu May 18 05:14:18 PDT 2017
end: Thu May 18 05:14:18 PDT 2017
duration: 0.011 sec
progress percentage: 100%
task count: 0
completed count: 0
deleted count: 0
failed count: 0
skipped count: 0
```

**Note** For in-progress operations, rather than an end time and duration the *opstatus* response will show an estimated completion time and estimated time remaining.

#### *opstatus for repair and repairec*

For description of a particular response item, click on the response item; or for the full list of response item descriptions see **"hsstool repair and hsstool opstatus repair Response Items"** (page 692) and **"hsstool repairec and hsstool opstatus repairec Response Items"** (page 704).

```
# hsstool -h cloudian-node1 opstatus repair
```

```
optype: REPAIR cmdno#: 1 status: COMPLETED
arguments: keyspaces=UserData max-modified-ts=1495109681947 primary-range=false min-modified-ts=0
```

```

logging=true check-metadata=true merkle-tree=true computed-digest=false
operation ID: ddec91f8-0b56-1149-bd85-5254005d772d
start: Fri Dec 08 05:14:42 PDT 2017
end: Fri Dec 08 05:14:50 PDT 2017
duration: 8.107 sec
progress percentage: 100%
total range count: 3
executed range count: 3
keyspace count: 1
repair file count: 0
failed count: 0
repaired count: 0
pr queued count: 0
completed count: 3
total bytes streamed: 0
scan time: 0.3685
stream time: 0.0

# hsstool -h cloudian-nodel opstatus repairedc

optype: REPAIREDC cmdno#: 1 status: COMPLETED
arguments: rebuild=false mountPoint=null logging=true range=null cmdno=1 computed-digest=false
operation ID: 06186b78-1bec-1be0-a6a5-026a20c18bd8
start: Thu Sep 05 06:58:32 UTC 2019
end: Thu Sep 05 06:58:34 UTC 2019
duration: 2.17 sec
progress percentage: 100%
time remaining: 0 ms
total ranges: 0
task count: 9
completed count: 9
repaired count: 9
failed count: 0
skipped count: 0
rcvd connection count: 0
open connection count: 0
message threadpool active: 0
timer:  cassandra.iterating.timer: count=10 mean=28916.792488333995;
        repairedc.task.timer: count=0 mean=NaN; digest.scan.per.Ep.timer: count=0 mean=NaN;
        distributor.batch.execution.timer: count=1 mean=1.684142835E9;
        unit.of.scan.task.timer: count=0 mean=NaN; session.sleep.timer: count=0 mean=NaN;
        RocksDB.digests.per.disk.timer: count=0 mean=NaN;

```

**Note** For in-progress operations, rather than an end time and duration the *opstatus* response will show an estimated completion time and estimated time remaining.

**Note** "REPAIR\_CASSANDRA" status metrics will appear in *opstatus* results for the Cassandra key-space repair part of the auto-repair feature; or for when you manually run *hsstool repair* either with its default behavior (which includes a repair of user data keyspaces in Cassandra) or with the "allkey-spaces" option (which includes a repair of user data keyspaces and service metadata keyspaces in Cassandra).

*opstatus for rebalance and rebalanceec*

For description of a particular response item, click on the response item; or for the full list of response item descriptions see "**hsstool rebalance and hsstool opstatus rebalance/rebalanceec Response Items**" (page 683).

```
# hsstool -h cloudian-node1 opstatus rebalance

optype: REBALANCE cmdno#: 1 status: COMPLETED
arguments: logging=true
operation ID: 2edcb684-94f0-198d-91da-5254007cc5f2
start: Thu May 04 16:51:03 CST 2017
end: Thu May 04 16:51:04 CST 2017
duration: 0.324 sec
progress percentage: 100%
task count: 1
completed count: 1
streamed count: 1
failed count: 0
skipped count: 0
stream jobs total: 5
stream jobs completed: 5
streamed bytes: 500

# hsstool -h cloudian-node1 opstatus rebalanceec

optype: REBALANCEEC cmdno#: 1 status: COMPLETED
arguments: logging=true
operation ID: 2edcb684-94f0-198d-91da-5254007cc5f2
start: Thu May 04 16:51:04 CST 2017
end: Thu May 04 16:51:04 CST 2017
duration: 0.248 sec
progress percentage: 100%
task count: 3
completed count: 3
streamed count: 3
failed count: 0
skipped count: 0
stream jobs total: 6
stream jobs completed: 6
streamed bytes: 1024
```

**Note** For in-progress operations, rather than an end time and duration the *opstatus* response will show an estimated completion time and estimated time remaining.

*opstatus for decommissionreplicas and decommissioneec*

**Note** The *decommission* operation is automatically invoked by the CMC's "Uninstall" feature, if you use the Uninstall feature to remove a node that's "live" in the Cassandra ring. Status reporting on a decommission operation is broken out to *decommissionreplicas* (for replicated data) and *decommisnonec* (for erasure coded data).

For description of a particular response item, click on the response item; or for the full list of response item descriptions see **"Response Items for hsstool opstatus decommissionreplicas or decommissionec"** (page 673).

```
# hsstool -h cloudian-node1 opstatus decommissionreplicas

optype: DECOMMISSIONREPLICAS cmdno#: 1 status: INPROGRESS
operation ID: d697d9c4-2fc5-124b-ab0c-525400137a9c
start: Tue May 23 20:31:24 PDT 2017
cassandra decommission time: 61.072 sec
task count: 2410
completed count: 2139
streamed count: 2139
failed count: 0
skipped count: 0
stream jobs total: 40
stream jobs completed: 31
streamed bytes: 4403497430
```

```
# hsstool -h cloudian-node1 opstatus decommissionec

optype: DECOMMISSIONEC cmdno#: 1 status: INPROGRESS
operation ID: d697d9c4-2fc5-124b-ab0c-525400137a9c
start: Tue May 23 20:32:38 PDT 2017
task count: 757
completed count: 743
streamed count: 743
failed count: 0
skipped count: 0
stream jobs total: 2
stream jobs completed: 1
streamed bytes: 0
```

### 11.1.7.3. *hsstool opstatus* Parameters

*-h* <host>

(Optional) Hostname or IP address of the node for which to retrieve token and load information. If not supplied, this defaults to the hostname of the host on which you are executing *hsstool*.

**Note** In the CMC UI for this command this parameter is called "Target Node".

<op-type>

(Optional) Type of operation for which to retrieve status. Valid types are listed below. If you do not specify a type, status is returned for all supported operation types.

- cleanup
- cleanupec
- decommissionreplicas
- decommissionec

**Note** The *decommission* operation is automatically invoked by the CMC's "Uninstall" feature, if you use the Uninstall feature to remove 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).

- *proactiverebalance*
- *proactiverepair*
- *proactiverepairec*
- *rebalance*
- *rebalanceec*
- *repair*
- *repaircassandra*
- *repairec*

**Note** The CMC does not support the "<op-type>" option.

*-a*

(Optional) Verbose status output for a repair or rebalance operation. This provides repair or rebalance status details per token range. It also shows relevant configuration settings. This detailed data can be helpful if you are working with Clodian Support to troubleshoot a repair or rebalance problem. This option is supported only for the "repair", "rebalance", and "rebalanceec" operation types — not for the other types.

For example:

```
[root]# /opt/cloudian/bin/hsstool -h <host> opstatus repair -a
```

**Note** The CMC does not support the "-a" option.

*-q history*

(Optional) Use *hsstool -h <host> opstatus -q history* to retrieve a history of rebalance, repair, and cleanup operations performed on the target node. By default this history spans the past 90 days. This period is configurable by *mts.properties.erb*: "**monitoring.ophistory.ttl**" (page 568).

If you want just a history of a particular repair or cleanup operation type, use:

```
hsstool -h <host> opstatus -q history <op-type>
```

where <op-type> is *rebalance*, *rebalanceec*, *repair*, *repairec*, *cleanup*, or *cleanupec*. For example:

```
hsstool -h <host> opstatus -q history repairec
```

**Note** The CMC does not support the "-q history" option.

#### 11.1.7.4. Response Items for *hsstool opstatus decommissionreplicas* or *decommissionec*

You can check on the status of an in-progress decommission operation by using the **"hsstool opstatus"** (page 666) command.

##### *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, or FAILED

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 decommission operation successes and failures (if any), see the other fields in the response.

A FAILED status means that the operation ended prematurely due to errors. For additional status detail see the other fields in the 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.

**Note** Decommission processes replica data first and then erasure coded data. After decommission finishes for erasure coded data the HyperStore Service on the node immediately shuts down and can no longer response to *hsstool* commands including *hsstool opstatus*. Therefore the *decommissionec* operation will never show a status of COMPLETED (since the HyperStore Service shuts down upon *decommissionec* completion).

##### *operation ID*

Globally unique identifier of the *decommission* run. This may be useful if Cloudian Support is helping you troubleshoot a decommission failure. Note that when *decommission* is run, the DECOMMISSIONREPLICAS part of the response (for replica data) and DECOMMISSIONEC 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.

##### *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.

##### *cassandra decommission time*

The time spent decommissioning the node from the Cassandra ring. (Applicable to the *decommissionreplicas*

task only -- not *decommissionec*).

#### *task count*

The total number of files that are evaluated for possible streaming from the node to be decommissioned to other nodes in the cluster. 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 for which processing has been completed). Each completed task results in the incrementing of either the "streamed count", the "failed count", or the "skipped count".

#### *stream jobs total*

The system breaks the decommission streaming operation down into a number of small "jobs", with the number of jobs reflecting factors such as the particular storage policies in the cluster and the token ranges in which data associated with those storage policies is stored. This metric shows the total number of such jobs.

#### *stream jobs completed*

The number of stream jobs completed so far.

#### *streamed count*

The number of object replica files successfully streamed (copied) from the decommissioned node to other nodes in the cluster.

#### *streamed bytes*

The total number of object replica file bytes successfully streamed (copied) from the decommissioned node to other nodes in the cluster.

#### *failed count*

The number of files for which the attempt to stream the file to a different node failed as a result of an error. For information about such failures, on the decommissioned node you can scan `/var/log/cloudian/cloudian-hyperstore.log` for error messages from the time period during which the decommission operation was running.

#### *skipped count*

The number of files for which the streaming operation is skipped because a file that was going to be streamed from the decommissioned node to a different target node in the cluster is found to already exist on the target node.

The "streamed count" plus the "failed count" plus the "skipped count" will equal the "completed count".

**Note** For "decommission" operations, *opstatus* can only report in-progress status, not final, completed status. This is because the HyperStore service on the decommissioned node is immediately stopped at the completion of the *decommission* operation. For final decommission operation status you can review `/var/log/cloudian/cloudian-hyperstore.log` on the decommissioned node.

### 11.1.8. `hsstool proactiverepairq`

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Command Syntax"** (page 675)
- **"Command/Response Examples"** (page 676)

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 more information about the HyperStore proactive repair feature see **"Automated Data Repair Feature Overview"** (page 150).

### 11.1.8.1. Command Syntax

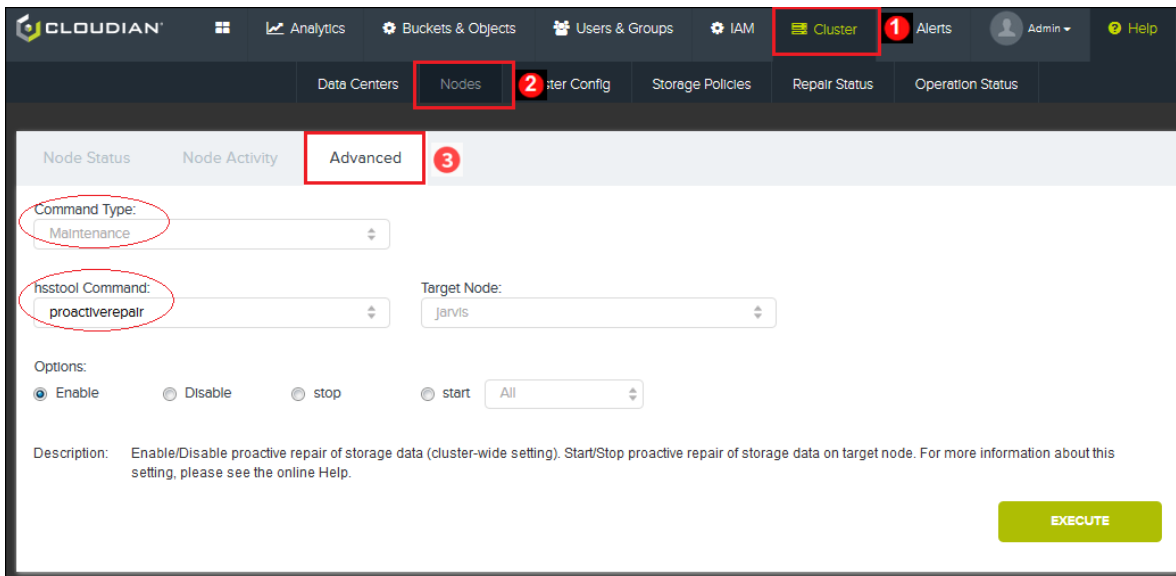
The *hsstool proactiverepairq* command line syntax is as follows. For description of a particular parameter, click on the parameter; or for the full list of parameter descriptions see **"hsstool proactiverepairq Parameters"** (page 676).

```
# hsstool -h <host> proactiverepairq [-a] [-delete <host>]
[-start [-type replicas|ec]] [-stop] [-enable true|false]
```

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. At the top, the 'Cluster' tab is selected. Below it, the 'Nodes' tab is active. In the 'Nodes' section, the 'Advanced' sub-tab is selected. The 'Command Type' dropdown is set to 'Info'. The 'hsstool Command' dropdown is set to 'proactiverepairqueue'. The 'Target Node' dropdown is set to 'jarvis'. Under 'Options', the checkbox for 'a' is checked. The 'Description' field reads 'Display proactive repair queues'. An 'EXECUTE' button is located at the bottom right of the form.

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):



### 11.1.8.2. 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). For description of a particular response item, click on the response item; or for the full list of response item descriptions see "**hsstool proactiverepairq Response Items**" (page 678).

```
# hsstool -h localhost 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). For description of a particular response item, click on the response item; or for the full list of response item descriptions see "**hsstool proactiverepairq Response Items**" (page 678).

```
# hsstool -h cloudian-node1 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.

### 11.1.8.3. *hsstool proactiverepairq* Parameters

*-h <host>*

(Mandatory) Hostname or IP address of the target node:

- 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*), this 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*), this is the host on which you want to start or stop a proactive repair.

**Note** In the CMC UI for this command this parameter is called "Target Node".

*-a*

(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 666) command on that node.

*-delete <host>*

(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 686) and "**hsstool repairec**" (page 697) 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 Cloudbian 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]*

(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 in the wake of 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-server.properties.erb*: "**hyperstore.proactiverepair.poll\_time**" (page 550)), 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*

(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*

(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 707) 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).

### 11.1.8.4. *hsstool proactiverepairq* 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.

## 11.1.9. hsstool rebalance

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Command Syntax"** (page 679)
- **"Command/Response Example"** (page 680)
- **"When to Use hsstool rebalance"** (page 681)

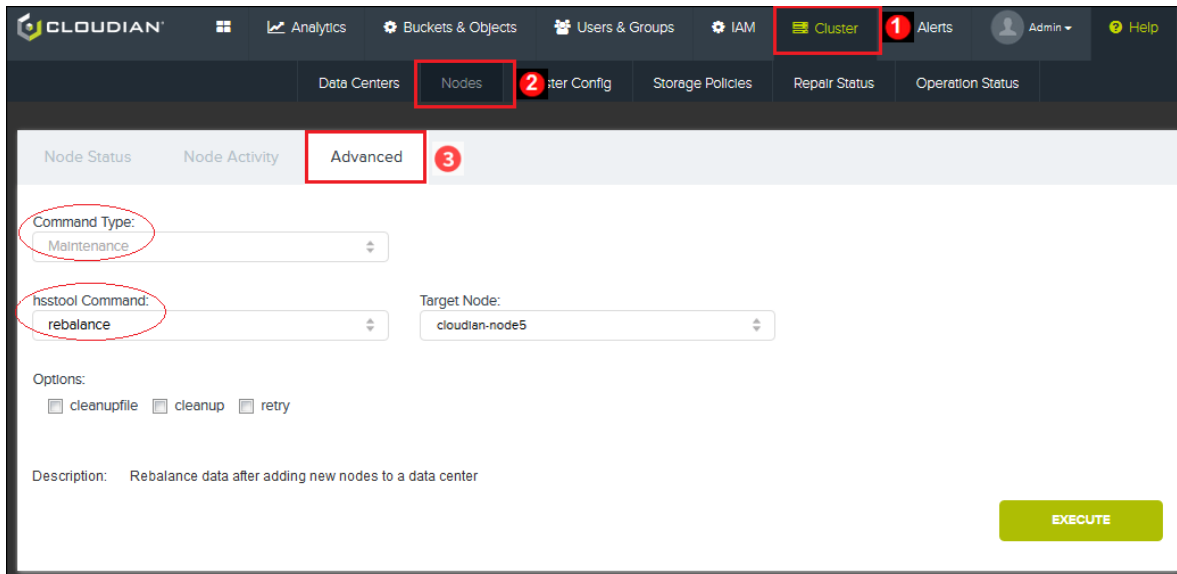
This [hsstool](#) command copies S3 object data from your existing nodes to a specified new node that you have added to your HyperStore cluster. The *rebalance* operation populates the new node with its share of S3 object replica data and erasure coded data, based on the token ranges that the system automatically assigned the new node when you added it to the cluster.

### 11.1.9.1. Command Syntax

The *hsstool rebalance* command line syntax is as follows. For description of a particular parameter, click on the parameter; or for the full list of parameter descriptions see **"hsstool rebalance Parameters"** (page 682).

```
# hsstool -h <host> rebalance [-list] [-cleanupfile|-cleanup] [-retry]
[-l <true|false>]
```

You can also run this 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.

### 11.1.9.2. Command/Response Example

The example below shows a *rebalance* operation being executed on a newly added node, and the command response. Note that rebalance is implemented separately for replica data ("REBALANCE cmdno #1" in the response) and erasure coded data ("REBALANCEEC cmdno #1"). For description of a particular response item, click on the response item; or for the full list of response item descriptions see "**hsstool rebalance and hsstool opstatus rebalance/rebalanceec Response Items**" (page 683).

**Note** For more detailed status information on a rebalance operation -- including a break-down of status by token range -- you can run `hsstool -h <host> opstatus rebalance -a` (or `opstatus rebalanceec -a`).

```
# hsstool -h cloudian-node6 rebalance
Executing 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 Nov 14 09:54:33 CST 2019
end: Thu Nov 14 10:02:43 CST 2019
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
optype: REBALANCEEC cmdno#: 1 status: COMPLETED
arguments: cleanup=false logging=false cmdno=1 cleanupFile=true
operation ID: 763d04a2-f8bc-1901-a07c-5254000f88dc
start: Thu Nov 14 10:02:43 CST 2019
end: Thu Nov 14 10:05:01 CST 2019
duration: 138.117 sec
progress percentage: 100%
time remaining: 0 ms
task count: 2035
completed count: 2035
streamed count: 972
failed count: 0
skipped count: 2
rcvd connection count: 0
open connection count: 0
prqueued count: 1061
job threadpool size: 4
message threadpool active: 0
stream jobs total: 8
stream jobs completed: 8
stream jobs failed: 0
streamed bytes: 325598464
delete count: /10.20.2.135=0 /10.20.2.172=315 /10.20.2.173=342
/10.20.2.174=241 /10.20.2.136=0 /10.20.2.171=74

```

### 11.1.9.3. 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.

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 420)

The rebalance is a background operation that may take many hours or even days, depending on your Hyper-Store cluster size and stored data volume. If you are adding multiple nodes to your cluster, it's OK to have rebalance operations running on multiple new nodes concurrently. See the **"Adding Nodes"** (page 420) procedure for detail.

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 151) feature.

#### 11.1.9.4. *hsstool rebalance* Parameters

*-h <host>*

(Mandatory) Hostname or IP address of the newly added node for which to perform the rebalance operation. In the CMC, only your newly added node(s) will appear in the drop-down node selection list. It is not appropriate to run the *rebalance* command on any node other than a newly added node.

**Note** In the CMC UI for this command this parameter is called "Target Node".

**Note** If you are using the *rebalance* command's "-list" option then the target host can be any host in the region. The returned list will be the list for the region, regardless of which host processes the command.

*-list*

(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 parameter 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 localhost rebalance -list
vm124(10.50.41.49):DC1:ca-sm3    DONE
```

The response format is *host(IPaddress):datacenter:region status*

*-cleanupfile | -cleanup*

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*. **Using the *-cleanupfile* option is the recommended method** in typical circumstances.

- Alternatively, if you use the `-cleanup` option -- instead of the `-cleanupfile` option -- the system will clean up entire token ranges on the appropriate existing nodes, after rebalance tasks (the copying over of replicas or fragments) have successfully completed for those token ranges. This option is appropriately only if you have reason to believe that your system had a lot of "garbage files" -- extra replicas and/or fragments, on nodes on which they do not belong -- before you added the new node.

**Note** If you use neither the `-cleanupfile` option nor the `-cleanup` 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 [hssstool cleanup](#) on each of the older nodes, one node at a time, in order to free up storage space on those nodes. For more details about this approach to cleaning up data after a rebalance operation, see **"When to Use hssstool cleanup"** (page 647).

#### `-retry`

(Optional) If an initial run of `hssstool rebalance` on a new node results in a failure for one or more of the token ranges that the system has assigned to the new node, run `hssstool rebalance` again but this time do it with the `-retry` option. The system will then retry to stream the data from the failed token range(s).

If you wish, when using the `-retry` option you can also use the [-cleanupfile or -cleanup option](#).

**Note** For detailed status information on a rebalance operation that you have already run -- including a break-down of status by token range, so you can see whether any token ranges failed -- you can run `hssstool -h <host> opstatus rebalance -a` (or `opstatus rebalanceec -a`) on the node.

#### `-l <true|false>`

(Optional, defaults to true) If this option is true, entries for each object rebalanced by this operation will be written to the `cloudian-hyperstore-repair.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".

This option defaults to true, so you only need to specify the `-l` option if you do **not** want rebalanced object logging (in which case you'd specify `-l false`).

**Note** This parameter is supported only on the command line -- not in the CMC UI.

### 11.1.9.5. `hssstool rebalance` and `hssstool opstatus rebalance/rebalanceec` Response Items

#### `otype`

The type of `hssstool` operation.

#### `cmdno#`

Command number of the run. Each run of a command is assigned a number.

#### `status`

Status of the command run: INPROGRESS, COMPLETED, or FAILED

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.

A FAILED status means that the operation ended prematurely due to errors. For additional status detail see the other fields in the operation 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.

#### *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.

#### *time remaining*

Estimated time remaining to complete 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.

#### *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 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 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.

*message threadpool active*

In support of implementing the rebalance operation, the current number of active threads managing communications between the target node (the node on which you ran the *rebalance* command) and the other nodes in the cluster.

*stream jobs total*

The system breaks the rebalance streaming operation down into a number of small "jobs", with the number of jobs reflecting factors such as the number of storage policies in the system and the number of token ranges in which data associated with those storage policies is stored. This metric shows the total number of such jobs.

**Note** For rebalance status detail for each individual job (also known as a "session"), run `hsstool -h <host> opstatus rebalance -a` for replicated object data or `hsstool -h <host> opstatus rebalanceec -a` for erasure coded object data. This detailed information can be helpful if you are working with Cloudian 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.

*streamed bytes*

The number of bytes of object data streamed to the new node.

## 11.1.10. `hsstool repair`

Subjects covered in this section:

- *Introduction (immediately below)*
- **"Command Syntax"** (page 686)
- **"Command/Response Example"** (page 687)
- **"When to Use `hsstool repair`"** (page 688)
- **"Problems Remedied by `repair` and `repair computedigest`"** (page 688)

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.

**Note** 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 one 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.

### 11.1.10.1. Command Syntax

The `hsstool repair` command line syntax is as follows. For description of a particular parameter, click on the parameter; or for the full list of parameter descriptions see **"`hsstool repair` Parameters"** (page 689).

```
# hsstool -h <host> repair [allkeyspaces|nokeyspaces] [-pr] [-l <true|false>]
[-m <true|false>] [-computedigest] [-stop] [-resume] [-d <mount-point>] [-rebuild]
[-range <start-token,end-token>] [-t <min-timestamp,max-timestamp>]
```

You can also run the `hsstool repair` command through the CMC UI:

CLLOUDIAN

Analytics Buckets & Objects Users & Groups IAM Cluster Alerts Admin Help

Data Centers Nodes Cluster Config Storage Policies Repair Status Operation Status

Node Status Node Activity Advanced

Command Type: Maintenance

hsstool Command: repair Target Node: jarvis

Options:

☒ default ☐ allkeyspaces ☐ nokeyspaces ☐ pr ☒ l ☒ m ☐ computedigest ☐ stop ☐ resume ☐ rebuild

☐ Mount Point:  ☐ Token Range start:  end:

Description: Repair HyperStore node data. For Cassandra Repair please choose allkeyspaces option.

EXECUTE

**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.

### 11.1.10.2. Command/Response Example

The example below shows a default run of `hsstool repair`, using no options. For description of a particular response item, click on the response item; or for the full list of response item descriptions see **"hsstool repair and hsstool opstatus repair Response Items"** (page 692).

```
# hsstool -h cloudbian-nodel repair
Executing repair.  keyspaces=UserData max-modified-ts=1495109681947 primary-range=false min-
modified-ts=0
logging=true check-metadata=true merkletrue=true computedigest=false
optype: REPAIR cmdno#: 1 status: COMPLETED
arguments: keyspaces=UserData max-modified-ts=1495109681947 primary-range=false min-modified-ts=0
logging=true
check-metadata=true merkletrue=true computedigest=false
operation ID: 085cab00-a069-1f51-92c5-525400814603
start: Fri Dec 08 01:03:53 PDT 2017
end: Fri Dec 08 01:03:58 PDT 2017
duration: 4.629 sec
progress percentage: 100%
total range count: 5
executed range count: 5
failed range count: 0
keyspace count: 1
repair file count: 195
failed count: 0
repaired count: 195
pr queued count: 0
completed count: 195
total bytes streamed: 504196225
```

```
scan time: 0.37325
stream time: 3.74097663698E8
```

**Note** In the example above there is very little data in the system and so the operation completes almost instantly. In a real-world environment this is a long-running operation and the command response will not return until the operation completes. In the meanwhile you can track operation progress as described in the Note in the [Command Format](#) section above.

### 11.1.10.3. 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 *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 443) for details.
- If a node is unavailable for longer than the configurable **"hyper-store.proactiverepair.queue.max.time"** (page 579) (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 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](#).

### 11.1.10.4. 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

Case	Will <i>repair</i> fix it?	Will <i>repair computedigest</i> fix it?
Missing blob file and digest	yes	yes
Corrupted blob file	no	yes
Corrupted digest	yes	yes
Corrupted blob file and digest	yes	yes

### 11.1.10.5. *hsstool repair* Parameters

*-h <host>*

(Mandatory) Hostname or IP address of the node to repair.

**Note** In the CMC UI for this command this parameter is called "Target Node".

*allkeyspaces | nokeyspaces*

(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.

*-pr*

(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 42).

If each node in the cluster is being repaired in succession, using this option makes the successive repair operations less duplicative 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 150).

*-l <true|false>*

(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 Clouidian HyperStore log directory of the target node. Activity associated with a particular instance of a command run is marked with a unique command number.

*-m <true|false>*

(Optional, defaults to true) If this option is true, then before repairing a given object the Merkle Tree based 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*).

*-computedigest*

(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 *"auto\_repair\_computedigest\_run\_number"* (page 518) setting in *common.csv*. By default *"-computedigest"* is not used in auto-repair runs.

*-stop*

(Optional) Use *hsstool -h <host> repair -stop* to terminate 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 666) 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.

**Note** The *-stop* option stops a single in-progress repair on a single node. It does **not** disable the HyperStore [scheduled auto-repair](#) feature.

*-resume*

(Optional) Use this option if you want to resume a previous repair operation that did not complete. This will repair token ranges that were not repaired by the previous repair.

You may want to resume an incomplete repair in any of these circumstances:

- The repair was interrupted by the *repair -stop* command.
- The repair was interrupted by a restart of the target node or a restart of the HyperStore Service on the target node
- The repair reported failed token ranges

**Note** 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*.

**Note** 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 666) results for *repair -resume* operations.

#### *-d <mount-point>*

(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 */cloudian1*), 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.

**Note** For best repair efficiency in repairing a mount point, use the *-rebuild* option together with the *-d <mount-point>* option -- for example *hsstool repair -d /cloudian1 -rebuild*.

**Note** If you use the *-d <mount-point>* option do not use the *-pr* option or the *-range <start-token,end-token>* option. The system does not support using the *-d* option together with the *-pr* option, or the *-d* option together with the *-range* option.

#### *-rebuild*

(Optional) Use this option together with the *-d <mount-point>* option -- for example *hsstool repair -d /cloudian1 -rebuild*. Using the *-rebuild* option makes the mount point repair process more efficient and less time-consuming than it would be without using the *-rebuild* option.

#### *-range <start-token,end-token>*

(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 712) command. The **start-token must be the next-lower token in the ring** from the end-token. Put differently, the start-token is the token that forms the lower boundary of the vNode that's identified by 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>* option. The system does not support using the *-range* option together with the *-pr* option, or the *-range* option together with the *-d* option.

*-t <min-timestamp,max-timestamp>*

(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.

#### 11.1.10.6. *hsstool repair* and *hsstool opstatus repair* Response Items

*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*.

*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 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*

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.

#### *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 Cloudian 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.

#### *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.

#### *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 in seconds that it took to scan the file systems and build the Merkle Tree that is used to detect discrepancies in object replicas across nodes.

#### *stream time*

Total time in seconds that was spent streaming replicas across nodes, in order to implement needed repairs.

## 11.1.11. hsstool repaircassandra

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Command Syntax"** (page 695)

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.

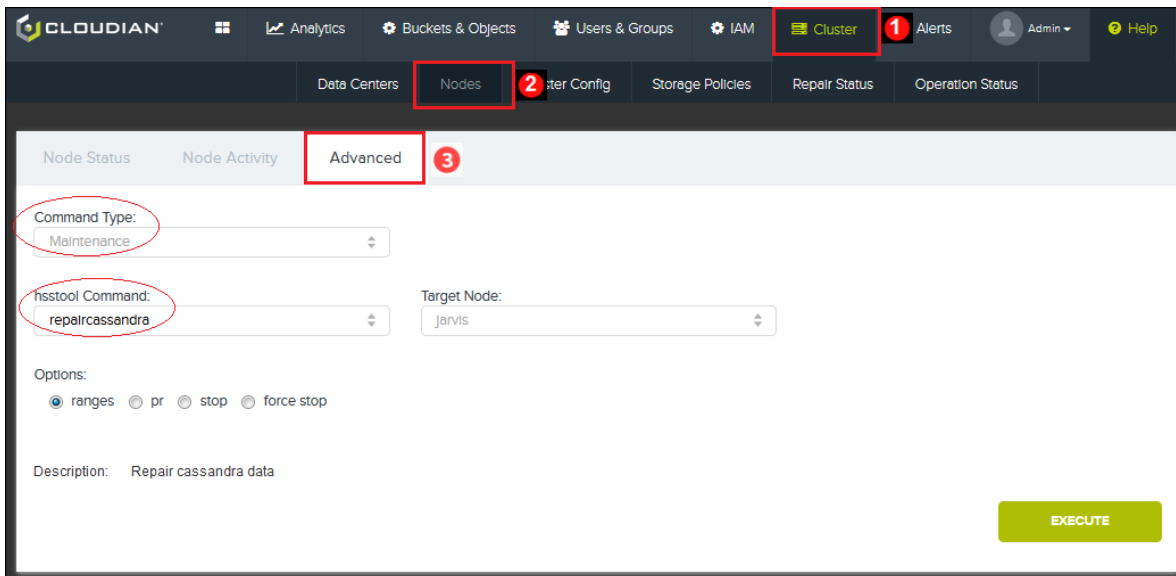
**IMPORTANT !** If you do need to initiate a Cassandra-only repair (with no repair of the object data in the HyperStore File System), use this command rather than using the native Cassandra utility *nodetool* to initiate the repair. Using *hsstool repaircassandra* has multiple advantages over using *nodetool 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.

### 11.1.11.1. Command Syntax

The *hsstool repaircassandra* command line syntax is as follows. For description of a particular parameter, click on the parameter; or for the full list of parameter descriptions see **"hsstool repaircassandra Parameters"** (page 696).

```
# hsstool -h <host> repaircassandra [-pr]
[-keyspace <keyspacename> [<columnfamily> <columnfamily>...]] [-stop [enforce]]
```

You can also run the *hsstool repaircassandra* command 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.

#### 11.1.11.2. *hsstool repaircassandra* Parameters

*-h <host>*

(Mandatory) Hostname or IP address of the node to repair.

**Note** In the CMC UI for this command this parameter is called "Target Node".

*-pr*

(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.

*-keyspace <keyspacename> [<columnfamily> <columnfamily>...]*

(Optional) Use the *-keyspace* option if you want to repair only a specific keyspace in Cassandra (rather than repairing all keyspaces, 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_a36d2c44fbfcc1222e9587b3a42997f CLOUDIAN_OBJMETADATA
```

**Note** The `-keyspace` option is supported only on the command line, not in the CMC UI.

`-stop [enforce]`

(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.

## 11.1.12. hsstool repairec

Subjects covered in this section:

- *Introduction (immediately below)*
- **"Command Syntax"** (page 698)
- **"Command/Response Example"** (page 699)
- **"When to Use hsstool repairec"** (page 700)
- **"Repairing Objects Specified in a File"** (page 700)

Use this [hsstool](#) command to evaluate and repair erasure coded object data. When you run `hsstool repairec` 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 repairec` operation repairs all erasure coded data on all nodes in the data center in which the target node resides.
- For a **distributed erasure coding storage policy spanning multiple data centers**, the `hsstool repairec` 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 repairec` 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 repairec` 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 command also supports options for repairing just a single disk or just a single token range.

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** In a large cluster with high data volume *hsstool repair* is a long-running operation that may take multiple weeks to complete.

**Note** The system will not allow you to run *hsstool 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.
- \* On any node if *hsstool repair* is already running on a different node in the same service region.

### 11.1.12.1. Command Syntax

The *hsstool repair* command line syntax is as follows. For description of a particular parameter, click on the parameter; or for the full list of parameter descriptions see "**hsstool repair Parameters**" (page 702).

```
# hsstool -h <host> repair [-l <true|false>] [-computedigest] [-stop]
[-rebuild] [-d <mount-point>] [-range <start-token,end-token>]
[-f <input-file-path>]
```

You can also run the *hsstool repair* command through the CMC UI:

The screenshot shows the Cloudian CMC UI 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', 'Repair Status', and 'Operation Status'. The 'Nodes' page has three tabs: 'Node Status', 'Node Activity', and 'Advanced' (highlighted with a red box and a red circle with '3'). The 'Advanced' tab contains the following fields:

- Command Type:** A dropdown menu with 'Maintenance' selected.
- hsstool Command:** A dropdown menu with 'repair' selected.
- Target Node:** A dropdown menu with 'jarvis' selected.
- Options:** A section with checkboxes for 'computedigest', 'stop', 'resume', and 'rebuild'. The 'stop' checkbox is checked.
- Mount Point:** A text input field.
- Token Range:** A section with 'start' and 'end' text input fields.
- Description:** A text input field with the value 'Repair erasure code HyperStore node data'.
- EXECUTE:** A green button 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.

### 11.1.12.2. Command/Response Example

The example below shows a default run of *repairrec*, using no options. For description of a particular response item, click on the response item; or for the full list of response item descriptions see "[hsstool repairrec and hsstool opstatus repairrec Response Items](#)" (page 704).

```
# hsstool -h cloudian-node1 repairrec
Executing repairrec. computedigest=false, logging=true, rebuild=false
optype: REPAIRREC cmdno#: 1 status: COMPLETED
arguments: rebuild=false mountPoint=null logging=true range=null cmdno=1 computedigest=false
operation ID: 06186b78-1bec-1be0-a6a5-026a20c18bd8
start: Thu Sep 05 06:58:32 UTC 2019
end: Thu Sep 05 06:58:34 UTC 2019
duration: 2.17 sec
progress percentage: 100%
time remaining: 0 ms
total ranges: 0
task count: 9
completed count: 9
repaired count: 9
failed count: 0
skipped count: 0
rcvd connection count: 0
open connection count: 0
message threadpool active: 0
timer: cassandra.iterating.timer: count=10 mean=28916.792488333995;
repairrec.task.timer: count=0 mean=NaN; digest.scan.per.Ep.timer: count=0 mean=NaN;
distributor.batch.execution.timer: count=1 mean=1.684142835E9;
unit.of.scan.task.timer: count=0 mean=NaN; session.sleep.timer: count=0 mean=NaN;
RocksDB.digests.per.disk.timer: count=0 mean=NaN;
```

**Note** In the example above there is very little data in the system and so the operation completes almost instantly. In a real-world environment this is a long-running operation and the command response will not return until the operation completes. In the meanwhile you can track operation progress as described in the Note in the [Command Format](#) section above.

To get more information about **repair failures** -- in the event that the "failed count" in the *repairrec* response is non-zero -- you can run *hsstool -h <host> opstatus repairrec -a* (the *-a* is the verbose output flag) on the same node on which you ran the *repairrec* operation. This will return the same status metrics that are returned by *repairrec* and *opstatus repairrec*, 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
```

This same information about failure types is logged in [cloudian-hyperstore.log](#), as part of the standard logging of *repair* operations.

Also, in [cloudian-hyperstore-repair.log](#) you can obtain repair status information for specific objects that the *repair* operation determined to be in need of repair.

### 11.1.12.3. 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 *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 *repair* on one node in each of your data centers. See **"Removing a Node"** (page 443) for details.
- If a node is unavailable for longer than the configurable **"hyper-store.proactiverepair.queue.max.time"** (page 579) (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 repair* and [hsstool repair](#)). This will repair objects that were written after the proactive repair queueing time maximum was reached.

### 11.1.12.4. Repairing Objects Specified in a File

The *hsstool repair* command supports an option for repairing one or more specific objects that you list in an input file:

```
# /opt/cloudian/bin/hsstool -h <host> repair -f <input-file-path>
```

For *<input-file-path>*, 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 node can be any node that is part of the erasure coding storage policies used by the objects that you specify in the input file.

**Note** The CMC does not support the `-f <input-file-path>` option. This option is only supported if you use *hsstool repairec* on the command line.

#### 11.1.12.4.1. Input File Format

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 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.

The *hsstool repairec -f <input-file-path>* feature operates on individual chunks, and so in the input file each line must specify a chunk name and the full path to the chunk, in the following format:

```
chunkName|chunkPath
```

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 **paths** within the HyperStore File System see "**HyperStore Service and the HSFS**" (page 23).)

```
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|chunkPath` combination constitutes just one line in the file.

Here is a second example, for an object that was uploaded by a regular 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 Multipart Upload. In this example which specifies one of the object's chunks, the chunk name includes a prefix based on the upload ID, as well as a number suffix (both shown in bold).

```
m-MDA1NTE5NjExNTU0MTIzODU3Mjg1/bucket1/cloudian-hyperfile_v-3.6.tar.gz..0001|/cloudian1/
ec/std8ZdRJDskcPvmOg4/0bb5332b429ccb76466e05bee2915d34/082/100/
39535768889303436494640495599026926454.1554123857285865726-0A3232C9
```

## Collecting Chunk Names and Paths

You have two options for obtaining the chunk name and chunk path for chunks that you want to target for repair. The first 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 second option for obtaining the chunk name and chunk path for chunks that you want to target for repair is to collect failed task information from [cloudian-hyperstore-repair.log](#). In this log, all repaired failed task log entries have this format:

```
FailedTask|timestamp|chunkName|chunkPath|failedReason
```

You can extract the required information from the repair log. An example Linux command for doing so is:

```
zgrep "FailedTask" /var/log/cloudian/cloudian-hyperstore-repair.log | cut -d "|"
-f3,4 >> /tmp/failedTasks.txt
```

This command extracts chunk name and chunk path from the failed task entries in *cloudian-hyperstore-repair.log* and exports that information to */tmp/failedTasks.txt*. You can then use */tmp/failedTasks.txt* as the input file for the *repairrec -f <input-file-path>* command.

### 11.1.12.5. *hsstool repairrec* Parameters

This command supports these parameters:

*-h <host>*

(Mandatory) Hostname or IP address of the target node on which to execute the *repairrec* operation. By default the *repairrec* operation will repair the erasure coded object data on all nodes in the data center in which the target node resides. Consequently as the target node you can choose any node in the data center for which you want to repair erasure coded data.

The exception is if you want to use the *-i* (proactive repair), *-d <mountpoint>*, or *-range <start-token,end-token>* option. For those options it does matter which host you specify as the target. See the description of those options for more detail.

**Note** In the CMC UI for this command this parameter is called "Target Node".

*-l <true|false>*

(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 Clouidian HyperStore log directory of the target node. Activity associated with a particular instance of a command run is marked with a unique command number.

#### *-computedigest*

(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 those 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 *repairec* 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 *"auto\_repair\_computedigest\_run\_number"* (page 518) setting in *common.csv*. By default *"-computedigest"* is not used in auto-repair runs.

#### *-stop*

(Optional) Use *hsstool -h <host> repairec -stop* to abort an in-progress erasure coded data repair. This stops the repair immediately. You can subsequently use the *"hsstool opstatus"* (page 666) 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 stop an in-progress *repairec* operation, you will not be able to resume the operation from the progress point at which you stopped it. Instead, to perform a complete repair of erasure coded data you will need to run the *repairec* operation again from the beginning.

**Note** If the repair that you want to stop is a rebuild of the erasure coded data on a particular disk -- a repair launched as *hsstool -h <host> repairec -d <mountpoint> -rebuild* -- you can stop it with *hsstool -h <host> repairec -stop*. Do not include the *-d <mountpoint> -rebuild* options when executing the stop command.

**Note** The *-stop* option stops a single in-progress repair. It does not disable the HyperStore [scheduled auto-repair](#) feature.

#### *-rebuild*

(Optional) Use the *-rebuild* option together with the *-d <mount-point>* option -- for example *hsstool -h localhost repairec -rebuild -d /cloudian1*. The *-rebuild* option serves to make the repair of erasure coded data on a single target disk more efficient and faster.

#### *-d <mount-point>*

(Optional) If you use this option the repair is performed only for object fragments mapped to the vNodes that are assigned to the specified HyperStore data mount point (for example */cloudian1*) on the target node. This option may be useful in circumstances where data is known or suspected to be missing or incorrect on a particular disk. **Use the *-rebuild* option together with the *-d <mount-point>* option** -- for example *hsstool -h*

*localhost repairc -rebuild -d /cloudian1* . The *-rebuild* option serves to make the repair of erasure coded data on a single target disk more efficient and faster.

**Note** If you use the *-d <mount-point>* option do not use the *-range <start-token,end-token>* option. The system does not support using the *-d* option together with the *-range* option.

*-range <start-token,end-token>*

(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 712) command. The **start-token must be the next-lower token in the ring** from the end-token. Put differently, the start-token is the token that forms the lower boundary of the vNode that's identified by the end-token.

This option may be useful if a previous full node repair reported failures.

**Note** If you use the *-range <start-token,end-token>* option do not use the *-d <mount-point>* option. The system does not support using the *-d* option together with the *-range* option.

*-f <input-file-path>*

(Optional) See **"hsstool repairc"** (page 697).

#### 11.1.12.6. *hsstool repairc* and *hsstool opstatus repairc* Response Items

*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 *repairc* 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.

A TERMINATED status means that the repair run was terminated by an operator, using *repairc -stop*.

*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 `repairec` 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*

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 `repairec` 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*ec 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*ec operation has finished its processing of that chunk, resulting in one of the three outcomes described above.

To get more information about **repair failures** -- in the event that the "failed count" in the *repair*ec response is non-zero -- you can run *hsstool -h <host> opstatus repair -a* (the *-a* is the verbose output flag) on the same node on which you ran the *repair*ec operation. This will return the same status metrics that are returned by *repair*ec and *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
```

This same information about failure types is logged in [cloudian-hyperstore.log](https://cloudian.com/docs/cloudian-hyperstore.log), as part of the standard logging of *repair*ec operations.

Also, in [cloudian-hyperstore-repair.log](https://cloudian.com/docs/cloudian-hyperstore-repair.log) you can obtain repair status information for specific objects that the *repair*ec operation determined to be in need of repair.

**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.

*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 *repair* 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 *repair* 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 *repair* command) and the other nodes in the cluster.

#### *timer*

This shows detailed timing metrics for various parts of the *repair* operation. These metrics may be useful if Cloudian Support is working with you to troubleshoot *repair* performance issues in your environment.

Note that even more detailed timing information for a completed *repair* 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". For example here is a log excerpt showing some of the timing detail (this is from a *repair* run in which there was no data to repair and so the timings are "0"):

```
Performance meters:
Timer name: RocksDB.digests.per.disk.timer, event count: 0, mean rate: 0.0,
recent 15 min rate: 0.0, mean duration: 0.0, median duration: 0.0,
75% events average duration: 0.0, 99% events average duration: 0.0.
Rate unit: events/s, Duration unit: milliseconds.
Timer name: cassandra.iterating.timer, event count: 0, mean rate: 0.0,
recent 15 min rate: 0.0, mean duration: 0.0, median duration: 0.0,
...
...
```

### 11.1.13. hsstool repairqueue

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Command Syntax"** (page 708)
- **"Command/Response Example "** (page 709)
- **"When to Use hsstool repairqueue"** (page 709)

The HyperStore "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. With the *hsstool repairqueue* command you can:

- **Check on the upcoming auto-repair schedule** as well as the status from the most recent auto-repair runs.
- **Temporarily disable auto-repair** for a particular repair type or all types.
- **Re-enable auto-repair**, if it has previously been disabled by the *hsstool repairqueue* command.

For background information on the auto-repair feature, see **"Automated Data Repair Feature Overview"** (page 150).

**Note** You cannot enable or disable auto-repair for just one particular node — the auto-repair feature is either enabled or disabled for the cluster as a whole.

### 11.1.13.1. Command Syntax

The *hsstool repairqueue* command line syntax is as follows. For description of a particular parameter, click on the parameter; or for the full list of parameter descriptions see "**hsstool repairqueue Parameters**" (page 710).

```
# hsstool [-h <host>] repairqueue [-enable true|false] [-t replicas|ec|cassandra] [-inc]
```

In the CMC UI you can run the *hsstool repairqueue* command's auto-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 '1'), 'Alerts', 'Admin', and 'Help'. The main navigation bar includes 'Data Centers', 'Nodes' (highlighted with a red box and '2'), 'Cluster Config', 'Storage Policies', 'Repair Status', and 'Operation Status'. The 'Advanced' tab is selected (highlighted with a red box and '3'). The 'Command Type' dropdown is set to 'Info'. The 'hsstool Command' dropdown is set to 'repairqueue'. The 'Target Node' dropdown is set to 'jarvis'. The description reads 'Display repair queues'. An 'EXECUTE' button is at the bottom right.

The function for disabling and re-enabling auto-repair has its own separate CMC interface and the command is there renamed as "autorepair" (although *hsstool repairqueue* is being invoked behind the scenes):

The screenshot shows the Cloudian CMC interface for the 'autorepair' command. The top navigation bar is the same. The main navigation bar is the same. The 'Advanced' tab is selected. The 'Command Type' dropdown is set to 'Maintenance'. The 'hsstool Command' dropdown is set to 'autorepair'. The 'Target Node' dropdown is set to 'jarvis'. The 'Options' section has 'Enable' selected (radio button), 'Disable' (radio button), and 'Type' set to 'Replicas' (dropdown). The description reads 'Enable/Disable auto-repair of storage data (cluster-wide setting). For more information about this setting, please see the online Help.' An 'EXECUTE' button is at the bottom right.

**Note** If you use this command to disable or re-enable auto-repair and you do not specify a repair type, then the disabling or re-enabling applies also to the "proactive repair" feature. (If you want to disable or re-enable only proactive repair without impacting the scheduled auto-repair feature see [hsstool proactiverepairq.](#))

### 11.1.13.2. Command/Response Example

The first *repairqueue* example below shows the "replicas" auto-repair queue status for a recently installed six-node cluster. For description of a particular response item, click on the response item; or for the full list of response item descriptions see "**hsstool repairqueue Response Items**" (page 711).

```
# hsstool -h cloudian-node1 repairqueue -t replicas

Queue: replicas Auto-repair: enabled #endpoints: 6
1: endpoint: 192.168.204.201 next repair at: Tue Mar 17 22:32:57 PDT 2015 last repair status:
INIT interval: 43200 mins count: 0
2: endpoint: 192.168.204.202 next repair at: Tue Mar 17 22:32:58 PDT 2015 last repair status:
INIT interval: 43200 mins count: 0
3: endpoint: 192.168.204.203 next repair at: Tue Mar 17 22:32:59 PDT 2015 last repair status:
INIT interval: 43200 mins count: 0
4: endpoint: 192.168.204.204 next repair at: Tue Mar 17 22:33:01 PDT 2015 last repair status:
INIT interval: 43200 mins count: 0
5: endpoint: 192.168.204.205 next repair at: Tue Mar 17 22:33:02 PDT 2015 last repair status:
INIT interval: 43200 mins count: 0
6: endpoint: 192.168.204.206 next repair at: Tue Mar 17 22:33:03 PDT 2015 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 cluster**. It does not matter which node you submit the command to.

```
# hsstool -h cloudian-node1 repairqueue -enable false -t replicas
Auto replicas repair disabled
```

### 11.1.13.3. When to Use *hsstool repairqueue*

With the *hsstool repairqueue* command you can disable (and subsequently re-enable) the HyperStore auto-repair feature. You should disable auto-repair before performing the following cluster operation:

- **"Removing a Node"** (page 443)

**IMPORTANT !** If you disable auto-repair in order to perform an operation, be sure to re-enable it afterward.

**Note** The system automatically disables the auto-repair feature when you upgrade your HyperStore software version or when you add nodes to your cluster; and the system automatically re-enables auto-repair after these operations are completed. You do not need to use *hsstool repairqueue* when you perform those operations.

#### 11.1.13.4. *hsstool repairqueue* Parameters

*-h <host>*

(Optional) Hostname or IP address of the node for which to retrieve token and load information. If not supplied, this defaults to the hostname of the host on which you are executing *hsstool*.

**Note** In the CMC UI for this command this parameter is called "Target Node".

*-enable true|false*

(Optional) Enable or disable the auto-repair feature. By default the feature is enabled.

If you use *hsstool -h <host> repairqueue -enable false* to disable the auto-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> repairqueue -enable true* re-enables the auto-repair feature for all nodes in the service region.

If you do not use the optional *-t* flag (described below) to specify an auto-repair type, then the disabling or re-enabling applies to **all auto-repair types and also to the proactive repair feature**. (If you want to disable or re-enable only proactive repair without impacting the scheduled auto-repair feature see "**hsstool pro-activerepair**" (page 674).)

Note that disabling the auto-repair feature **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 686) and "**hsstool repairec**" (page 697)).

**IMPORTANT !** The scheduled auto-repair feature is important for maintaining data integrity in your system. Do not leave it disabled permanently.

**Note** In the CMC UI the enable/disable option is presented as part of a **Maintenance -> autorepair** command rather than the **Info -> repairqueue** command.

*-t replicas|ec|cassandra*

(Optional) You can use this option if you want to restrict the *repairqueue* command action to a particular type of auto-repair -- *replicas* or *ec* or *cassandra*:

- In combination with the *-enable true|false* option, you can use the *-t* option to disable or re-enable just a particular type of auto-repair. For example, use *hsstool -h <host> repairqueue -enable false -t ec* to disable auto-repairs of erasure coded object data. In this example auto-repairs would continue to be enabled for replicated object data and for Cassandra metadata.

**Note** If you do not use the optional `-t` flag to specify an auto-repair type, then the disabling or re-enabling applies to **all auto-repair types and also to the proactive repair feature**. (If you want to disable or re-enable only proactive repair without impacting the scheduled auto-repair feature see "**hsstool proactiverepairq**" (page 674).)

- Without the `-enable true|false` option, you can use the `-t` option to retrieve scheduling information for just a particular type of scheduled auto-repair. For example, use `hsstool -h <host> repairqueue -t replicas` to retrieve scheduling information for auto-repairs of replicated object data. Using `hsstool -h <host> repairqueue` by itself with no `-t` flag will retrieve scheduling information for all auto-repair types.

*-inc*

(Optional) You can use this option in combination with the `-enable true|false -t cassandra` option 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 350).

If you use the `-enable true|false -t cassandra` option without the `-inc` option then your enabling or disabling action applies to both types of Cassandra auto-repair (incremental and full).

**Note** The `-inc` option is only supported on the command line, not in the CMC.

#### 11.1.13.5. *hsstool repairqueue* Response Items

When you use the *repairqueue* command to retrieve 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 686) or **"hsstool repairec"** (page 697)), 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.

## 11.1.14. hsstool ring

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Command Syntax"** (page 712)
- **"Command/Response Example"** (page 713)

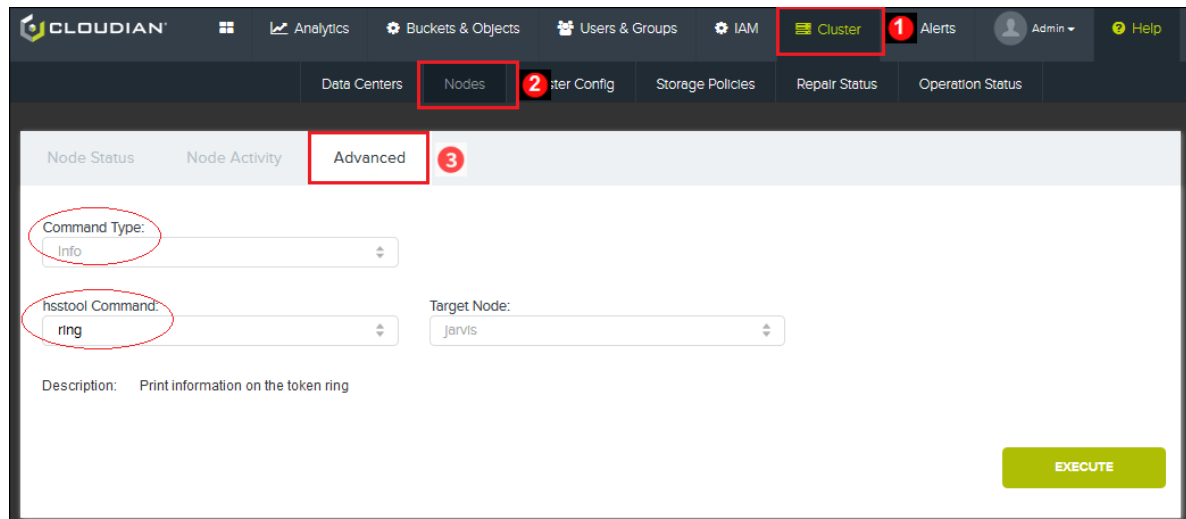
This [hsstool](#) command provides status information for each of the dozens or hundreds of [virtual nodes](#) (vNodes) in your storage cluster. It is very granular and verbose. In most circumstances you will find more value in using the [hsstool info](#) or [hsstool status](#) commands rather than *ring*.

### 11.1.14.1. Command Syntax

The *hsstool ring* command line syntax is as follows.

```
# hsstool [-h <host>] ring
```

You can also run the *hsstool ring* command through the CMC UI:



### 11.1.14.2. 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. For description of a particular response item, click on the response item; or for the full list of response item descriptions see "**hsstool ring Response Items**" (page 714).

```
# 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							
...							
...							

### 11.1.14.3. *hsstool ring* Parameters

*-h <host>*

(Optional) Hostname or IP address of the node for which to retrieve token and load information. If not supplied, this defaults to the hostname of the host on which you are executing *hsstool*.

**Note** In the CMC UI for this command this parameter is called "Target Node".

#### 11.1.14.4. *hsstool ring* 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).

#### 11.1.15. *hsstool status*

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Command Syntax"** (page 714)
- **"Command/Response Example"** (page 715)

This [hsstool](#) command returns status information for a storage cluster as a whole.

##### 11.1.15.1. Command Syntax

The *hsstool status* command line syntax is as follows.

```
# hsstool [-h <host>] status
```

You can also run the *hsstool status* 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'. Below this, the 'Nodes' tab is selected (highlighted with a red box and a red circle with '2'). Under the 'Nodes' tab, the 'Advanced' sub-tab is selected (highlighted with a red box and a red circle with '3'). In the 'Advanced' sub-tab, 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 below the dropdowns reads 'Print cluster information'. An 'EXECUTE' button is located at the bottom right of the form.

### 11.1.15.2. Command/Response Example

The *status* command example below retrieves the status of a four-node cluster. For description of a particular response item, click on the response item; or for the full list of response item descriptions see "**hsstool status Response Items**" (page 716).

```
# hsstool -h localhost status
Datacenter: DC1
=====
```

Address	DC	Rack	Cassandra	Tokens	Cassandra-Load	Hss	State	Hyperstore-Disk	Host-ID
10.20.2.54	DC1	RAC1	Up	32	0.24118415	Up	Normal	427.66MB/7.75GB (5.39%)	cfa851df-4170-4006-a516-d5b56f99a820 cld05-04
10.20.2.57	DC1	RAC1	Down	32	0.2239672	Up	Normal	79.45MB/11.17GB (0.69%)	6bf98091-5318-4ae6-acc3-f1524f5e7da3 cld05-07
10.20.2.55	DC1	RAC1	Up	32	0.23458646	Up	Normal	1.1GB/20.48GB (5.36%)	c77ff93a-5579-4f9f-b5a3-9e22d68fab84 cld05-05
10.20.2.52	DC1	RAC1	Up	32	0.3002622	Up	Normal	692.7MB/12.61GB (5.37%)	914117cb-3ec5-4c79-9dff-5a9f33002ce3 cld05-02

### 11.1.15.3. *hsstool status* Parameters

*-h <host>*

(Optional) Hostname or IP address of the node for which to retrieve token and load information. If not supplied, this defaults to the hostname of the host on which you are executing *hsstool*.

**Note** In the CMC UI for this command this parameter is called "Target Node".

#### 11.1.15.4. *hsstool status* Response Items

##### *Address*

IP address of the node.

##### *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 leaving the cluster), or "?" (host cannot be reached).

##### *Tokens*

Number of tokens (also known as vNodes) assigned to the physical node. The tokens assigned to a physical node determine which S3 objects will be stored on the node. For more information on how tokens are assigned to physical nodes, see **"How vNodes Work"** (page 42).

##### *Cassandra-Load*

From among the total token space in the storage cluster, the portion of token space that is owned by this node. This is expressed as a decimal value. For example, if 25% of the cluster's total token space is owned by this node, this field displays .25.

Note that this is a different meaning of "Cassandra-Load" than is used for *hsstool ring* and *hsstool info* results.

##### *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*

The total volume of S3 object data stored in the HyperStore File System on the node. This includes S3 objects for which the node serves as a secondary or tertiary replica as well as S3 objects for which the node is the primary replica.

This field also shows the total amount of disk space designated for S3 object storage on the node.

##### *Host-ID*

System-generated hexadecimal number uniquely identifying the physical node. Note that with the use of vNodes, tokens uniquely identify vNodes while Host IDs uniquely identify each physical node within the cluster.

##### *Hostname*

Hostname of the node.

## 11.1.16. hsstool trmap

[Command]

Subjects covered in this section:

- *Introduction (immediately below)*
- **"Command Syntax"** (page 717)
- **"Command/Response Examples"** (page 718)

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 Clodian Support to troubleshoot a failed attempt to add nodes to your HyperStore cluster or a failed attempt to rebalance the cluster after adding nodes.

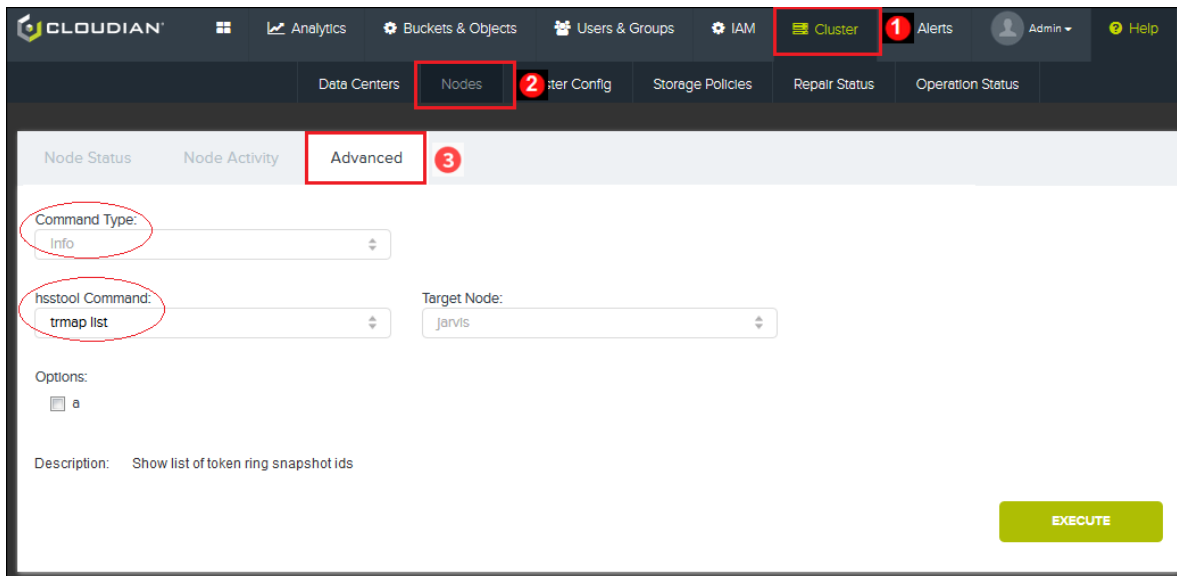
### 11.1.16.1. Command Syntax

The *hsstool trmap* command line syntax is as follows. For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"hsstool trmap Parameters"** (page 719).

```
# /opt/cloudian/bin/hsstool -h <host> trmap [list \[-a\]] [show <snapshotId>]
\[set <snapshotId> <active/disabled>\] [delete <snapshotId>]
```

**IMPORTANT !** Do not use the *set* or *delete* options unless instructed to do so by Clodian Support.

You can also run the *hsstool trmap* command through the CMC UI:



### 11.1.16.2. 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/cloudbian/bin/hsstool -h localhost trmap list
061e9190-6d62-429d-85dc-e4baec07f49e Wed Jun 19 15:26:07 EDT 2019
22351e62-e921-4130-afaf-1ca3183046e1 Thu Jun 20 14:17:04 EDT 2019
1c84d57e-dccb-47b0-af6d-015e823eab07 Wed Jun 19 21:42:42 EDT 2019
9e096b8f-5914-42fc-8dc7-063de85deb4a Thu Jun 20 06:56:17 EDT 2019
```

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. For description of a particular response item, click on the response item; or for the full list of response item descriptions see **"hsstool trmap Response Items"** (page 720).

```
# /opt/cloudbian/bin/hsstool -h cloudbian-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" : [ {
    ...
    ...
  } ]
} ]

```

#### See Also:

- "How vNodes Work" (page 42)

### 11.1.16.3. *hsstool trmap* Parameters

*-h <host>*

(Mandatory) 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.

**Note** In the CMC UI for this command this parameter is called "Target Node".

*list [-a]*

(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 localhost trmap list -a
59af5410-b303-41bd-94a0-31ce92058e11 Tue Jun 26 09:03:56 EDT 2018 DISABLED
061e9190-6d62-429d-85dc-e4baec07f49e Wed Jun 19 15:26:07 EDT 2019 ACTIVE
22351e62-e921-4130-afaf-1ca3183046e1 Thu Jun 20 14:17:04 EDT 2019 ACTIVE
1c84d57e-dccb-47b0-af6d-015e823eab07 Wed Jun 19 21:42:42 EDT 2019 ACTIVE
9e096b8f-5914-42fc-8dc7-063de85deb4a Thu Jun 20 06:56:17 EDT 2019 ACTIVE

```

```
3bf72d7e-a2a1-4551-972c-1223a4485335 Mon Jun 25 17:23:32 EDT 2018 DISABLED
dbf98852-2557-4cfd-8708-0398009fc6d7 Mon Jun 25 18:38:47 EDT 2018 DISABLED
```

*show <snapshotId>*

(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>*

(Optional) **Do not use this option unless instructed to do so by Clouidian 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>*

(Optional) **Do not use this option unless instructed to do so by Clouidian Support.** This deletes the specified token range map snapshot. This option is only supported on the command line, not in the CMC interface.

#### 11.1.16.4. *hsstool tmap* 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 420).

*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.

## 11.1.17. hsstool whereis

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Command Syntax"** (page 721)
- **"Command/Response Examples "** (page 722)
- **"Log File for "whereis -a""** (page 723)

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.

### 11.1.17.1. Command Syntax

The *hsstool whereis* command line syntax is as follows. For description of a particular parameter, click on the parameter; or for the full list of parameter descriptions see **"hsstool whereis Parameters"** (page 723).

```
# hsstool [-h <host>] whereis <bucket>/<object> [-v <version>] [-ck] [-a]
```

You also can run the *hsstool whereis* command through the CMC UI:

### 11.1.17.2. 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 detail information for the object replica is truncated in this example. For description of a particular response item, click on the response item; or for the full list of response item descriptions see "**hsstool whereis Response Items**" (page 724).

```
# 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/111t...
```

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 (a configuration that is no longer supported). Note that the response indicates that one of the expected six fragments is missing. The detail information for each found fragment is truncated in this example. For description of a particular response item, click on the response item; or for the full list of response item descriptions see "**hsstool whereis Response Items**" (page 724).

```
# hsstool -h cloudian-node3 whereis bucket1/obj1.txt
Key: bucket1/obj1
Policy ID: 1d140a90b17285cf2d1502cbd424d621
Version: null
Compression: NONE
Create Time: 2018-01-29T23:07:52.626Z
Last Modified: 2018-01-29T23:07:52.626Z
```

```

Last Access Time: 2018-01-29T23:07:52.626Z
Size: 11231
Type: EC
Region: region1
5 out of 6 fragments were found, 1 fragments missing
[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] 0 replication found

```

**Note** For objects for which the Type is "TRANSITIONED" (auto-tiered), the response will also include a [URL](#) field.

### 11.1.17.3. 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. These rotation settings are configurable in the *RollingRandomAccessFile name="APP"* section of the */etc/cloudian-<version>-puppet/modules/cloudians3/templates/log4j-hsstool.xml.erb* file. For information about changing these settings see "Log Configuration Settings" (page 626).

### 11.1.17.4. *hsstool whereis* Parameters

*-h <host>*

(Optional) Hostname or IP address of the node for which to retrieve token and load information. If not supplied, this defaults to the hostname of the host on which you are executing *hsstool*.

**Note** In the CMC UI for this command this parameter is called "Target Node".

*<bucket>/<object>*

(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\_2016-06-27.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\_2016-06-27.mpg*.

*-v <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*

(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.

*-a*

(Optional) Use the *-a* option if you want a full list of **all replicas 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.

#### 11.1.17.5. *hsstool whereis* Response Items

##### *Key*

Key that uniquely identifies the S3 object, in format *<bucketname>/<objectname>*. For example, *bucket1/Documents/Meetings\_2018-06-27.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. An object's 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*

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*.

*Location detail*

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>
```

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>
```

- 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 (for background information see **"How vNodes Work"** (page 42)).
- 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..
- 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.

**Note** If an object replica or fragment is supposed to be on a node (according to system metadata) but is missing, the node's IP address is listed in the command results along with a message stating "0 replication found". For example, "[10.10.3.52] 0 replication found".

**Note** For multipart objects (large objects uploaded via the S3 multipart upload method), storage location detail is shown **for each part**.

#### Fragment count summary

For erasure coded objects, the *hsstool whereis* response includes a line stating the number of fragments found and (if applicable) the number of fragments missing. For objects for which all expected fragments were found, the line will state "x out of x fragments were found". For objects for which one or more of the expected fragments are missing, the line will state "x out of y fragments were found, z fragments missing".

## 11.2. Redis Monitor Commands

The HyperStore **"Redis Monitor Service"** (page 27) monitors Redis Credentials and Redis QOS cluster health and implements automatic failover of the Redis 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***

If you are using the [HyperStore Shell \(HSH\)](#) as a [Trusted user](#), at the HSH prompt you can open a *netcat* session to the Redis Monitor port:

```
$ nc <redismon_primary_host> 9078
```

You can then use any of the Redis Monitor CLI commands listed below. When you're done, enter **quit** to end your Redis Monitor CLI session and then **<ctrl>-d** to end your *netcat* session and return to the HSH prompt.

- **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*) -- for all the other Redis Monitor commands you must use the Redis Monitor CLI or a JMX client.

The Redis Monitor supports the following commands:

## 11.2.1. get cluster

### get cluster

Use this command to retrieve basic status information that the Redis Monitor currently has for a specified Redis cluster. The cluster status information includes:

- The identity of the Redis 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)

#### 11.2.1.1. Command Line Syntax

```
get cluster redis.credentials|redis.qos.<region>
```

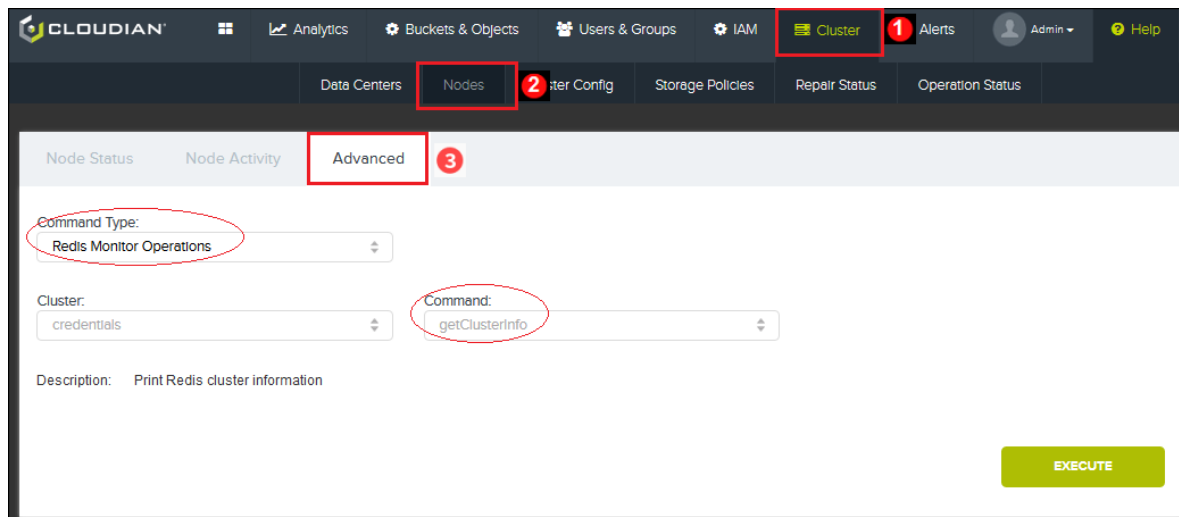
**Note** For this and all other Redis Monitor commands, the Redis QoS cluster identifier includes the name of the service region in which the cluster resides. This is necessary since in a multi-region Hyper-Store system each region has its own Redis QoS cluster. By contrast the Redis Credentials cluster, 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 21 16:17:23 PDT 2018
```

### 11.2.1.2. CMC UI

Path: **Cluster** → **Nodes** → **Advanced**



### 11.2.2. 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.

### 11.2.2.1. 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
```

## 11.2.3. 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.

### 11.2.3.1. 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]]
```

## 11.2.4. 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.

### 11.2.4.1. 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.

## 11.2.5. 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.

#### 11.2.5.1. Command Line Syntax

```
enable monitoring redis.credentials|redis.qos.<region>
```

Example:

```
# nc 10.50.20.12 9078  
enable monitoring redis.credentials  
OK enabled
```

## 11.2.6. 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.)

#### 11.2.6.1. Command Line Syntax

```
disable monitoring redis.credentials|redis.qos.<region>
```

Example:

```
# nc 10.50.20.12 9078  
disable monitoring redis.qos.region1  
OK disabled
```

## 11.2.7. 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.

### 11.2.7.1. Command Line Syntax

```
enable notifications redis.credentials|redis.qos.<region>
```

Example:

```
# nc 10.50.20.12 9078
enable notifications redis.credentials
OK enabled
```

## 11.2.8. 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 730).

### 11.2.8.1. Command Line Syntax

```
disable notifications redis.credentials|redis.qos.<region>
```

Example:

```
# nc 10.50.20.12 9078
disable notifications redis.qos.region1
OK disabled
```

## 11.2.9. 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 Redis Credentials Master or QoS Master Role"** (page 458).

### 11.2.9.1. 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.

### 11.2.9.2. CMC UI

Path: **Cluster** → **Nodes** → **Advanced**

The screenshot shows the Cloudian CMC UI 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 'Redis Monitor Operations' (circled in red). The 'Cluster' dropdown is set to 'credentials'. The 'Command' dropdown is set to 'setClusterMaster' (circled in red). The 'Hostname' dropdown is set to 'cloudian-node1'. A description at the bottom reads: 'Assign a Redis cluster's master role to a different node within the cluster'. An 'EXECUTE' button is located at the bottom right.

In the CMC UI, use the "Hostname" field to specify the host to which you want to move the Redis master role.

## 11.2.10. 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 Puppet master 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.

### 11.2.10.1. 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
```

## 11.2.11. 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 420)), 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).

### 11.2.11.1. 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
```

## 11.2.12. 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.

### 11.2.12.1. 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
```

### 11.2.13. 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.

### 11.2.13.1. 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
```

## 11.2.14. 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 733)).

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.** This is controlled by the setting **"redis-monitor.skip.dc.monitoring"** (page 571) in [mts.properties.erb](#) (which defaults to true, so that DC partition monitoring/failover is "skipped" [disabled]).

Since DC partition monitoring/failover is disabled by default, the only circumstances in which you might want to use the *disable dc partition monitoring* command is if you have previously changed the *redis-monitor.skip.dc.monitoring* configuration property (so that DC partition monitoring/failover is enabled by configuration) or if you have previously used the [enable dc partition monitoring](#) command (so that DC partition monitoring/failover is enabled in the current session of the Redis Monitor).

**Note** If the Redis Monitor (or its host) is restarted, it will revert to using the value of the *redis-monitor.skip.dc.monitoring* configuration property to determine whether DC partition monitoring/failover is enabled or disabled. Note also that the configuration property applies to all Redis clusters in the system, while the enable/disable commands apply only to the Redis cluster that you specify when you run the command.

**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.

### 11.2.14.1. 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
```

## 11.2.15. 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 733)).

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.** This is controlled by the setting **"redis-monitor.skip.dc.monitoring"** (page 571) in [mts.properties.erb](#) (which defaults to true, so that DC partition monitoring/failover is "skipped" [disabled]).

**Note** If the Redis Monitor (or its host) is restarted, it will revert to using the value of the *redis-monitor.skip.dc.monitoring* configuration property to determine whether DC partition monitoring/failover is enabled or disabled. Note also that the configuration property applies to all Redis clusters in the system, while the enable/disable commands apply only to the Redis cluster that you specify when you run the command.

**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.

### 11.2.15.1. 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
```

## 11.2.16. 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.** This is controlled by the setting "**redis-monitor.skip.brain.monitoring**" (page 571) in [mts.properties.erb](#) (which defaults to false, so that split brain monitoring/resolution is enabled [is not "skipped"]).

**Note** If the Redis Monitor (or its host) is restarted, it will revert to using the value of the *redis-monitor.skip.brain.monitoring* configuration property to determine whether split brain monitoring/resolution is enabled or disabled. Note also that the configuration property applies to all Redis clusters in the system, while the enable/disable commands apply only to the Redis cluster that you specify when you run the command.

**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.

### 11.2.16.1. 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
```

## 11.2.17. 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.** This is controlled by the setting "**redis-monitor.skip.brain.monitoring**" (page 571) in [mts.properties.erb](#) (which defaults to false, so that split brain monitoring/resolution is enabled [is not "skipped"]).

Since split brain monitoring/resolution is enabled by default, the only circumstances in which you might want to use the *enable split brain monitoring* command is if you have previously changed the *redis-monitor.skip.brain.monitoring* configuration property (so that split brain monitoring/resolution is disabled by configuration) or if you have previously used the [disable split brain monitoring](#) command (so that split brain monitoring/resolution is disabled in the current session of the Redis Monitor).

**Note** If the Redis Monitor (or its host) is restarted, it will revert to using the value of the *redis-monitor.skip.brain.monitoring* configuration property to determine whether split brain monitoring/resolution is enabled or disabled. Note also that the configuration property applies to all Redis clusters in the system, while the enable/disable commands apply only to the Redis cluster that you specify when you run the command.

**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.

### 11.2.17.1. 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
```

## 11.2.18. resolve split brain

### resolve split brain

This command is applicable **only if you have previously disabled automatic split brain resolution** (either by having changed the `redis.monitor.skip.brain.monitoring` configuration property to "true" in `mts.properties.erb`, or by having run the [disable split brain monitoring](#) command).

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.

### 11.2.18.1. 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
```

This page left intentionally blank

# Chapter 12. Admin API

## 12.1. Introduction

### 12.1.1. HyperStore Admin API Introduction

**IMPORTANT !** The Admin API is not designed to be exposed to end users of the Clouidian HyperStore storage service. It is intended to be accessed only within an internal network, by the CMC and by system administrators using other types of clients (such as cURL). Do not expose the Admin Service to an external network.

Clouidian HyperStore provides a RESTful HTTP API through which you can provision users and groups, manage rating plans and quality of service (QoS) controls, retrieve monitoring data, and perform other administrative tasks. This Admin API is implemented by the Admin Service, which runs on the same nodes as your S3 Service.

By default the HTTPS listening port for the Admin Service is 19443 and the HTTP port is 18081. In HyperStore systems for which the first installed version was 6.0.2 or later, the Admin Service supports **only HTTPS** connections, and clients are required to use Basic Authentication. (For more detail see "**HTTP and HTTPS for Admin API Access**" (page 747) and "**HTTP/S Basic Authentication for Admin API Access**" (page 748)).

The Clouidian Management Console (CMC) accesses the Admin API to implement its provisioning and reporting functions. You also have the option of accessing the Admin API directly, using a command line tool such as cURL or a REST client application of your own creation. When you access the Admin API directly, you can submit requests to any HyperStore node in your **default service region**.

HyperStore Admin API response payloads are JSON encoded. For POST or PUT requests that require a request payload, the request payloads must be JSON encoded as well.

#### 12.1.1.1. RBAC Versions of Admin API Methods

For some read-only Admin API methods, there are alternative versions of the method implemented in the HyperStore IAM Service. This feature provides for granular role-based access control (RBAC) to a subset of HyperStore administrative functions, invoked by making calls to the HyperStore IAM Service (rather than the Admin Service). For more information on this feature, including information about the client tool that HyperStore provides to help you use this feature, see "**IAM Extensions for Role-Based Access to HyperStore Admin Functions**" (page 1027).

#### 12.1.1.2. Admin API Behavior in Multi-Region Systems

If your HyperStore system has multiple [service regions](#), then:

- The Admin Service in the **default service region** supports executing all of the Admin API operations in this document.
- The Admin Service in **regions other than the default region** supports executing only the following subset of Admin API operations:

- [POST /usage/storage](#)
- [POST /usage/storageall](#)
- [POST /usage/rollup](#)
- [POST /usage/repair/dirtyusers](#)

If in a non-default region you send your local Admin Service a request to execute an operation other than those listed above, you will receive a 403:Forbidden response.

Consequently, in a multi-region system your [DNS configuration](#) must resolve the Admin Service endpoint to nodes in the default service region. The CMC will use this endpoint to submit requests to the Admin API. And if you access the Admin API directly -- through a command line tool or a client application of your own creation -- you must submit the requests to nodes in the default service region (with the exception of the four calls listed above)

For API calls that involve retrieving data from multiple regions, this is all handled by the Admin Service in the default region. For example in a [GET /usage](#) call submitted to the Admin Service in your default service region you can retrieve service usage data for all of your regions or for any single one of your regions.

#### See Also:

- "Admin API Methods List" (page 742)
- "Common Request and Response Headers" (page 745)
- "Common Response Status Codes" (page 746)
- "cURL Examples" (page 746)
- "HTTP and HTTPS for Admin API Access" (page 747)
- "HTTP/S Basic Authentication for Admin API Access" (page 748)
- "Admin API Logging" (page 750)

### 12.1.2. Admin API Methods List

The table below shows all of the HyperStore Admin API methods. For more detail about a method or methods, click on the corresponding Resource link.

Resource	Method	Purpose
<a href="#">billing</a>	GET /billing	Get a bill for a user or group
	POST /billing	Create a bill for a user or group
<a href="#">bppolicy</a>	GET /bppolicy/bucketsperpolicy	Get list of buckets using each storage policy
	GET /bppolicy/listpolicy	Get list of storage policy IDs
<a href="#">bucketops</a>	GET /bucketops/id	Get a bucket's canonical ID
	GET /bucketops/gettags	Get bucket tags for users in a group
	POST /bucketops/purge	Delete all the objects in a bucket

Resource	Method	Purpose
<a href="#">group</a>	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
<a href="#">monitor</a>	DELETE /monitor/notificationrule	Delete a notification rule
	GET /monitor/events	Get the event list for a node
	GET /monitor/nodelist	Get the list of monitored nodes
	GET /monitor/host	Get current monitoring statistics for a node
	GET /monitor	Get current monitoring statistics for a service region
	GET /monitor/history	Get historical monitoring statistics for a node
	GET /monitor/notificationrules	Get the list of notification rules
	POST /monitor/acknowledgeevents	Acknowledge monitoring events
	POST /monitor/notificationruleenable	Enable or disable notification rules
	POST /monitor/notificationrule	Change a notification rule
	PUT /monitor/notificationrule	Create a new notification rule
<a href="#">permissions</a>	GET /permissions/publicUrl	Get public URL permissions for an object
	POST /permissions/publicUrl	Create or change public URL permissions for an object
<a href="#">qos</a>	DELETE /qos/limits	Delete QoS settings for a user or group
	GET /qos/limits	Get QoS settings for a user or group
	POST /qos/limits	Create QoS settings for a user or group
<a href="#">ratingPlan</a>	DELETE /ratingPlan	Delete a rating plan
	GET /ratingPlan	Get a rating plan
	GET /ratingPlan/list	Get the list of rating plans in the system
	POST /ratingPlan	Change a rating plan
	PUT /ratingPlan	Create a new rating plan

Resource	Method	Purpose
<a href="#">system</a>	GET /system/audit	Get summary counts for system
	GET /system/bucketcount	Get count of buckets owned by a group's members
	GET /system/bucketlist	Get list of buckets owned by a group's members
	GET /system/bytecount	Get stored byte count for the system, a group, or a user
	GET /system/bytestiered	Get tiered byte count for the system, a group, or a user
	GET /system/groupbytecount	Get stored byte counts for all of a group's users
	GET /system/groupobjectcount	Get stored object counts for all of a group's users
	GET /system/license	Get HyperStore license terms
	GET /system/objectcount	Get stored object count for the system, a group, or a user
	GET /system/version	Get HyperStore system version
	POST /system/processProtectionPolicy	Process pending storage policy deletion or creation jobs
	POST /system/repairusercount	Reconcile user counts in Redis and Cassandra
<a href="#">tiering</a>	DELETE /tiering/credentials	Delete a tiering credential for Amazon, Google, or other S3-compliant destination
	DELETE /tiering/azure/credentials	Delete a tiering credential for Azure
	DELETE /tiering/spectra/credentials	Delete a tiering credential for Spectra
	GET /tiering/credentials	Get a tiering credential for Amazon, Google, or other S3-compliant destination
	GET /tiering/credentials/src	Check whether a bucket uses a bucket-specific or system default tiering credential
	GET /tiering/azure/credentials	Get a tiering credential for Azure
	GET /tiering/spectra/credentials	Get a tiering credential for Spectra
	POST /tiering/credentials	Post a tiering credential for Amazon, Google, or other S3-compliant destination
	POST /tiering/azure/credentials	Post a tiering credential for Azure
	POST /tiering/spectra/credentials	Post a tiering credential for Spectra

Resource	Method	Purpose
<a href="#">usage</a>	DELETE /usage	Delete usage data
	GET /usage	Get usage data for group, user, or bucket
	POST /usage/bucket	Get raw usage data for multiple buckets
	POST /usage/repair	Repair storage usage data for group or system
	POST /usage/repair/bucket	Retrieve total bytes and total objects for a bucket
	POST /usage/repair/dirtyusers	Repair storage usage data for users with recent activity
	POST /usage/repair/user	Repair storage usage data for a user
	POST /usage/rollup	Roll up usage data
	POST /usage/storage	Post raw storage usage data for users with recent activity
	POST /usage/storageall	Post raw storage usage data for all users
<a href="#">user</a>	DELETE /user	Delete a user
	DELETE /user/credentials	Delete a user's S3 security credential
	DELETE /user/deleted	Purge profile data of a deleted user or users
	GET /user	Get a user's profile
	GET /user/credentials	Get a user's S3 security credential
	GET /user/credentials/list	Get a user's list of S3 security credentials
	GET /user/credentials/list/active	Get a user's list of active S3 security credentials
	GET /user/list	Get a list of user profiles
	GET /user/password/verify	Verify a user's CMC password
	GET /user/ratingPlan	Get a user's rating plan content
	GET /user/ratingPlanId	Get a user's rating plan ID
	POST /user	Change a user's profile
	POST /user/credentials	Post a user's supplied S3 credential
	POST /user/credentials/status	Deactivate or reactivate a user's S3 credential
	POST /user/password	Create or change a user's CMC password
	POST /user/ratingPlanId	Assign a rating plan to a user
	PUT /user	Create a new user
	PUT /user/credentials	Create a new S3 credential for a user
<a href="#">whitelist</a>	GET /whitelist	Get whitelist content
	POST /whitelist	Change whitelist content (by request body object)
	POST /whitelist/list	Change whitelist content (by query parameters)

### 12.1.3. Common Request and Response Headers

#### 12.1.3.0.1. Common Request Headers

For PUT requests, the Content-Type header should be set to "application/json". For POST requests, the Content-Type header should be set to "application/json", "application/x-www-form-urlencoded", or "multipart/form-

data".

Depending on the request type and the result, the response from the Admin API may be in format `application/json`, `text/html`, or `text/plain`. So in your requests do not use an `Accept` header that excludes these content types.

### 12.1.3.0.2. Common Response Headers

In responses, the `Content-Type` will be either `"application/json"`, `"text/html"`, or `"text/plain"` depending on the type of request being processed and the result.

## 12.1.4. Common Response Status Codes

The following HTTP Status Codes are relevant for every Admin API method. Each method may return these codes, in addition to method-specific status codes indicated in the method documentation.

Status Code	Description
200	OK
500	Internal Server Error
404	<p>Not found.</p> <div> <p><b>Note</b> URIs for the Admin API are <b>case-sensitive</b>. If you submit a request wherein the case of the specified request resource and URI parameters does not match the case documented in this Admin API Guide — or a request wherein the URI contains any other typographical error — the system will return a 404 error response.</p> </div>

The following HTTP Status Code is relevant for all Admin API methods that are forbidden in the non-default regions of a multi-region HyperStore system.

Status Code	Description
403	<p>Not allowed. Only the Admin API service in the default region can execute this method.</p> <p>The only Admin API methods that are allowed in non-default regions of a multi-region deployment are:</p> <ul style="list-style-type: none"> <li>◦ <a href="#">POST /usage/storage</a></li> <li>◦ <a href="#">POST /usage/storageall</a></li> <li>◦ <a href="#">POST /usage/rollup</a></li> <li>◦ <a href="#">POST /usage/repair/dirtyusers</a></li> </ul> <p>For any other Admin API method, submitting the method request to a non-default region's Admin Service will result in the 403 response.</p>

## 12.1.5. cURL Examples

This Admin API documentation includes examples using the open source command-line utility [cURL](#). When a JSON object is the expected response payload, the example commands pipe the output through the standard Python tool `mjson.tool` so that the JSON pretty-prints. If you wish you can copy the commands from the documentation, paste them on to your command line, customize them appropriately, and the commands should

work against your local HyperStore Admin Service (so long as you have cURL and Python on your local machine).

Here is a sample command, for retrieving a user group's profile:

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/group?groupId=QA | python -mjson.tool
```

With this example you would replace "sysadmin" and "public" with whatever the [HTTP Basic Authentication user name and password](#) are in your HyperStore system; replace "localhost" with the IP address of one of your HyperStore nodes in the default service region; and replace "QA" with the name of one of your user groups. Note that the backslash in this and other examples indicates line continuation -- telling the Linux shell to ignore the newline for purposes of running the command. These are used in the examples so that a long command can be split to multiple lines in this documentation, while still allowing you to copy all the text (including the backslash) and paste it on your command line and be able to run the command.

**Note** By default the Admin Server uses a self-signed SSL certificate and so in the example cURL commands the "-k" flag is used to disable certificate checking.

In cases where a JSON object is required as the request payload, the examples use the cURL "-d" flag to reference the name of a text file that contains the JSON object. For example:

```
curl -X PUT -H "Content-Type: application/json" -k -u sysadmin:public \
-d @group_QA.txt https://localhost:19443/group
```

In the full documentation the content of the referenced text file is also shown. Note that if a request includes multiple query parameters with ampersand delimitation, the URL must be enclosed in single quotes -- as in this example which deletes a user:

```
curl -X DELETE -k -u sysadmin:public \
'https://localhost:19443/user?userId=John&groupId=QA'
```

## 12.1.6. HTTP and HTTPS for Admin API Access

The Admin Service by default requires clients to use HTTPS. It rejects attempts to connect with regular HTTP. The exception is if your original HyperStore installation was a version older than 6.0.2 -- for such systems, both HTTPS and HTTP are accepted by default. For information about configuring such systems so that the Admin Service accepts only HTTPS connections and not regular HTTP connections, see **"If Your Original HyperStore Install Was Older Than Version 6.0.2"** (page 747).

The Admin Service requires Basic Authentication from connecting HTTP(S) clients. For more information including how to customize the required Basic Authentication password see **"HTTP/S Basic Authentication for Admin API Access"** (page 748).

### 12.1.6.1. If Your Original HyperStore Install Was Older Than Version 6.0.2

If your original HyperStore install was older than version 6.0.2 and you have upgraded to the current version, the Admin Service by default accepts regular HTTP requests (through port 18081) as well as HTTPS requests (through port 19443). Also for such systems, the CMC uses regular HTTP when submitting requests to the Admin Service.

If you want the Admin Service to accept **only HTTPS** requests from clients -- and to reject regular HTTP requests -- follow the steps below. Following these steps also has the effect of reconfiguring the CMC so that it uses exclusively HTTPS when submitting requests to the Admin Service.

1. On your Puppet master, open this configuration file in a text editor:

```
/etc/cloudian-7.2.3-puppet/manifests/extdata/common.csv
```

2. Anywhere in the "S3/Admin Services" section of *common.csv*, add this line:

```
admin_secure,true
```

**Note** By default the "admin\_secure" setting does not appear in the *common.csv* file for systems that were originally installed as version 6.0.2 or older (even after the upgrade process). You must manually add an "admin\_secure" line to the file, set to "true" as shown above.

Save your change and close the file.

3. Still on your Puppet master node, change into the [installation staging directory](#) and 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 Puppet 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 main menu select "Cluster Management", and then select "Push Configuration Settings to Cluster". Follow the prompts to trigger a Puppet push out to the cluster.
5. Return to the "Cluster Management" menu, then select "Manage Services". Select the S3 Service, then enter "restart". This automatically restarts the Admin Service as well as the S3 Service.
6. From the same menu, restart your CMC service. The CMC needs to be restarted so that it can update its configuration settings and start using exclusively HTTPS to communicate with the Admin Service.

**Note** You do not need to take any action in regard to an SSL certificate for the Admin Service (for HTTPS support). A self-signed certificate -- unique to your system -- is generated automatically during HyperStore installation and is used by the Admin Service.

### 12.1.7. HTTP/S Basic Authentication for Admin API Access

The Admin Service requires that clients use HTTP/S Basic Authentication credentials (user name and password) when connecting to the service. By default the required user name for this purpose is "sysadmin" and the default password is either a randomly generated password unique to your system (if your original HyperStore install was version 7.2.2 or newer) or "public" (if your original HyperStore install was older than version 7.2.2). To check to see what the current password is, in [common.csv](#) check the value of the *admin\_auth\_pass* setting which shows both a Jetty-obfuscated version of the password and the clear text version of the password.

**If the Admin API HTTP/S Basic Authentication password in your system is "public", you should change it to something more secure.** Even if the password is a random one generated upon system install, you may still

wish to change it to a password of your own creation. The procedure below describes how to do so. Optionally you can also change the user name, as also described below.

**Note** If your original HyperStore install was older than version 6.0.2, the Admin Service by default does not require Basic Authentication. The procedure below includes instructions for enabling the Basic Authentication requirement, if it is not already enabled in your system.

1. To change the Admin Service HTTP/S Basic Authentication password, start by logging into the Puppet master node and using the Jetty password tool that's included in your HyperStore package to generate a Jetty-obfuscated version of your desired password. The following example runs the tool to generate a Jetty-obfuscated version of the password "test1234":

```
# cd /opt/cloudian/lib
# java -cp jetty-util* org.eclipse.jetty.util.security.Password test1234
test1234
OBF:1mf31j8x1lts1ltu1lq41lq61j651mbj
MD5:16d7a4fca7442dda3ad93c9a726597e4
```

After running the tool, copy or make a note of the "OBF" (Jetty-obfuscated) version of the new password; you will need to supply it in later steps of this procedure. The "OBF" prefix is not a part of the password -- in the example above the obfuscated password starts with *1mf31*...

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

If you are using the [HyperStore Shell](#), the Jetty password tool referenced above is not available from the shell. Instead you can obtain an OBF version of your new password by running the *jetty\_password.sh* script as shown in the example below (replace *newpassword* with your new password).

```
sa_admin@node1$ jetty_password.sh newpassword
2019-09-08 11:59:41.520:INFO::main: Logging initialized @217ms
newpassword
OBF:luo91vn61ymf1yt41v1plym71v2plytilylz1vnwlunp
MD5:5e9d11a14ad1c8dd77e98ef9b53fd1ba
```

2. Still on your Puppet master node, open this configuration file in a text editor:

```
/etc/cloudian-7.2.3-puppet/manifests/extdata/common.csv
```

3. In *common.csv* edit these settings, then save and close the file:
  - *admin\_auth\_user*: Set to the user name you want to use for HTTP/S Basic Authentication for the Admin Service (or just leave this at the default which is "sysadmin").
  - *admin\_auth\_pass*: Set to a quote-enclosed comma-separated pair: "<Jetty\_obfuscated\_password>, <cleartext\_password>". The obfuscated version is what you generated in Step 1, and the clear text version is the plain password without obfuscation. For example, "1mf31j8x1lts1ltu1lq41lq61j651mbj,test1234".
  - *admin\_auth\_enabled*: Set to *true*, if it's not already set to *true* (it will be *true* by default if your original HyperStore install was 6.0.2 or newer).

**Note** Leave the *admin\_auth\_realm* setting at its default of "CloudianAdmin"

4. Still on your Puppet master node, use the installer to:
  - a. Push your changes to the cluster.

- b. Restart the S3 Service (doing so will automatically restart the Admin Service as well).
- c. Restart the CMC (the CMC needs to be restarted so that it can update its configuration settings for using Basic Authentication when communicating with the Admin Service).

If you need more detailed instructions for this step see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

### 12.1.8. Admin API Logging

Admin API transactions are logged in the Admin Service application log, for which the default location is `/var/log/cloudian/cloudian-admin.log`. API transactions are logged at log level INFO. As with all Admin Service application log entries, the format for API transaction entries is:

```
Date(ISO8601) PriorityLevel [ThreadId] ClassName:MethodName(Line#) MESSAGE
```

In the following example, first a request to retrieve the current list of user groups in the system is successfully processed. Then a request is received which is asking to retrieve a specific group that doesn't actually exist in the system (perhaps the requestor made a typo in the group ID). That request results in a 204 HTTP error response from the system. Note that each API transaction is identified by an integer within the MESSAGE element — "7" for the first transaction in the example and "8" for the second transaction.

```
2016-01-02 09:45:07,841 INFO [qtp21028611-57] LoggingFilter:log(153) 7 * Server has
received a request on thread qtp21028611-57
7 > GET http://192.168.2.16:18081/group/list
7 > Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
7 > Accept-Encoding: gzip, deflate
7 > Accept-Language: null
7 > Connection: keep-alive
7 > DNT: 1
7 > Host: 192.168.2.16:18081
7 > User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:43.0) Gecko/20100101 Firefox/43.0

2016-01-02 09:45:08,607 INFO [qtp21028611-57] LoggingFilter:log(153) 7 * Server
responded with a response on thread qtp21028611-57
7 < 200
7 < Content-Type: application/json
2016-01-02 09:48:17,596 INFO [qtp21028611-59] LoggingFilter:log(153) 8 * Server
has received a request on thread qtp21028611-59
8 > GET http://192.168.2.16:18081/group?groupId=SalesGroup
8 > Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
8 > Accept-Encoding: gzip, deflate
8 > Accept-Language: null
8 > Connection: keep-alive
8 > DNT: 1
8 > Host: 192.168.2.16:18081
8 > User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:43.0) Gecko/20100101 Firefox/43.0

2016-01-02 09:48:17,621 INFO [qtp21028611-59] GroupResource:getGroup(88) Group Id
not found:SalesGroup

2016-01-02 09:48:17,623 INFO [qtp21028611-59] LoggingFilter:log(153) 8 * Server
responded with a response on thread qtp21028611-59
8 < 204
```

## 12.2. billing

The Admin API methods built around the **billing** resource are for generating or retrieving a billable activity report for a specified user or group. The report shows the user or group's billable activity and the charges for that activity based on the assigned [rating plan\(s\)](#).

For an overview of the HyperStore billing feature, see **"Usage Reporting and Billing Feature Overview"** (page 138).

### 12.2.1. GET /billing

#### GET /billing    Get a bill for a user or group

The request line syntax for this method is as follows.

```
GET /billing? [userId=xxx&] [groupId=xxx] [canonicalUserId=xxx] &billingPeriod=xxx
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"billing Query Parameters"** (page 755).

There is no request payload.

This method retrieves an **existing** bill for a user or group (a bill that has already been generated by the [POST /billing](#) method.) You must use the [POST /billing](#) method for the user or group and billing period of interest before you can use this method.

**Note** The HyperStore IAM Service supports a role-based access version of this API method as described in **"RBAC Version of this Method"** (page 752).

#### 12.2.1.0.1. Example Using cURL

The [example](#) below generates a billable activity report for the user "glad" from the "eng" group, for the month of July 2017. This is an existing billable activity report that has previously been generated by the [POST /billing](#) method.

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/billing?userId=glad&groupId=eng&billingPeriod=201707' \
| python -mjson.tool
```

The response payload is a JSON-formatted *Bill* object, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"Bill Object"** (page 756).

```
{
  "billId": "936265a2-fbd5-47c2-82ed-d62298299a1b",
  "canonicalUserId": "d47151635ba8d94efe981b24db00c07e",
  "currency": "USD",
  "endCal": 1501545599000,
  "groupId": "eng",
  "notes": null,
  "regionBills": [
    {
      "currency": "USD",
      "items": {
```

```

    "SB": {
      "item": "SB",
      "quantity": 108.00,
      "rules": "1,0.14:5,0.12:0,0.10",
      "subtotal": 10.94
    },
    "region": "taoyuan",
    "total": 10.94,
    "whitelistItems": {},
    "whitelistTotal": 0
  },
  "startCal": 1498867200000,
  "total": 10.94,
  "userId": "glad",
  "whitelistTotal": 0
}

```

### 12.2.1.0.2. Response Format

The response payload is a JSON-formatted *Bill* object (see example above). For response status code this method will return one of the [Common Status Codes](#) or one of these method-specific status codes:

Status Code	Description
204	Billing data does not exist
400	User does not exist
400	Missing required parameter : {billingPeriod}
400	Conflicting parameters: {canonicalUserId, groupId, userId}

#### 12.2.1.1. RBAC Version of this Method

**IMPORTANT !** Before the RBAC version of this method can be used to retrieve billing data for a specified user and billing period, you must either execute the Admin API method [POST /billing](#) to generate billing data for that user and billing period, or else use the CMC's [Account Activity](#) page to generate billing data for that user and billing period. There is currently no RBAC version of the *POST /billing* call that generates user billing data.

HyperStore's IAM Service supports an RBAC version of this API method. For an overview of the RBAC feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- Action name: *GetCloudianBill*
- Parameters: Same as for *GET /billing*, except:
  - *userId* and *groupId* are not supported. A user can only be specified by canonical ID, and retrieving a bill for a whole group is not supported.
  - All parameter names start with an upper case letter rather than lower case

- Response body: Same response data as for *GET /billing* except the data is formatted in XML rather than JSON
- Role-based restrictions:
  - HyperStore system admin user can get a bill for any user
  - HyperStore group admin user can only get bills for users within her group
  - HyperStore regular user can only get own bill
  - IAM user can only use this method if granted *admin:GetCloudianBill* permission by an IAM policy, and subject to the same restriction as the parent HyperStore user.

**Note** The "GetCloudianBill" action retrieves billing data for Cloudian HyperStore user accounts, not for subsidiary IAM users. The system does not maintain billing data per IAM user. For example, if a HyperStore group administrator grants *admin:GetCloudianBill* permission to an IAM user, the IAM user will be able to retrieve billing information for **any HyperStore user in the group administrator's group**. And if a HyperStore regular user grants *admin:GetCloudianBill* permission to an IAM user, the IAM user will be able to retrieve billing information for the **parent HyperStore user**.

- Sample request and response (abridged):

```
REQUEST

http://
localhost:16080/?Action=GetCloudianBill&CanonicalUserId=d47151635ba8d94efe981b24db00c07e
&BillingPeriod=201807

<request headers including authorization info>

RESPONSE

200 OK

<?xml version="1.0" encoding="utf-8"?>
<GetCloudianBillResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">
  <ResponseMetadata>
    <RequestId>system-generated-request-id</RequestId>
  </ResponseMetadata>
  <Bill>
    <billID>936265a2-fbd5-47c2-82ed-d62298299a1b</billID>
    etc...
    ...
    ...
  </Bill>
</GetCloudianBillResponse>
```

### 12.2.2. POST /billing

#### POST /billing Create a bill for a user or group

The request line syntax for this method is as follows.

```
POST /billing?[userId=xxx&][groupId=xxx][canonicalUserId=xxx]&billingPeriod=xxx
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see "[billing Query Parameters](#)" (page 755).

There is no request payload.

This method generates a user's monthly bill or a whole group's monthly bill, and returns the bill in the response body. The billing period **must be a month that has already completed**. You cannot generate a bill for the current, in-progress month.

**IMPORTANT !** Billing calculation is derived from hourly rollup usage data. The retention period for hourly rollup usage data is configured by *mts.properties.erb*: "[reports.rolluphour.ttl](#)" (page 567). The default retention period is 65 days. Once this rollup data is deleted it can no longer be used to generate users' bills.

### 12.2.2.0.1. Example Using cURL

The [example](#) below generates a billable activity report for the user "glad" from the "eng" group, for the month of July 2017.

```
curl -X POST -k -u sysadmin:public \
'https://localhost:19443/billing?userId=glad&groupId=eng&billingPeriod=201707' \
| python -mjson.tool
```

The response payload is a JSON-formatted *Bill* object, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see "[Bill Object](#)" (page 756).

```
{
  "billId": "936265a2-fbd5-47c2-82ed-d62298299a1b",
  "canonicalUserId": "d47151635ba8d94efe981b24db00c07e",
  "currency": "USD",
  "endCal": 1501545599000,
  "groupId": "eng",
  "notes": null,
  "regionBills": [
    {
      "currency": "USD",
      "items": {
        "SB": {
          "item": "SB",
          "quantity": 108.00,
          "rules": "1,0.14:5,0.12:0,0.10",
          "subtotal": 10.94
        }
      },
      "region": "taoyuan",
      "total": 10.94,
      "whitelistItems": {},
      "whitelistTotal": 0
    }
  ],
  "startCal": 1498867200000,
```

```

"total": 10.94,
"userId": "glad",
"whitelistTotal": 0
}

```

### 12.2.2.0.2. Response Format

The response payload is a JSON-formatted *Bill* object (see example above). For response status code this method will return one of the [Common Status Codes](#) or one of these method-specific status codes:

Status Code	Description
204	No billing data
400	Invalid billing period
400	Missing required parameter : {billingPeriod}
400	Conflicting parameters: {canonicalUserId, groupId, userId}

## 12.2.3. billing Query Parameters

*userId, groupId, canonicalUserId*

(Optional, strings) Identifiers of the user or group for which to generate or retrieve a bill.

- To generate or retrieve a bill for a **user who currently is part of the service**, you can either use the "userId" parameter in combination with the "groupId" parameter (for example *userId=martinez@groupId=operations*), or use the "canonicalUserId" parameter by itself (with no "groupId" parameter).
- To generate or retrieve a bill for a **user who has been deleted from the service**, you must use the "canonicalUserId" parameter by itself (not the "userId" or "groupId" parameter).

**Note** If you don't know the user's system-generated canonical ID, you can obtain it by using the *GET /user/list* method.

- To generate or retrieve a bill for a **whole user group**, use the "groupId" parameter by itself (not the "userId" or "canonicalUserId" parameter). Note that if you generate a bill for a whole group, the bill will be based on the rating plan assigned to the group as a whole, and will not take into account any different rating plans that administrators may have assigned to specific users within the group.

*billingPeriod*

(Mandatory, string) Specifies the year and month of bill. Format is *yyyyMM* — for example "201708" for August 2017. Note that the system uses GMT time when demarcating exactly when a month begins and ends.

## 12.2.4. billing Objects

This section describes the JSON-formatted data objects that constitute the request or response payloads for the Billing related Admin API methods.

**Note** For examples of this object see the API method request and response examples.

- **"Bill Object"** (page 756)

### 12.2.4.1. Bill Object

The *Bill* object consists of the following attributes and nested objects:

*billID*

(String) System-generated globally unique bill ID. Example:

```
"billID": "936265a2-fbd5-47c2-82ed-d62298299a1b"
```

*canonicalUserId*

(String) System-generated canonical user ID for the user. Empty if the bill is for a whole group. Example:

```
"canonicalUserId": "d47151635ba8d94efe981b24db00c07e"
```

*currency*

(String) Currency string. Example:

```
"currency": "USD"
```

*endCal*

(String) End date/time of the billing period in UTC milliseconds. Example:

```
"endCal": 1501545599000
```

*groupId*

(String) ID of the group to which the user belongs (or of the group for which the bill was generated, in the case of a whole group bill). Example:

```
"groupId": "eng"
```

*notes*

(String) Notes regarding the bill, if any. Example:

```
"notes": null
```

*regionBills*

(Map<string,RegionBill>) List of *RegionBill* objects, with one such object per service region. The *RegionBill* object consists of the following attributes and nested objects:

*currency*

(String) Currency string. Example:

```
"currency": "USD"
```

*items*

(Map<string,BillItem>) List of *BillItem* objects, with one such object for each activity type that's being charged for, per the terms of the user's rating plan. Supported activity types are "SB" (storage bytes), "BI" (bytes in), "BO" (bytes out), "HG" (HTTP Gets), "HP" (HTTP Puts), "HD" (HTTP

Deletes). This list excludes activity for whitelisted IP addresses. Note that some or even most activity types may not appear, depending on the rating plan terms. For example, it may be that only storage bytes ("SB") are billed for, if that's how the user's rating plan is configured.

In the items list, each *BillItem* object is preceded by its activity type string, such as "SB": {*BillItem* data}. In the example only storage bytes ("SB") are charged for in the rating plan that was applied when this bill was generated.

The *BillItem* object consists of the following attributes:

#### *item*

(String) Usage type being billed for. Types are "SB" (storage bytes), "BI" (bytes in), "BO" (bytes out), "HG" (HTTP GETs), "HP" (HTTP PUTs), "HD" (HTTP DELETEs). Example:

```
"item": "SB"
```

#### *quantity*

(Number) Usage quantity during billing period. Usage quantity metrics depend on the usage type:

- For storage bytes (SB), the metric is GB-Month (average number of GBs of data stored for the billing month). This is calculated by summing the month's hourly readings of stored bytes, converting to GB, then dividing by the number of hours in the month. In the example above the usage quantity during the billing period was 108 GB-months (that is, the user's storage bytes volume average 108GBs over the course of the month)
- For data transfer bytes in (BI) or out (BO), the metric is number of bytes.
- For HTTP GETs (HG), PUTs (HP), or DELETEs (HD), the metric is number of multiples of 10,000 requests. For example, if usage type is HG and quantity is 7.50, that means 75,000 HTTP GET requests.

Example:

```
"quantity": 108.00
```

#### *rules*

(String) Specification of billing rules for this usage type (as configured in the user's assigned rating plan). In the example the "rules" attribute indicates that the user's rating plan is such that the first 1 GB-month is charged at \$0.14, the next 5 GB-months is charged at \$0.12 per GB-month, and all GB-months above that are charged at \$0.10 per GB-month.

Example:

```
"rules": "1,0.14:5,0.12:0,0.10"
```

#### *subtotal*

(Number) Total billing charge for the particular usage type specified by the "item" attribute. This will be in units of the currency specified by the "currency" attribute of the *RegionBill* object that contains this *BillItem* object. It's labeled as "subtotal" because it will be added together with subtotals for other usage types (from other *BillItem* object instances within the *RegionBill* object, if any) to compute the "total" attribute for the encompassing *RegionBill* instance. In the example the \$10.94 sub-total comes from applying the billing

rules to the 108 GB-months usage quantity  $([1 \times .14] + [5 \times .12] + [102 \times .10] = 10.94)$ .

Example:

```
"subtotal":10.94
```

#### *region*

(String) Region name. Example:

```
"region": "taoyuan"
```

#### *total*

(Number) For the region, the total charges incurred — excluding activity originating from whitelisted source IP addresses. Example:

```
"total": 10.94
```

#### *whitelistItems*

(Map<string, *BillItem*>) List of *BillItem* objects, for activity originating from whitelisted IP addresses (if any). Types are "BI" (bytes in), "BO" (bytes out), "HG" (HTTP Gets), "HP" (HTTP Puts), "HD" (HTTP Deletes). Example:

```
"whitelistItems": {}
```

#### *whitelistTotal*

(Number) For the region, the total charges incurred for activity originating from whitelisted source IP addresses. Typically there are no charges for such activity. Example:

```
"whitelistTotal": 0
```

#### *startCal*

(String) Start date/time of the billing period in UTC milliseconds. Example:

```
"startCal": 1498867200000
```

#### *total*

(Number) The total charges incurred by the user during the billing period, excluding activity for whitelisted source IP addresses. Example:

```
"total": 10.94
```

#### *userId*

(String) ID of the user for whom the bill was generated. Empty if the bill is for a whole group. Example:

```
"userId": "glad"
```

#### *whitelistTotal*

(Number) The total charges incurred by the user for activity originating from whitelisted source IP addresses. Example:

```
"whitelistTotal": 0
```

## 12.3. bppolicy

The Admin API methods built around the **bppolicy** resource are for retrieving certain information about HyperStore storage policies (also known as bucket protection policies).

For an overview of the HyperStore storage policy feature, see "**Storage Policies Feature Overview**" (page 76). To create or change storage policies use the CMC's [Storage Policies](#) page.

### 12.3.1. GET /bppolicy/bucketsperpolicy

#### GET /bppolicy/bucketsperpolicy    Get list of buckets using each storage policy

The request line syntax for this method is as follows.

```
GET /bppolicy/bucketsperpolicy
```

There is no request payload.

**Note** If you have a storage policy in your system that was created prior to the release of HyperStore version 5.2 (when support for multiple storage policies was introduced), the *GET /bppolicy/bucketsperpolicy* method does not work for listing buckets that use that storage policy. This is true even for buckets that were created after HyperStore version 5.2, if the buckets use that legacy storage policy. In the *GET /bppolicy/bucketsperpolicy* response, the policy ID for such a legacy storage policy will be *DEFAULT\_<regionName>* and the bucket list for the storage policy will be empty.

#### 12.3.1.0.1. Example Using cURL

The [example](#) below retrieves the list of buckets using each storage policy.

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/bppolicy/bucketsperpolicy | python -mjson.tool
```

The response payload is a JSON-formatted list of *BucketsInPolicy* objects, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see "**BucketsInPolicy Object**" (page 762).

```
[
  {
    "buckets": [
      "qa.tests",
      "dev.specs"
    ],
    "policyId": "b06c5f9213ae396de1a80ee264092b56",
    "policyName": "Replication-3X"
  },
  {
    "buckets": [
      "release.packages.archive",
      "techpubs.manuals.archive"
    ],
    "policyId": "af37905a8523d8d403d993c4f2e2c1a1",
    "policyName": "EC-4-2"
  }
]
```

```
}  
]
```

#### 12.3.1.0.2. Response Format

The response payload is a JSON-formatted list of *BucketsInPolicy* objects (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746).

### 12.3.2. GET /bppolicy/listpolicy

#### GET /bppolicy/listpolicy    Get list of storage policy IDs

The request line syntax for this method is as follows.

```
GET /bppolicy/listpolicy[?<a href="#">region=xxx</a>] [<a href="#">groupId=xxx</a>] [<a href="#">status=xxx</a>]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"bppolicy Query Parameters"** (page 761).

**Note** If you use more than one of the three optional filters -- *region*, *groupId*, and *status* -- then the returned list of storage policy IDs will be for storage policies that match all of your specified filters. For example if you specify a *region* and a *groupId*, then the returned list will consist only of policies that are both associated with that region and available to that group.

There is no request payload.

Use this method if you want to retrieve the system-generated policy IDs associated with each of your storage policies.

#### 12.3.2.0.1. Example Using cURL

The [example](#) below retrieves the list of all storage policies currently in the system.

```
curl -X GET -k -u sysadmin:public \  
https://localhost:19443/bppolicy/listpolicy | python -mjson.tool
```

The response payload is a JSON-formatted list of *BucketProtectionPolicy* objects, with one such object for each storage policy. Among the attributes for each policy is the "policyName" and "policyId". In the example that follows there are two storage policies in the system, and the response payload is truncated so as to show only the policy ID and policy name attributes.

```
[  
  {  
    ...  
    ...  
    "policyId": "b06c5f9213ae396de1a80ee264092b56",  
    "policyName": "Replication-3X",  
    ...  
    ...  
  },  
  {  
    ...  
    ...  
  }  
]
```

```

    "policyId": "af37905a8523d8d403d993c4f2e2c1a1",
    "policyName": "EC-4-2",
    ...
    ...
  }
]

```

### 12.3.2.0.2. Response Format

The response payload is a JSON-formatted list of *BucketProtectionPolicy* objects (see excerpt above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746).

## 12.3.3. bppolicy Query Parameters

### *region*

(Optional, string) If you use this parameter, then only policies associated with the specified service region will be returned.

### *groupId*

(Optional, string) If you use this parameter, then only policies that are available to the specified group will be returned. This includes system default storage policies (which are available to all groups) as well as storage policies that are explicitly made available to the specified group.

### *status*

(Optional, string) If you use this parameter, then only policies that have the specified status will be returned. The supported statuses are:

- *pending* — The policy is in the process of being created in the system. In this state the policy is not yet available to be used.
- *active* — The policy is currently available to users when they create a new bucket.
- *disabled* — The policy is no longer available to users when they create a new bucket. However, the policy still exists in the system and is still being applied to any buckets to which the policy was assigned during the period when it was active.
- *deleted* — The policy has been marked for deletion and is no longer available to users. However, the policy has not yet been purged from the system by the daily cron job.
- *failed* — During the policy creation, the policy failed to be fully set up in the system. Though a *BucketProtectionPolicy* JSON object exists and can be retrieved, the policy is not actually set up in the system and is not usable.

## 12.3.4. bppolicy Objects

This section describes the JSON-formatted data objects that constitute the request or response payloads for the bucket protection policy related Admin API methods.

**Note** For examples of this object see the API method request and response examples.

- **"BucketsInPolicy Object"** (page 762)

### 12.3.4.1. BucketsInPolicy Object

The *BucketsInPolicy* object consists of the following attributes:

*buckets*

(List<string>) List of buckets that use the storage policy. Example:

```
"buckets": ["qa.tests", "dev.specs"]
```

*policyId*

(String) System-generated unique identifier of the storage policy. Example:

```
"policyId": "b06c5f9213ae396de1a80ee264092b56"
```

*policyName*

(String) Storage policy name. Example:

```
"policyName": "Replication-3X"
```

## 12.4. bucketops

### 12.4.1. GET /bucketops/id

**GET /bucketops/id**    Get a bucket's canonical ID

The request line syntax for this method is as follows.

```
GET /bucketops/id?bucketName=xxx
```

For parameter description click on the parameter name or see "**bucketops Query Parameters**" (page 765).

There is no request payload.

This operation returns a bucket's canonical ID, if one exists. A bucket will have a canonical ID (a system-generated unique identifier) if either of the following applies:

- The bucket was created in HyperStore 7.0 or later.
- The bucket has been subjected to a successful *POST /bucketops/purge* operation.

After a successful *POST /bucketops/purge* operation a bucket will have a different canonical ID than the one it had before (if it had any) but will have the same bucket name.

#### 12.4.1.0.1. Example Using cURL

The [example](#) below retrieves the canonical ID of a bucket named "bucket1".

```
curl -X GET -k -u sysadmin:public \  
https://localhost:19443/bucketops/id?bucketName=bucket1
```

The response payload is the bucket's canonical ID in plain text, which in this example is as follows:

```
40cc2eba37fd82df4ce04bce2bc35a94
```

### 12.4.1.0.2. Response Format

The response payload is a plain text string (see example above). For response status code this method will return either one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameter : {bucketName}

## 12.4.2. GET /bucketops/gettags

### GET /bucketops/gettags    Get bucket tags for users in a group

The request line syntax for this method is as follows.

```
GET /bucketops/gettags?groupId=xxx[&limit=xxx] [&userId=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"bucketops Query Parameters"** (page 765).

There is no request payload.

For each user in the specified group this operation returns the bucket tags associated with the user's buckets (after such tags have been created by the S3 API method [PUT Bucket tagging](#)). Pagination of the response is supported by use of the optional *limit* and *userId* settings. By default a maximum of 10 users is returned per request.

#### Note

- \* **The *userId* parameter is to be used only for pagination.** You cannot use this parameter to retrieve bucket tags for just one user of your choosing.
- \* **Only buckets that have bucket tags are listed in the response.** Buckets that do not have bucket tags are excluded from the response.

### 12.4.2.0.1. Example Using cURL

The [example](#) below returns the bucket tags for the buckets owned by users in the group "Cloudian".

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/bucketops/gettags?groupId=Cloudian \
| python -mjson.tool
```

The response payload is a JSON-formatted *BucketTags* object, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"BucketTags"** (page 766).

```
{
  "groupId": "Cloudian",
  "nextUserId": null,
  "userBucket": {
    "kthompson": {
      "bbucket": { "Project": "Project1", "Manager": "jsmith" },
      "cbucket": { "security": "public" },
    }
  }
}
```

```
"gwashtington":
  {"dbucket":{"security":"public"}}
}
```

**Note** Only buckets that have bucket tags are listed in the response. Buckets that do not have bucket tags are excluded from the response.

```
}
```

#### 12.4.2.0.2. Response Format

The response payload is a JSON-formatted *BucketTags* object (see example above). For response status code this method will return either one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameter : {groupId}

### 12.4.3. POST /bucketops/purge

#### POST /bucketops/purge Delete all the objects in a bucket

The request line syntax for this method is as follows.

```
POST /bucketops/purge?bucketName=xxx
```

For parameter description click on the parameter name or see **"bucketops Query Parameters"** (page 765).

There is no request payload.

This operation results in the system marking all the objects in the bucket 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 **"cloud-ian.delete.queue.poll.interval"** (page 566) property in [mts.properties.erb](#).)

The *POST /bucketops/purge* operation does not invoke the S3 *DELETE Object* API and does not create the Cassandra tombstone issues that can sometimes be caused by mass delete operations that use HyperStore's S3 interface.

The bucket itself continues to exist after this operation. If you run an S3 *GET Bucket (List Objects)* call on the bucket -- or get the bucket in the CMC -- after successfully calling the *POST /bucketops/purge* operation, the *GET Bucket* response will indicate that the bucket is empty even though the actual deletion of objects may not have been completed by the cron job yet. Any objects that you upload at this point -- **after** you've successfully called *POST /bucketops/purge* -- will not be deleted by the batch processing.

Note that:

- Any S3 multipart upload operations in-progress for the bucket at the time that you execute the *POST /bucketops/purge* operation will be aborted.
- If you have [versioning](#) configured on the bucket, the *POST /bucketops/purge* operation will purge all versions of all objects in the bucket.

- If you have [auto-tiering](#) configured on the bucket, any objects that have been tiered from the bucket to the remote tiering destination will also be deleted (at the next running of the hourly system cron job mentioned above).
- The *POST /bucketops/purge* operation is not allowed on buckets that have [Object Lock](#) enabled.
- In a multi-region system, the *POST /bucketops/purge* call should be submitted to the Admin API service in the default region regardless of which region the target bucket is in.

#### 12.4.3.0.1. Example Using cURL

The [example](#) below purges the contents of a bucket named "bucket1".

```
curl -X POST -k -u sysadmin:public \
https://localhost:19443/bucketops/purge?bucketName=bucket1
```

The response indicates that the bucket contents have been successfully purged (marked for deletion):

```
Bucket: bucket1 purged.
```

#### 12.4.3.0.2. Response Format

The response payload is a plain text string (see example above). For response status code this method will return either one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameter : {bucketName}
403	Can't delete locked bucket

### 12.4.4. bucketops Query Parameters

*bucketName*

(Mandatory, string) Name of the bucket.

*groupId*

(Mandatory, string) For a *GET /bucketops/gettags* operation, the group for which to retrieve a list of users and their bucket tags.

*limit*

(Optional, integer) For a *GET /bucketops/gettags* operation, the maximum number of users to return (along with those users' bucket tags) per operation.

Users are retrieved in alphanumeric order. If the number of users in the group exceeds the number specified by *limit*, in the response to the first *GET /bucketops/gettags* operation the *nextUserId* attribute will indicate the user ID of the next available user in the alphanumeric ordering (the alphanumerically first of the users that has not yet been returned). That user ID can then be used as the *userId* parameter value in a subsequent *GET /bucketops/gettags* operation. That operation will again return up to the number of users specified by *limit*, and if there are more remaining users beyond that, the return will again use the *nextUserId* attribute to indicate the next available user's ID; and so on.

Admin API client applications can use the *limit* and *userId* parameters in combination to support pagination of results.

Defaults to 10. Maximum allowed value for *limit* is 100.

*userId*

(Optional, string) For a *GET /bucketops/gettags* operation, the alphanumerically first user to retrieve. See the description of *limit* above for more detail about how the *userId* and *limit* parameters can be used to support pagination.

In the first *GET /bucketops/gettags* request for a group the client should omit the *userId* parameter. If *userId* is omitted from the request, the operation's returned list of users starts with the alphanumerically first user in the group.

**Note** The *userId* parameter is to be used only for pagination. You cannot use this parameter to retrieve bucket tags for just one user of your choosing.

### 12.4.5. bucketops Objects

This section describes the JSON-formatted data objects that constitute the request or response payloads for the *bucketops* related Admin API methods.

**Note** For examples of these objects see the API method request and response examples.

- **"BucketTags"** (page 766)

#### 12.4.5.1. BucketTags

The *BucketTags* object consists of the following attributes:

*groupId*

(String ) The group for which users and bucket tags have been retrieved. Example:

```
"groupId": "Cloudian"
```

*nextUserId*

(String ) Users are retrieved in alphanumeric order. If the number of users in the group exceeds the number specified by the query parameter *limit* (which defaults to 10), in the *GET /bucketops/gettags* response the *nextUserId* attribute will indicate the user ID of the next available user in the alphanumeric ordering (the alphanumerically first of the users that has not yet been returned). That user ID can then be used as the *userId* query parameter value in a subsequent *GET /bucketops/gettags* operation. That operation will again return up to the number of users specified by *limit*; and if there are more remaining users beyond that, the return will again use the *nextUserId* attribute to indicate the next available user's ID.

If alphanumerically there are no additional users beyond those returned in the current response, the *nextUserId* attribute value will be null.

Example:

```
"nextUserId": null
```

*userBucket*

(Map<String, Map<String, Map<String, String>>>) This entity contains the map of users, their owned buckets

that have bucket tags, and the bucket tags. The format is as follows:

```
{ "userId": { "bucketName": { "tagName": "tagValue" }, { "tagName2": "tagValue2" } ... }, { "bucketName2": { "tagName": "tagValue" }, { "tagName2": "tagValue2" } ... }, ... "userId2": { "bucketName": ... } }
```

Example:

```
"userBucket":
{
  "kthompson":
  {
    "bbucket": { "Project": "Project1", "Manager": "jsmith" },
    "cbucket": { "security": "public" },
    "gwashtington":
    {
      "dbucket": { "security": "public" }
    }
  }
}
```

**Note** Only buckets that have bucket tags are listed in the response. Buckets that do not have bucket tags are excluded from the response.

## 12.5. group

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.

### 12.5.1. DELETE /group

**DELETE /group** Delete a group

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

The request line syntax for this method is as follows.

```
DELETE /group?groupId=xxx
```

For parameter description click on the parameter name or see **"group Query Parameters"** (page 776).

There is no request payload.

#### 12.5.1.0.1. Example Using cURL

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

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

#### 12.5.1.0.2. Response Format

There is no response payload. For response status code this method will return either one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameters : {groupId}

Status Code	Description
400	Group does not exist
409	Cannot delete. Group is not empty.

## 12.5.2. GET /group

### GET /group Get a group's profile

The request line syntax for this method is as follows.

```
GET /group?groupId=xxx
```

For parameter description click on the parameter name or see **"group Query Parameters"** (page 776).

There is no request payload.

**Note** The HyperStore IAM Service supports a role-based access version of this API method as described in **"RBAC Version of this Method"** (page 769).

#### 12.5.2.0.1. Example Using cURL

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

```
curl -X GET -k -u sysadmin:public https://localhost:19443/group?groupId=QA \
| python -mjson.tool
```

The response payload is a JSON-formatted *GroupInfo* object, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"GroupInfo Object"** (page 777).

```
{
  "active": "true",
  "groupId": "QA",
  "groupName": "Quality Assurance Group",
  "ldapEnabled": false,
  "ldapGroup": "",
  "ldapMatchAttribute": "",
  "ldapSearch": "",
  "ldapSearchUserBase": "",
  "ldapServerURL": "",
  "ldapUserDNTemplate": "",
  "s3endpointshttp": ["ALL"],
  "s3endpointshttps": ["ALL"],
  "s3websiteendpoints": ["ALL"]
}
```

#### 12.5.2.0.2. Response Format

The response payload is a JSON-formatted *GroupInfo* object (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

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

### 12.5.2.1. RBAC Version of this Method

HyperStore's IAM Service supports an RBAC version of this API method. For an overview of the RBAC feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- Action name: *GetCloudianGroup*
- Parameters: Same as for *GET /group*, except all parameter names start with an upper case letter rather than lower case
- Response body: Same response data as for *GET /group* except the data is formatted in XML rather than JSON
- Role-based restrictions:
  - HyperStore system admin user can get any group
  - HyperStore group admin user can only get his own group
  - HyperStore regular user cannot use this method
  - IAM user can only use this method if granted *admin:GetCloudianGroup* permission by policy, and subject to the same restriction as the parent HyperStore user

**Note** The "GetCloudianGroup" action retrieves group profile data for Cloudian HyperStore groups, not for HyperStore users' subsidiary IAM groups.

- Sample request and response (abridged):

```
REQUEST

http://localhost:16080/?Action=GetCloudianGroup&GroupId=QA

<request headers including authorization info>

RESPONSE

200 OK

<?xml version="1.0" encoding="utf-8"?>
<GetCloudianGroupResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">
  <ResponseMetadata>
    <RequestId>system-generated-request-id</RequestId>
  </ResponseMetadata>
  <GroupInfo>
    <active>true</active>
    etc...
    ...
  </GroupInfo>
</GetCloudianGroupResponse>
```

### 12.5.3. GET /group/list

#### GET /group/list    Get a list of group profiles

The request line syntax for this method is as follows.

```
GET /group/list[?<u>prefix</u>=xxx] [&<u>limit</u>=xxx] [&<u>offset</u>=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"group Query Parameters"** (page 776).

There is no request payload.

**Note** The HyperStore IAM Service supports a role-based access version of this API method as described in **"RBAC Version of this Method"** (page 771).

#### 12.5.3.0.1. Example Using cURL

The [example](#) below retrieves the profiles of all groups currently in the system.

```
curl -X GET -k -u sysadmin:public https://localhost:19443/group/list \  
| python -mjson.tool
```

The response payload is a JSON-formatted list of *GroupInfo* objects, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"GroupInfo Object"** (page 777).

```
[  
  {  
    "<u>active</u>": "true",  
    "<u>groupId</u>": "QA",  
    "<u>groupName</u>": "Quality Assurance Group",  
    "<u>ldapEnabled</u>": false,  
    "<u>ldapGroup</u>": "",  
    "<u>ldapMatchAttribute</u>": "",  
    "<u>ldapSearch</u>": "",  
    "<u>ldapSearchUserBase</u>": "",  
    "<u>ldapServerURL</u>": "",  
    "<u>ldapUserDNTemplate</u>": "",  
    "<u>s3endpointshttp</u>": ["ALL"],  
    "<u>s3endpointshttps</u>": ["ALL"],  
    "<u>s3websiteendpoints</u>": ["ALL"]  
  },  
  {  
    "active": "true",  
    "groupId": "Support",  
    "groupName": "Technical Support Group",  
    "ldapEnabled": false,  
    "ldapGroup": "",  
    "ldapMatchAttribute": "",  
    "ldapSearch": "",  
    "ldapSearchUserBase": "",  
    "ldapServerURL": "",  
    "ldapUserDNTemplate": "",  
    "s3endpointshttp": ["ALL"],  
    "s3endpointshttps": ["ALL"],  
    "s3websiteendpoints": ["ALL"]  
  }  
]
```

```

    "ldapServerURL": "",
    "ldapUserDNTemplate": "",
    "s3endpointshttp": ["ALL"],
    "s3endpointshttps": ["ALL"],
    "s3websiteendpoints": ["ALL"]
  },
  {
    "active": "true",
    "groupId": "engineering",
    "groupName": "Engineering Group",
    "ldapEnabled": false,
    "ldapGroup": "",
    "ldapMatchAttribute": "",
    "ldapSearch": "",
    "ldapSearchUserBase": "",
    "ldapServerURL": "",
    "ldapUserDNTemplate": "",
    "s3endpointshttp": ["ALL"],
    "s3endpointshttps": ["ALL"],
    "s3websiteendpoints": ["ALL"]
  }
]

```

### 12.5.3.0.2. Response Format

The response payload is a JSON-formatted list of *GroupInfo* objects (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Limit should be greater than zero

### 12.5.3.1. RBAC Version of this Method

HyperStore's IAM Service supports an RBAC version of this API method. For an overview of the RBAC feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- Action name: *GetCloudianGroupList*
- Parameters: Same as for *GET /group/list*, except all parameter names start with an upper case letter rather than lower case
- Response body: Same response data as for *GET /group/list* except the data is formatted in XML rather than JSON
- Role-based restrictions:
  - HyperStore system admin user can use this method
  - HyperStore group admin user cannot use this method
  - HyperStore regular user cannot use this method
  - IAM user can only use this method if granted *admin:GetCloudianGroupList* permission by policy, and subject to the same restriction as the parent HyperStore user

**Note** The "GetCloudianGroupList" action retrieves a list of Cloudian HyperStore groups, not a list of HyperStore users' subsidiary IAM groups.

- Sample request and response (abridged):

```
REQUEST

http://localhost:16080/?Action=GetCloudianGroupList

<request headers including authorization info>

RESPONSE

200 OK

<?xml version="1.0" encoding="utf-8"?>
<GetCloudianGroupListResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">
  <ResponseMetadata>
    <RequestId>system-generated-request-id</RequestId>
  </ResponseMetadata>
  <ListWrapper>
    <groupInfo>
      <active>true</active>
      etc...
    ...
    ...
  </groupInfo>
  <groupInfo>
    etc...
    ...
    ...
  </groupInfo>
</ListWrapper>
</GetCloudianGroupListResponse>
```

#### 12.5.4. GET /group/ratingPlanId

##### GET /group/ratingPlanId    Get a group's rating plan ID

The request line syntax for this method is as follows.

```
GET /group/ratingPlanId?groupId=xxx[&region=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"group Query Parameters"** (page 776).

There is no request payload.

##### 12.5.4.0.1. Example Using cURL

The [example](#) below retrieves the ID of the rating plan assigned to the "QA" group.

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/group/ratingPlanId?groupId=QA
```

The response payload is the rating plan identifier in plain text, which in this example is as follows.

```
Default-RP
```

#### 12.5.4.0.2. Response Format

The response payload is a plain text string (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
204	Rating Plan does not exist
400	Missing Required parameters : {groupId}
400	Region {region} is not valid

### 12.5.5. POST /group

#### POST /group Change a group's profile

The request line syntax for this method is as follows.

```
POST /group
```

The required request payload is a JSON-formatted *GroupInfo* object.

#### 12.5.5.0.1. Example Using cURL

The [example](#) below modifies the group profile that was created in the [PUT /group](#) example. Again the *GroupInfo* object is specified in a text file named *group\_QA.txt* which is then referenced as the data input to the cURL command.

```
curl -X POST -H "Content-Type: application/json" -k -u sysadmin:public \
-d @group_QA.txt https://localhost:19443/group
```

Note that in editing the *GroupInfo* object in the *group\_QA.txt* file before doing the POST operation you could edit any attribute except for the "groupId" attribute. The "groupId" attribute must remain the same, so that you're modifying an existing group rather than creating a new one. For an example *GroupInfo* object see [PUT /group](#).

#### 12.5.5.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Group does not exist
400	Missing required attribute : {groupId}
400	Invalid Active Status for Post Group

Status Code	Description
400	Invalid JSON Object

### 12.5.5.1. 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 by editing the group's profile you must also perform this additional configuration step:

1. Log in to the Puppet Master 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:

```
hscctl config set hsh.ldap.bindDN=<bind Distinguished Name>
hscctl config set hsh.ldap.bindPassword=<bind password>
hscctl config apply hsh
```

### 12.5.6. POST /group/ratingPlanId

**POST /group/ratingPlanId** Assign a rating plan to a group

The request line syntax for this method is as follows.

```
POST /group/ratingPlanId?groupId=xxx&ratingPlanId=xxx[&region=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"group Query Parameters"** (page 776).

There is no request payload.

#### 12.5.6.0.1. Example Using cURL

The [example](#) below assigns the "Gold" rating plan to the "QA" group.

```
curl -X POST -k -u sysadmin:public \
'https://localhost:19443/group/ratingPlanId?groupId=QA&ratingPlanId=Gold'
```

#### 12.5.6.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing Required parameters : {groupId, ratingPlanId}
400	Region {region} is not valid

### 12.5.7. PUT /group

**PUT /group** Create a new group

The request line syntax for this method is as follows.

```
PUT /group
```

The required request payload is a JSON-formatted *GroupInfo* object. See example below.

### 12.5.7.0.1. Example Using cURL

The [example](#) below creates a new group with "QA" as its unique identifier. In this example the JSON-formatted *GroupInfo* object is specified in a text file named *group\_QA.txt* which is then referenced as the data input to the cURL command.

```
curl -X PUT -H "Content-Type: application/json" -k -u sysadmin:public \
-d @group_QA.txt https://localhost:19443/group
```

The *group\_QA.txt* file content in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"GroupInfo Object"** (page 777).

```
{
  "active": "true",
  "groupId": "QA",
  "groupName": "Quality Assurance Group",
  "ldapEnabled": false,
  "ldapGroup": "",
  "ldapMatchAttribute": "",
  "ldapSearch": "",
  "ldapSearchUserBase": "",
  "ldapServerURL": "",
  "ldapUserDNTemplate": "",
  "s3endpointshttp": ["ALL"],
  "s3endpointshttps": ["ALL"],
  "s3websiteendpoints": ["ALL"]
}
```

**Note** If you set the "ldapEnabled" attribute to "false" for a group that you are creating, you do not need to include the other "ldap\*" attributes in the *GroupInfo* object. However they are shown above for completeness. The S3 endpoint attributes are also optional.

### 12.5.7.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required attribute : {groupId}
400	Invalid JSON Object
400	Invalid Group ID
400	Invalid Active Status for Add Group
409	Unique constraint violation : {groupId}

## 12.5.8. group Query Parameters

### *groupId*

(Mandatory, string) Unique identifier of the group.

**Note** The System Admin group's "groupId" is "0".

### *prefix*

(Optional, string) With a *GET /group/list* request: A group ID prefix to use for filtering. For example, if you specify "prefix=usa" then only groups whose group ID starts with "usa" would be retrieved.

Defaults to empty string (meaning that no prefix-based filtering is performed).

### *limit*

(Optional, integer) With a *GET /group/list* request: For purposes of pagination, the maximum number of groups to return in one response. If more than this many groups meet the filtering criteria, then the actual number of groups returned will be "limit plus 1". The last group returned — the "plus 1" — is an indicator that there are more matching groups than could be returned in the current response (given the specified "limit" value). That group's ID can be used as the "offset" value in the next request. Note that if the offset group happens to be the last group in the entire set of matching groups, the subsequent query using the offset will return no groups.

Defaults to 100.

### *offset*

(Optional, string) With a *GET /group/list* request: The group ID with which to start the response list of groups for the current request, sorted alphanumerically. The "offset" parameter can be used for purposes of pagination within a large result set that is being retrieved via multiple sequential requests. See the description of "limit" above for more information.

If "offset" is not specified, the first group in the response list will be the alphanumerically first group from the entire result set.

### *ratingPlanId*

(Mandatory, string) With a *POST /group/ratingPlanId* request: Unique identifier of the rating plan to assign to the group.

### *region*

(Optional, string) If your service deployment has multiple service regions, rating plan assignment is on a per-region basis. With the *POST /group/ratingPlanId* method, use the "region" parameter to indicate the service region in which to apply the specified rating plan. For example, if *groupId=Engineering&ratingPlanId=Gold&region=East*, then the Gold rating plan will be applied to the Engineering group's service activity in the East region.

## 12.5.9. group Objects

This section describes the JSON-formatted data objects that constitute the request or response payloads for the Group related Admin API methods.

**Note** For examples of these objects see the API method request and response examples.

- **"GroupInfo Object"** (page 777)

### 12.5.9.1. GroupInfo Object

The *GroupInfo* object consists of the following attributes:

#### *active*

(Optional, string) Whether the group is enabled ("true") or disabled ("false") in the system. The users associated with a disabled group will be unable to access HyperStore data storage or to log in to the Cloudian Management Console. On a PUT of a new group, the "active" attribute defaults to "true" if not explicitly set. If not specified in a POST update of an existing group, the group retains its existing active or inactive status.

Example:

```
"active": "true"
```

#### *groupId*

(Mandatory, string) Group ID. Only letters, numbers, dashes, and underscores are allowed. Length must be at least 1 character and no more than 64 characters.

Example:

```
"groupId": "QA"
```

**Note** The System Admin group's "groupId" is "0".

**Note** In the CMC interface the field "Group Name" maps to the "groupId" attribute in the Admin API.

#### *groupName*

(Optional, string) Group name. Maximum length 64 characters. Example:

```
"groupName": "Quality Assurance Group"
```

**Note** In the CMC interface the field "Group Description" maps to the "groupName" attribute in the Admin API.

#### *ldapEnabled*

(Optional, boolean) Whether LDAP authentication is enabled for members of this group, *true* or *false*. Defaults to *false*. If LDAP authentication is enabled for the group, then by default when users from this group log into the CMC, the CMC will check against an LDAP system (with details as specified by other *GroupInfo* attributes, below) in order to authenticate the users. You can override this behavior on a per-user basis -- that is, you can configure certain users within the group so that they are authenticated by reference to a CMC-based password rather than an LDAP system. For more high-level information

about HyperStore's support for LDAP-based user authentication, see "**LDAP Integration**" (page 131).

Example:

```
"ldapEnabled": false
```

**Note** If you enable LDAP Authentication for an existing group to which users have already been added 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.

#### *ldapGroup*

(Optional, string) 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 *ldapGroup* value.

```
"ldapGroup": "Quality Assurance (U.S.)"
```

#### *ldapMatchAttribute*

(Optional, string) For background information about the purpose of this attribute, see the description of the *ldapSearch* attribute below.

Use the *ldapMatchAttribute* setting 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>*.

Example:

```
"ldapMatchAttribute": "l=California"
```

Example:

```
"ldapMatchAttribute": "memberOf=Sales"
```

#### *ldapSearch*

(Optional, string) If this is an LDAP-enabled group, and if you want to establish a LDAP-based user authorization filter to complement the user authentication template that you set with the *ldapUserDNTemplate* attribute, then use the *ldapSearch*, *ldapSearchUserBase*, and *ldapMatchAttribute* attributes 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.

Use the *ldapSearch* attribute to specify the user identifier type that you used in the *ldapUserDNTemplate*, in format "*(<LDAP\_user\_identifier\_attribute>={userId})*". This is used to retrieve a user's LDAP record in order to apply the filtering.

Example:

```
"ldapSearch": "(uid={userId})"
```

Example:

```
"ldapSearch": "(userPrincipalName={userId})"
```

*ldapSearchUserBase*

(Optional, string) For background information about the purpose of this attribute, see the description of the *ldapSearch* attribute above.

Use the *ldapSearchUserBase* attribute to specify the LDAP search base from which the CMC should start when retrieving the user's LDAP record in order to apply filtering. .

Example:

```
"ldapSearchUserBase": "dc=my-company,dc=com"
```

Example:

```
"ldapSearchUserBase": "uid={userId},ou=engineering,dc=my-company,dc=com"
```

*ldapServerURL*

(Optional, string) If this is an LDAP-enabled group, use this attribute to specify the URL that the CMC should use to access the LDAP Server when authenticating users in this group.

Note that if you use *ldaps* (LDAP secured by SSL/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.

Example:

```
"ldapServerURL": "ldap://my.ldap.server:389"
```

Example:

```
"ldapServerURL": "ldap://my.ldap.server:389/o=MyCompany"
```

*ldapUserDNTemplate*

(Optional, string) If this is an LDAP-enabled group, 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}" (exactly as shown 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:

```
"ldapUserDNTemplate": "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.

- **userPrincipalName.** With this approach the template would simply map the LDAP attribute "userPrincipalName" to the CMC variable "{userId}", like this:

```
"ldapUserDNTemplate": "userPrincipalName={userId}"
```

With the approach above LDAP-enabled users from this group will log in with their LDAP *user-PrincipalName* 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 *IdapSearch*, *IdapSearchUserBase*, and *IdapMatchAttribute* attributes (all described earlier in this topic) when you create the group in HyperStore.

#### *s3endpointshhttp*

(Optional, list<string>) The S3 HTTP service endpoint(s) that will be displayed to this group's users when those users log into the CMC and go to the **Security Credentials** page. The value can be:

- A single endpoint
- Multiple endpoints in a comma-separated list
- The string "ALL" (to indicate that this group's users will be able to see all of the system's configured S3 HTTP endpoints in the CMC's **Security Credentials** page)
- The string "NONE" (to indicate that this group's users will not be able to see any S3 HTTP endpoints in the CMC's **Security Credentials** page)

If the *s3endpointshhttp* attribute is omitted from the *GroupInfo* object in a *PUT /group* request, the attribute defaults to ["ALL"].

Example:

```
"s3endpointshhttp": ["ALL"]
```

**Note** This attribute and the other S3 endpoint attributes do not impact a group's users' authorization to access S3 endpoints. They only impact what S3 endpoint information is displayed to users in the CMC's **Security Credentials** page.

#### *s3endpointshhttps*

(Optional, list<string>) The S3 HTTPS service endpoint(s) that will be displayed to this group's users when those users log into the CMC and go to the **Security Credentials** page. The value can be:

- A single endpoint
- Multiple endpoints in a comma-separated list
- The string "ALL" (to indicate that this group's users will be able to see all of the system's configured S3 HTTPS endpoints in the CMC's **Security Credentials** page)
- The string "NONE" (to indicate that this group's users will not be able to see any S3 HTTPS endpoints in the CMC's **Security Credentials** page)

If the *s3endpointshhttps* attribute is omitted from the *GroupInfo* object in a *PUT /group* request, the attribute defaults to ["ALL"].

Example:

```
"s3endpointshhttps": ["ALL"]
```

#### *s3websiteendpoints*

(Optional, list<string>) The S3 website service endpoint(s) that will be displayed to this group's users when those users log into the CMC and go to the **Security Credentials** page. The value can be:

- A single endpoint
- Multiple endpoints in a comma-separated list

- The string "ALL" (to indicate that this group's users will be able to see all of the system's configured S3 website endpoints in the CMC's **Security Credentials** page)
- The string "NONE" (to indicate that this group's users will not be able to see any S3 website endpoints in the CMC's **Security Credentials** page)

If the *s3websiteendpoints* attribute is omitted from the *GroupInfo* object in a *PUT /group* request, the attribute defaults to ["ALL"].

Example:

```
"s3websiteendpoints": ["ALL"]
```

## 12.6. monitor

The Admin API methods built around the **monitor** resource are for monitoring the health and performance of your HyperStore system. There are methods for retrieving system and node statistics and for implementing system alert functionality.

### 12.6.1. DELETE /monitor/notificationrule

**DELETE /monitor/notificationrule** Delete a notification rule

The request line syntax for this method is as follows.

```
DELETE /monitor/notificationrule?ruleId=xxx[&region=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see "**monitor Query Parameters**" (page 799).

There is no request payload.

#### 12.6.1.0.1. Example Using cURL

The [example](#) below deletes a notification rule.

```
curl -X DELETE -k -u sysadmin:public \
https://localhost:19443/monitor/notificationrule?ruleId=8ef63b63-4961-4e17-88c7-d53c966557db
```

#### 12.6.1.0.2. Response Format

There is no response payload. For response status code this method will return one of the "**Common Response Status Codes**" (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameters : {ruleId}
400	Notification rule does not exist : {ruleId}

### 12.6.2. GET /monitor/events

**GET /monitor/events** Get the event list for a node

The request line syntax for this method is as follows.

```
GET /monitor/events?nodeId=xxx [&showAck=xxx] [&limit=xxx] [&region=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"monitor Query Parameters"** (page 799).

**Note** The CMC interface uses the term "alerts" rather than "events". Alert lists in the CMC are retrieved by this API method.

**Note** The HyperStore IAM Service supports a role-based access version of this API method as described in **"RBAC Version of this Method"** (page 783).

### 12.6.2.0.1. Example Using cURL

The [example](#) below retrieves the list of unacknowledged events for the node that has hostname "store1".

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/monitor/events?nodeId=store1 | python -mjson.tool
```

The response payload is a JSON-formatted list of *MonitoringEvent* objects, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"MonitoringEvent Object"** (page 802).

```
[
  {
    "ack": false,
    "condition": "<",
    "conditionVal": "0.15",
    "count": 1,
    "eventType": "13|/dev/mapper/vg0-root",
    "nodeId": "store1",
    "severityLevel": 2,
    "statId": "diskInfo",
    "timestamp": "1502797442785",
    "value": "/dev/mapper/vg0-root: 0.11198006761549427"
  },
  {
    "ack": false,
    "condition": "",
    "conditionVal": "",
    "count": 1,
    "eventType": "14|",
    "nodeId": "store1",
    "severityLevel": 0,
    "statId": "repairCompletionStatus",
    "timestamp": "1502794743351",
    "value": "REPAIR cmdno#: 610 status: COMPLETED"
  }
]
```

### 12.6.2.0.2. Response Format

The response payload is a JSON-formatted list of *MonitoringEvent* objects (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these

method-specific status codes:

Status Code	Description
400	Missing required parameters : {nodeId}
400	Invalid region : {region}
400	Invalid limit

### 12.6.2.1. RBAC Version of this Method

HyperStore's IAM Service supports an RBAC version of this API method. For an overview of the RBAC feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- Action name: *GetCloudianMonitorEvents*
- Parameters: Same as for *GET /monitor/events*, except all parameter names start with an upper case letter rather than lower case
- Response body: Same response data as for *GET /monitor/events* except the data is formatted in XML rather than JSON
- Role-based restrictions:
  - HyperStore system admin user can use this method
  - HyperStore group admin user cannot use this method
  - HyperStore regular user cannot use this method
  - IAM user can only use this method if granted *admin:GetCloudianMonitorEvents* permission by policy, and subject to the same restriction as the parent HyperStore user
- Sample request and response (abridged):

REQUEST

```
http://localhost:16080/?Action=GetCloudianMonitorEvents&NodeId=store1
```

<request headers including authorization info>

RESPONSE

200 OK

```
<?xml version="1.0" encoding="utf-8"?>
<GetCloudianMonitorEventsResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">
  <ResponseMetadata>
    <RequestId>system-generated-request-id</RequestId>
  </ResponseMetadata>
  <ListWrapper>
    <monitoringEvent>
      <ack>false</ack>
      etc...
    ...
    ...
  </monitoringEvent>
  <monitoringEvent>
    etc...
```

```
...
...
</monitoringEvent>
</ListWrapper>
</GetCloudianMonitorEventsResponse>
```

### 12.6.3. GET /monitor/nodelist

#### GET /monitor/nodelist    Get the list of monitored nodes

The request line syntax for this method is as follows.

```
GET /monitor/nodelist[?region=xxx]
```

For parameter description click on the parameter name or see **"monitor Query Parameters"** (page 799).

There is no request payload.

**Note** The HyperStore IAM Service supports a role-based access version of this API method as described in **"RBAC Version of this Method"** (page 784).

#### 12.6.3.0.1. Example Using cURL

The [example](#) below retrieves the list of monitored nodes in the default service region.

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/monitor/nodelist | python -mjson.tool
```

The response payload is a JSON-formatted list of hostnames, which in this example is as follows.

```
[
  "store1",
  "store2",
  "store3"
]
```

#### 12.6.3.0.2. Response Format

The response payload is a JSON-formatted list of hostnames (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746).

#### 12.6.3.1. RBAC Version of this Method

HyperStore's IAM Service supports an RBAC version of this API method. For an overview of the RBAC feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- Action name: *GetCloudianMonitorNodeList*
- Parameters: Same as for *GET /monitor/nodelist*, except all parameter names start with an upper case letter rather than lower case
- Response body: Same response data as for *GET /monitor/nodelist* except the data is formatted in XML rather than JSON

- Role-based restrictions:
  - HyperStore system admin user can use this method
  - HyperStore group admin user cannot use this method
  - HyperStore regular user cannot use this method
  - IAM user can only use this method if granted *admin:GetCloudianMonitorNodeList* permission by policy, and subject to the same restriction as the parent HyperStore user
- Sample request and response (abridged):

```

REQUEST

http://localhost:16080/?Action=GetCloudianMonitorNodeList

<request headers including authorization info>

RESPONSE

200 OK

<?xml version="1.0" encoding="utf-8"?>
<GetCloudianMonitorNodeListResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">
  <ResponseMetadata>
    <RequestId>system-generated-request-id</RequestId>
  </ResponseMetadata>
  <String>
hyperstore1
  </String>
</GetCloudianMonitorNodeListResponse>

```

## 12.6.4. GET /monitor/host

**GET /monitor/host** Get current monitoring statistics for a node

The request line syntax for this method is as follows.

```
GET /monitor/host?nodeId=xxx[&region=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"monitor Query Parameters"** (page 799).

There is no request payload.

**Note** The HyperStore IAM Service supports a role-based access version of this API method as described in **"RBAC Version of this Method"** (page 790).

### 12.6.4.0.1. Example Using cURL

The [example](#) below retrieves current monitoring statistics for the node that has hostname "store1".

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/monitor/host?nodeId=store1 | python -mjson.tool
```

The response payload is a JSON-formatted *MonitorNodeInfo* object, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"MonitorNodeInfo Object"** (page 803).

```
{
  "adminHeapMax": {
    "timestamp": "1502799543355",
    "value": "409075712"
  },
  "adminHeapUsed": {
    "timestamp": "1502799543355",
    "value": "109849080"
  },
  "cassMSGCCCount": {
    "timestamp": "1502799543355",
    "value": "3"
  },
  "cassMSGCTime": {
    "timestamp": "1502799543355",
    "value": "151"
  },
  "cassCopyGCCCount": null,
  "cassCopyGCTime": null,
  "cassHeapMax": {
    "timestamp": "1502799543355",
    "value": "2086666240"
  },
  "cassHeapUsed": {
    "timestamp": "1502799543355",
    "value": "1233215776"
  },
  "cassParNewGCCCount": {
    "timestamp": "1502799543355",
    "value": "4882"
  },
  "cassParNewGCTime": {
    "timestamp": "1502799543355",
    "value": "87598"
  },
  "cpu": {
    "timestamp": "1502799663530",
    "value": "0.06"
  },
  "diskAvailKb": {
    "timestamp": "1502799543355",
    "value": "21912316"
  },
  "diskIORead": {
    "timestamp": "1502799663530",
    "value": "0"
  },
  "diskIOWrite": {
    "timestamp": "1502799663530",
```

```

    "value": "93811"
  },
  "diskTotalKb": {
    "timestamp": "1502799543355",
    "value": "36056096"
  },
  "diskUsedKb": {
    "timestamp": "1502799543355",
    "value": "13330724"
  },
  "disksInfo": {
    "disks": [
      {
        "deviceName": "/dev/mapper/vg0-root",
        "diskAvailKb": "1776316",
        "diskIORead": "724419584",
        "diskIOWrite": "471087837184",
        "diskTotalKb": "15874468",
        "diskUsedKb": "13285096",
        "mountPoint": "/",
        "status": "OK",
        "storageUse": [
          "CASSANDRA",
          "REDIS",
          "LOG"
        ]
      },
      {
        "deviceName": "/dev/vdb1",
        "diskAvailKb": "20135940",
        "diskIORead": "3163136",
        "diskIOWrite": "50221056",
        "diskTotalKb": "20181628",
        "diskUsedKb": "45688",
        "mountPoint": "/cloudian1",
        "status": "OK",
        "storageUse": [
          "HS"
        ]
      }
    ]
  },
  "timestamp": "1502799663530"
},
"hyperStoreHeapMax": {
  "timestamp": "1502799543355",
  "value": "1635909632"
},
"hyperStoreHeapUsed": {
  "timestamp": "1502799543355",
  "value": "139187600"
},
"ioRx": {
  "timestamp": "1502799663530",

```

```
    "value": "17216"
  },
  "ioTx": {
    "timestamp": "1502799663530",
    "value": "28179"
  },
  "s3GetLatency": null,
  "s3GetTPS": null,
  "s3GetThruput": {
    "timestamp": "1502799543355",
    "value": "0"
  },
  "s3HeapMax": {
    "timestamp": "1502799543355",
    "value": "818020352"
  },
  "s3HeapUsed": {
    "timestamp": "1502799543355",
    "value": "164786136"
  },
  "s3PutLatency": {
    "timestamp": "1502799543355",
    "value": "18.4"
  },
  "s3PutTPS": {
    "timestamp": "1502799543355",
    "value": "0.0"
  },
  "s3PutThruput": {
    "timestamp": "1502799543355",
    "value": "0"
  },
  "status": {
    "ipaddr": "",
    "status": [
      "LOG_WARN"
    ],
    "timestamp": "1502799663530",
    "value": "[LOG_WARN]"
  },
  "svcAdmin": {
    "ipaddr": "10.10.2.91",
    "status": [
      "OK"
    ],
    "timestamp": "1502799663530",
    "value": "[OK]"
  },
  "svcCassandra": {
    "ipaddr": "10.10.2.91",
    "status": [
      "OK"
    ],
```

```

    "timestamp": "1502799663530",
    "value": "[OK]"
  },
  "svcHyperstore": {
    "ipaddr": "10.10.2.91",
    "status": [
      "OK"
    ],
    "timestamp": "1502799663530",
    "value": "[OK]"
  },
  "svcRedisCred": {
    "ipaddr": "10.10.2.91",
    "status": [
      "OK"
    ],
    "timestamp": "1502799663530",
    "value": "[OK]"
  },
  "svcRedisMon": {
    "ipaddr": "10.10.2.91",
    "status": [
      "OK"
    ],
    "timestamp": "1502799663530",
    "value": "[OK]"
  },
  "svcRedisQos": {
    "ipaddr": "10.10.2.91",
    "status": [
      "OK"
    ],
    "timestamp": "1502799663530",
    "value": "[OK]"
  },
  "svcS3": {
    "ipaddr": "10.10.2.91",
    "status": [
      "OK"
    ],
    "timestamp": "1502799663530",
    "value": "[OK]"
  }
}

```

#### 12.6.4.0.2. Response Format

The response payload is a JSON-formatted *MonitorNodeInfo* object (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameters : {nodeId}

### 12.6.4.1. RBAC Version of this Method

HyperStore's IAM Service supports an RBAC version of this API method. For an overview of the RBAC feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- Action name: *GetCloudianMonitorHost*
- Parameters: Same as for *GET /monitor/host*, except all parameter names start with an upper case letter rather than lower case
- Response body: Same response data as for *GET /monitor/host* except the data is formatted in XML rather than JSON
- Role-based restrictions:
  - HyperStore system admin user can use this method
  - HyperStore group admin user cannot use this method
  - HyperStore regular user cannot use this method
  - IAM user can only use this method if granted *admin:GetCloudianMonitorHost* permission by policy, and subject to the same restriction as the parent HyperStore user
- Sample request and response (abridged):

```
REQUEST

http://localhost:16080/?Action=GetCloudianMonitorHost

<request headers including authorization info>

RESPONSE

200 OK

<?xml version="1.0" encoding="utf-8"?>
<GetCloudianMonitorHostResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">
  <ResponseMetadata>
    <RequestId>system-generated-request-id</RequestId>
  </ResponseMetadata>
  <MonitorNodeInfo>
    <adminHeapMax>
      <timestamp>1534534923619</timestamp>
      <value>1538260992</value>
    </adminHeapMax>
    etc...
    ...
    ...
  </MonitorNodeInfo>
</GetCloudianMonitorHostResponse>
```

### 12.6.5. GET /monitor

**GET /monitor**    Get current monitoring statistics for a service region

The request line syntax for this method is as follows.

```
GET /monitor? [region=xxx]
```

For parameter description click on the parameter name or see "**monitor Query Parameters**" (page 799).

There is no request payload.

**Note** The HyperStore IAM Service supports a role-based access version of this API method as described in "**RBAC Version of this Method**" (page 792).

### 12.6.5.0.1. Example Using cURL

The [example](#) below retrieves current monitoring statistics for the default service region.

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/monitor | python -mjson.tool
```

The response payload is a JSON-formatted *MonitorSystemInfo* object, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see "**MonitorSystemInfo Object**" (page 810).

```
{
  "diskAvailKb": {
    "timestamp": "1502799843254",
    "value": "61855592"
  },
  "diskTotalKb": {
    "timestamp": "1502799843254",
    "value": "88115680"
  },
  "diskUsedKb": {
    "timestamp": "1502799843254",
    "value": "23814368"
  },
  "nodeStatuses": [
    {
      "hostname": "store1",
      "ipaddr": null,
      "status": [
        "LOG_WARN"
      ],
      "timestamp": "1502799843254",
      "value": "[LOG_WARN]"
    },
    {
      "hostname": "store2",
      "ipaddr": null,
      "status": [
        "LOG_WARN"
      ],
      "timestamp": "1502799843254",
      "value": "[LOG_WARN]"
    }
  ]
}
```

```

    "hostname": "store3",
    "ipaddr": null,
    "status": [
        "LOG_WARN"
    ],
    "timestamp": "1502799843254",
    "value": "[LOG_WARN]"
  },
  "s3GetLatency": null,
  "s3GetTPS": null,
  "s3GetThruput": {
    "timestamp": "1502799843254",
    "value": "0"
  },
  "s3PutLatency": {
    "timestamp": "1502799843254",
    "value": "130.1"
  },
  "s3PutTPS": {
    "timestamp": "1502799843254",
    "value": "0.0"
  },
  "s3PutThruput": {
    "timestamp": "1502799843254",
    "value": "0"
  },
  "status": {
    "ipaddr": "",
    "status": [
        "OK"
    ],
    "timestamp": "1502799843254",
    "value": "[OK]"
  }
}

```

#### 12.6.5.0.2. Response Format

The response payload is a JSON-formatted *MonitorSystemInfo* object (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Invalid region : {region}

#### 12.6.5.1. RBAC Version of this Method

HyperStore's IAM Service supports an RBAC version of this API method. For an overview of the RBAC feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- Action name: *GetCloudianMonitorRegion*
- Parameters: Same as for *GET /monitor*, except all parameter names start with an upper case letter rather than lower case
- Response body: Same response data as for *GET /monitor* except the data is formatted in XML rather than JSON
- Role-based restrictions:
  - HyperStore system admin user can use this method
  - HyperStore group admin user cannot use this method
  - HyperStore regular user cannot use this method
  - IAM user can only use this method if granted *admin:GetCloudianMonitorRegion* permission by policy, and subject to the same restriction as the parent HyperStore user
- Sample request and response (abridged):

```

REQUEST

http://localhost:16080/?Action=GetCloudianMonitorRegion

<request headers including authorization info>

RESPONSE

200 OK
<?xml version="1.0" encoding="utf-8"?>
<GetCloudianMonitorRegionResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">
  <ResponseMetadata>
    <RequestId>system-generated-request-id</RequestId>
  </ResponseMetadata>
  <MonitorSystemInfo>
    <status>
    <timestamp>1534535223489</timestamp>
    etc...
    ...
  </MonitorSystemInfo>
</GetCloudianMonitorRegionResponse>

```

### 12.6.6. GET /monitor/history

**GET /monitor/history** Get historical monitoring statistics for a node

The request line syntax for this method is as follows.

```
GET /monitor?nodeId=xxx [&region=xxx] &statId=xxx&startTime=xxx&endTime=xxx
```

For parameter description click on the parameter name or see "**monitor Query Parameters**" (page 799).

There is no request payload.

### 12.6.6.0.1. Example Using cURL

The [example](#) below retrieves history for the "cpu" statistic, for a one hour period (2019 July 20th, midnight to 1AM). Note that the system interprets the start and end times as **GMT** times.

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/monitor/history?nodeId=
hs1&statId=cpu&startTime=201907201200&endTime=201907201300' \
| python -mjson.tool
```

The response payload is a JSON-formatted list of timestamp/value pairs (with the timestamps as UTC milliseconds). Note that the response body does **not** include the statistic ID. Depending on the statistic, there will be either one timestamp/value pair per minute or one timestamp/value pair per five minutes, throughout the requested startTime / endTime interval (see [statId](#) for detail). In this truncated example, it's one per five minutes.

```
[
  {
    "timestamp": "1563624003885",
    "value": "0.21"
  },
  {
    "timestamp": "1563624303754",
    "value": "0.21"
  },
  {
    "timestamp": "1563624603451",
    "value": "0.21"
  },
  ...
]
```

### 12.6.6.0.2. Response Format

The response payload is a JSON-formatted list of timestamp/value pairs (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing parameter : {parameter}
400	Both regionId and nodeId are empty
400	Invalid region : {region}
400	Invalid parameter : {parameter}

## 12.6.7. GET /monitor/notificationrules

**GET /monitor/notificationrules** Get the list of notification rules

The request line syntax for this method is as follows.

```
GET /monitor/notificationrules[?region=xxx]
```

For parameter description click on the parameter name or see **"monitor Query Parameters"** (page 799).

There is no request payload.

### 12.6.7.0.1. Example Using cURL

The [example](#) below retrieves the current list of notification rules for the default service region.

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/monitor/notificationrules | python -mjson.tool
```

The response payload is a JSON-formatted list of *NotificationRule* objects, which in this example is as follows. The example is truncated so that only a few rules are shown. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"NotificationRule Object"** (page 813).

```
[
  {
    "condition": ">",
    "conditionVal": "0.9",
    "email": "default",
    "enabled": true,
    "region": "",
    "ruleId": "12",
    "severityLevel": 1,
    "snmpTrap": false,
    "statId": "cpu"
  },
  {
    "condition": "",
    "conditionVal": "",
    "email": "default",
    "enabled": true,
    "region": "",
    "ruleId": "19",
    "severityLevel": 3,
    "snmpTrap": false,
    "statId": "currentFailDiskInfo"
  },
  {
    "condition": "<",
    "conditionVal": "0.1",
    "email": "default",
    "enabled": true,
    "region": "",
    "ruleId": "11",
    "severityLevel": 2,
    "snmpTrap": false,
    "statId": "diskAvail"
  },
  ...
  ...
]
```

### 12.6.7.0.2. Response Format

The response payload is a JSON-formatted list of *NotificationRule* objects (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746).

## 12.6.8. POST /monitor/acknowledgeevents

### POST /monitor/acknowledgeevents Acknowledge monitoring events

The request line syntax for this method is as follows.

```
POST /monitor/acknowledgeevents?nodeId=xxx[&region=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"monitor Query Parameters"** (page 799).

The required request payload is a JSON-formatted *EventsAck* object. See example below.

#### 12.6.8.0.1. Example Using cURL

The [example](#) below acknowledges two monitoring events from the node with hostname "store1" (the same two events that were retrieved in the [GET /monitor/events](#) example). In this example the JSON-formatted *Event-sAck* object is specified in a text file named *event\_acknowledge.txt* which is then referenced as the data input to the cURL command.

```
curl -X POST -H "Content-Type: application/json" -k -u sysadmin:public \
-d @event_acknowledge.txt \
https://localhost:19443/monitor/acknowledgeevents?nodeId=store1
```

The *event\_acknowledge.txt* file content in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"EventsAck Object"** (page 801).

```
{
  "eventTypes": ["13|/dev/mapper/vg0-root", "14|"],
  "nodeId": "store1"
}
```

#### 12.6.8.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameter : {nodeId, events}
400	Invalid region : {region}
400	Invalid JSON object

## 12.6.9. POST /monitor/notificationruleenable

### POST /monitor/notificationruleenable Enable or disable notification rules

The request line syntax for this method is as follows.

```
POST /monitor/notificationruleenable
```

The required request payload is a JSON-formatted *NotificationRulesEnable* object. See example below.

You can use this method to disable notification rules or to re-enable rules that you've previously disabled. When a notification rule is disabled the rule will not trigger system event notifications.

**Note** To disable or re-enable just one notification rule, you can use either the *POST /monitor/notificationruleenable* method or the [POST /monitor/notificationrule](#) method. To disable or re-enable multiple notification rules in one operation use the *POST /monitor/notificationruleenable* method.

#### 12.6.9.0.1. Example Using cURL

The [example](#) below disables two notification rules. In this example the JSON-formatted *NotificationRulesEnable* object is specified in a text file named *rule\_disable.txt* which is then referenced as the data input to the cURL command.

```
curl -X POST -H "Content-Type: application/json" -k -u sysadmin:public \
-d @rule_disable.txt https://localhost:19443/monitor/notificationruleenable
```

The *rule\_disable.txt* file content in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"NotificationRulesEnable Object"** (page 818).

```
{
  "enable":false,
  "regionId": "",
  "ruleList": [ "836da4bf-c6cc-4f73-afa3-9854ce407ca6", "8ef63b63-4961-4e17-88c7-d53c966557db" ]
}
```

#### 12.6.9.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameters : {ruleId}
400	Notification rule does not exist : {ruleId}

### 12.6.10. POST /monitor/notificationrule

#### POST /monitor/notificationrule    Change a notification rule

The request line syntax for this method is as follows.

```
POST /monitor/notificationrule
```

The required request payload is a JSON-formatted *NotificationRule* object. See example below.

### 12.6.10.0.1. Example Using cURL

The [example](#) below changes an existing notification rule (the rule that was created in the [PUT /monitor/notificationrule](#) description). In this example the JSON-formatted *NotificationRule* object is specified in a text file named *rule\_s3GetLatency.txt* which is then referenced as the data input to the cURL command.

```
curl -X POST -H "Content-Type: application/json" -k -u sysadmin:public \
-d @rule_s3GetLatency.txt https://localhost:19443/monitor/notificationrule
```

The *rule\_s3GetLatency.txt* file content in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"NotificationRule Object"** (page 813).

```
{
  "condition": ">",
  "conditionVal": "150",
  "email": "default",
  "enabled": true,
  "region": "",
  "ruleId": "836da4bf-c6cc-4f73-afa3-9854ce407ca6",
  "severityLevel": 2,
  "snmpTrap": false,
  "statId": "s3GetLatency"
}
```

**Note** Unlike when you create a new notification rule, when you change an existing rule you must specify the rule's "ruleId" value in the *NotificationRule* object. If you're not sure of a rule's ID you can retrieve it using the [GET /monitor/notificationrules](#) method.

### 12.6.10.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Invalid JSON object
400	Notification rule Id does not exist : {ruleId}

## 12.6.11. PUT /monitor/notificationrule

### PUT /monitor/notificationrule Create a new notification rule

The request line syntax for this method is as follows.

```
PUT /monitor/notificationrule
```

The required request payload is a JSON-formatted *NotificationRule* object. See example below.

**Note** The HyperStore system comes with many pre-configured notification rules. To see the existing set of rules go to the CMC's [Alert Rules](#) page. The CMC interface uses the term "alert rules" rather than "notification rules".

### 12.6.11.0.1. Example Using cURL

The [example](#) below creates a new notification rule. In this example the JSON-formatted *NotificationRule* object is specified in a text file named *rule\_s3GetLatency.txt* which is then referenced as the data input to the cURL command.

```
curl -X PUT -H "Content-Type: application/json" -k -u sysadmin:public \
-d @rule_s3GetLatency.txt https://localhost:19443/monitor/notificationrule
```

The *rule\_s3GetLatency.txt* file content in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"NotificationRule Object"** (page 813).

```
{
  "condition": ">",
  "conditionVal": "150",
  "email": "default",
  "enabled": true,
  "region": "",
  "ruleId": "",
  "severityLevel": 1,
  "snmpTrap": false,
  "statId": "s3GetLatency"
}
```

### 12.6.11.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Invalid JSON object

## 12.6.12. monitor Query Parameters

### *ruleId*

(Mandatory, string) For a *DELETE /monitor/notificationrule* request: The system-generated unique ID for the notification rule. For the default notification rules that come packaged with the HyperStore system this will be a simple integer like "1", "2", or "14". For rules that you create yourself the system will generate a ruleId in the form of a UUID string like "8e4cc533-360a-4dd5-bfe4-6b5f5b6c40da".

If you do not know the ruleId, you can retrieve it by using the *GET /monitor/notificationrules* method. That method returns a list of notification rules which includes each rule's ruleId.

### *region*

(Optional, string) Service region. If you do not specify a region, the default region is assumed.

### *nodeId*

(Mandatory, string) For a *GET /monitor/events* or *GET /monitor/host* or *GET /monitor/history* or *POST /monitor/acknowledgeevents* request: The hostname of the target node.

### *showAck*

(Optional, boolean) For a *GET /monitor/events* request: Whether to return acknowledged events as well

as unacknowledged events, true or false.

If not specified in the request, defaults to false (only unacknowledged events are returned).

#### *limit*

(Optional, integer) For a *GET /monitor/events* request: The maximum number of events to return in the response.

If not specified in the request, the default limit is 100 events.

#### *statId*

(Mandatory, string) For a *GET /monitor/history* request: the statistic for which to retrieve a history. The supported statistic IDs are listed in the table below. Depending on the statistic, the returned history will include either one data point (one instance of the statistic value) per minute or one data point per five minutes, across the time interval bounded by the *startTime* and *endTime* specified in the request.

The *GET /monitor/history* call only supports one statistic ID per request. You cannot request multiple or all statistics IDs in a single request.

For more information about a particular statistic, see **"MonitorNodeInfo Object"** (page 803).

StatId	Data Point Frequency
diskIORead	Every minute
diskIOWrite	
ioTx	
ioRx	
cpu	Every five minutes
s3GetTPS	
s3PutTPS	
s3GetThruput	
s3PutThruput	
s3GetLatency	
s3PutLatency	
adminHeapUsed	
cassHeapUsed	
hyperStoreHeapUsed	
s3HeapUsed	

#### *startTime*

(Mandatory, string) For a *GET /monitor/history* request: the start time of the interval for which to retrieve the statistic history, in format *yyyyMMddHHmm* (for example 201907200000). The system interprets this as a **GMT time**, so when specifying your desired start time do it in terms of the GMT time zone -- not the local time.

#### *endTime*

(Mandatory, string) For a *GET /monitor/history* request: the end time of the interval for which to retrieve

the statistic history, in format *yyyyMMddHHmm* (for example 201907201200). The system interprets this as a **GMT time**, so when specifying your desired end time do it in terms of the GMT time zone -- not the local time.

### 12.6.13. monitor Objects

This section describes the JSON-formatted data objects that constitute the request or response payloads for the Monitor related Admin API methods.

**Note** For examples of these objects see the API method request and response examples.

- **"EventsAck Object"** (page 801)
- **"MonitoringEvent Object"** (page 802)
- **"MonitorNodeInfo Object"** (page 803)
- **"MonitorSystemInfo Object"** (page 810)
- **"NotificationRule Object"** (page 813)
- **"NotificationRulesEnable Object"** (page 818)

#### 12.6.13.1. EventsAck Object

The *EventsAck* object consists of the following attributes:

##### *delete*

(Optional, boolean) If this attribute is included and set to *true* then the events specified by the "eventTypes" attribute will be immediately deleted from the system.

If this attribute is omitted or set to *false* then the events specified by the "eventTypes" attribute will be marked as acknowledged but will not yet be deleted from system. Such acknowledged events will instead be deleted automatically after a time period configured by the *events.acknowledged.ttl* property in the *mts.properties.erb* configuration file. By default this period is 86400 seconds (one day).

Example:

```
"delete": true
```

##### *eventTypes*

(Mandatory, list<string>) List of the eventTypes being acknowledged. For non-log events (such as a service down event or a threshold crossing event), the eventType is the integer ruleId value from the notification rule that triggered the event. For log events (events triggered by the writing of an application log message), the eventType is formatted as "<ruleId>|<logCategory>". The specific eventTypes currently present on a node can be retrieved with the *GET /monitor/events* method.

Example:

```
"eventTypes": ["13|/dev/mapper/vg0-root", "14|"]
```

##### *nodeId*

(Mandatory, string) Hostname of the node on which the event(s) occurred. Example:

```
"nodeId": "store1"
```

*regionId*

(Optional, string) Name of service region in which the event(s) occurred. Defaults to the default service region. Example:

```
"regionId": "southwest"
```

### 12.6.13.2. MonitoringEvent Object

The *MonitoringEvent* object consists of the following attributes:

*ack*

(Boolean) Whether the event has been acknowledged, true or false. This will be false unless you explicitly retrieved acknowledged events when you executed the *GET /monitor/events* method. Example:

```
"ack": false
```

*condition*

(String) From the notification rule that triggered this event, the condition comparison type in the rule definition. This will be "=", "<", or ">". Example:

```
"condition": "<"
```

*conditionVal*

(String) From the notification rule that triggered this event, the condition value against which to compare. For example, this may be a numerical threshold value, or a service status such as "SVC\_DOWN", or a log message level such as "LOG\_ERR". Example:

```
"conditionVal": "0.15"
```

*count*

(Integer) Number of times that the event has occurred without being acknowledged. Example:

```
"count": 1
```

*eventType*

(String) From the notification rule that triggered this event, the integer <ruleId> value . (Or, for log events, a concatenation of "<ruleId>|<logCategory>". The logCategory is derived from the line of code that generates the specific log message.) Example:

```
"eventType": "13|/dev/mapper/vg0-root"
```

*nodeId*

(String) Hostname of the node on which the event occurred. Example:

```
"nodeId": "store1"
```

*severityLevel*

(Integer) Severity level of the event, as configured in the notification rule for the event. This is an integer with meaning as follows:

- 0 = Low
- 1 = Medium

- 2 = High
- 3 = Critical

Example:

```
"severityLevel": 2
```

*statId*

(String) From the notification rule that triggered this event, the statId. See **"NotificationRule Object"** (page 813) for a list of supported statIds.

Example:

```
"statId": "diskInfo"
```

**Note** The "svcS3" statId encompasses events pertaining to auto-tiering and cross-region replication, as well as events pertaining to providing S3 service to clients.

*timestamp*

(String) Timestamp of latest event occurrence in UTC milliseconds. Example:

```
"timestamp": "1502797442785"
```

*value*

(String) On the node, the statistic value that triggered this event. For example, if the event was triggered by a statistic rising to a threshold-exceeding value, this attribute would indicate that value. In the case of a log event, the "value" is the log message.

Example:

```
"value": "/dev/mapper/vg0-root: 0.11198006761549427"
```

### 12.6.13.3. MonitorNodeInfo Object

The *MonitorNodeInfo* object consists of the following attributes and nested objects:

**Note** Within the *MonitorNodeInfo* object, all statistics for which the data type (in the descriptions below) is *MonitorStat* are formatted as "**<statName>**": {"**timestamp**": "**<UTCMilliseconds>**", "**value**": "**<statValue>**"}.

*adminHeapMax*

([MonitorStat](#)) The maximum JVM heap size allocated to the Admin Service, in bytes. Example:

```
"adminHeapMax": {"timestamp": "1502799543355", "value": "409075712"}
```

*adminHeapUsed*

([MonitorStat](#)) The Admin Service's current JVM heap memory usage on the node, in bytes. This is measured each five minutes. Example:

```
"adminHeapUsed": {"timestamp": "1502799543355", "value": "109849080"}
```

*cassMSGCCCount*

([MonitorStat](#)) The number of concurrent mark-sweep (CMS) garbage collections executed since the last start-up of the Cassandra service on this node. This collection type targets old-generation objects. Example:

```
"cassMSGCCCount": {"timestamp": "1502799543355", "value": "3"}
```

#### *cassMSGCTime*

([MonitorStat](#)) The aggregate time (in milliseconds) spent on executing CMC garbage collections since the last start-up of the Cassandra service on this node. Example:

```
"cassMSGCTime": {"timestamp": "1502799543355", "value": "151"}
```

#### *cassCopyGCCCount*

([MonitorStat](#)) The number of Copy garbage collections executed since the last start-up of the Cassandra service on this node. Example:

```
"cassCopyGCCCount": null
```

#### *cassCopyGCTime*

([MonitorStat](#)) The aggregate time (in milliseconds) spent on executing Copy garbage collections since the last start-up of the Cassandra service on this node. Example:

```
"cassCopyGCTime": null
```

#### *cassHeapMax*

([MonitorStat](#)) The maximum JVM heap size allocated to the Cassandra Service, in bytes. Example:

```
"cassHeapMax": {"timestamp": "1502799543355", "value": "2086666240"}
```

#### *cassHeapUsed*

([MonitorStat](#)) The Cassandra Service's current JVM heap memory usage on the node, in bytes. This is measured each five minutes. Example:

```
"cassHeapUsed": {"timestamp": "1502799543355", "value": "1233215776"}
```

#### *cassParNewGCCCount*

([MonitorStat](#)) The number of parallel new-generation (ParNew) garbage collections executed since the last start-up of the Cassandra service on this node. Example:

```
"cassParNewGCCCount": {"timestamp": "1502799543355", "value": "4882"}
```

#### *cassParNewGCTime*

([MonitorStat](#)) The aggregate time (in milliseconds) spent on executing ParNew garbage collections since the last start-up of the Cassandra service on this node. Example:

```
"cassParNewGCTime": {"timestamp": "1502799543355", "value": "87598"}
```

#### *cpu*

([MonitorStat](#)) Current CPU utilization percentage on the node. This is measured once per every five minutes. Example:

```
"cpu": {"timestamp": "1502799663530", "value": "0.06"}
```

#### *diskAvailKb*

([MonitorStat](#)) On the node, the total mounted disk space that's still available for S3 object storage

(HyperStore data directories) or Cassandra metadata storage (Cassandra data directory). Reported as a number of kilobytes.

The following are deducted from the total amount of unused disk space to arrive at the "available" disk space amount that is reported by the *diskAvailKb* statistic:

- Each disk's "reserved-blocks-percentage" (the portion of the disk that's reserved for privileged processes)
- Each HyperStore data disk's "stop-write" buffer (10% of capacity by default). For more information on the stop-write feature see **"Automatic Stop of Writes to a Disk at 90% Usage"** (page 157).

Example:

```
"diskAvailKb": {"timestamp": "1502799543355", "value": "21912316"}
```

**Note** Because the reserved-blocks-percentage and the stop-write buffer percentage are not counted as available space, the *diskUsedKb* and *diskAvailKb* will add up to less than the *diskTotalKb*.

#### *diskIORead*

([MonitorStat](#)) Across all the node's disks that are being used for S3 object storage or Cassandra metadata storage, the average disk read throughput in bytes per second. This stat is recalculated each minute, based on the most recent minute of activity. Example:

```
"diskIORead": {"timestamp": "1502799663530", "value": "0"}
```

#### *diskIOWrite*

([MonitorStat](#)) Across all the node's disks that are being used for S3 object storage or Cassandra metadata storage, the average disk write throughput in bytes per second. This stat is recalculated each minute, based on the most recent minute of activity. Example:

```
"diskIOWrite": {"timestamp": "1502799663530", "value": "93811"}
```

#### *diskTotalKb*

([MonitorStat](#)) On the node, the total size of the disks mounted for S3 object storage (HyperStore data directories) or Cassandra metadata storage (Cassandra data directory). Reported as a number of kilobytes. Example:

```
"diskTotalKb": {"timestamp": "1502799543355", "value": "36056096"}
```

#### *diskUsedKb*

([MonitorStat](#)) On the node, the total disk space that's been consumed for S3 object storage (HyperStore data directories) or Cassandra metadata storage (Cassandra data directory). Reported as a number of kilobytes. Example:

```
"diskUsedKb": {"timestamp": "1502799543355", "value": "13330724"}
```

#### *disksInfo*

(DiskMonitorStat) Current information about each disk on the node. Formatted as {"disks": List<DiskInfo>, "timestamp": "<UTCMilliseconds>"}. There is one nested *DiskInfo* object for each disk on the node. Each *DiskInfo* object consists of the following attributes:

*deviceName*

(String) Disk drive device name. Example:

```
"deviceName": "/dev/mapper/vg0-root"
```

*diskAvailKb*

(String) Total remaining free space on the disk in number of KBs. In calculating the available space, a disk's "reserved-blocks-percentage" (the portion of the disk space that's reserved for privileged processes) is considered to be unavailable. By default in Linux systems the configurable "reserved-blocks-percentage" for a file system is 5% of disk capacity. If this is a Hyper-Store data disk, then the "stop-write" buffer (10% of disk capacity by default) is also considered to be unavailable.

Example:

```
"diskAvailKb": "1776316"
```

*diskIORead*

(String) The average disk read throughput for this disk, in bytes per second. This stat is recalculated each minute, based on the most recent minute of data. Example:

```
"diskIORead": "724419584"
```

*diskIOWrite*

(String) The average disk write throughput for this disk, in bytes per second. This stat is recalculated each minute, based on the most recent minute of data. Example:

```
"diskIOWrite": "471087837184"
```

*diskTotalKb*

(String) Total capacity of the disk in number of KBs. Example:

```
"diskTotalKb": "15874468"
```

*diskUsedKb*

(String) Amount of used space on the disk in number of KBs. Example:

```
"diskUsedKb": "13285096"
```

*mountPoint*

(String) File system mount point for the disk. Example:

```
"mountPoint": "/"
```

*status*

(EDiskStatus) Disk status string. One of "OK", "ERROR", or "DISABLED". For description of these disk statuses, see **"View a Node's Disk Detail"** (page 317).

Example:

```
"status": "OK"
```

*storageUse*

(EStorageType) List of storage type strings, indicating what type of data is being stored on the

disk. One or more of:

- "CASSANDRA" — System metadata and S3 object metadata in Cassandra.
- "REDIS" — System metadata in Redis.
- "LOG" — Application logs
- "HS" — Replicated S3 object data.
- "EC" — Erasure coded S3 object data.
- "NOTAVAIL" — Storage usage information cannot be retrieved for this disk.

Example:

```
"storageUse": ["CASSANDRA", "REDIS", "LOG"]
```

#### *hyperStoreHeapMax*

([MonitorStat](#)) The maximum JVM heap size allocated to the HyperStore Service, in bytes. Example:

```
"hyperStoreHeapMax": {"timestamp": "1502799543355", "value": "1635909632"}
```

#### *hyperStoreHeapUsed*

([MonitorStat](#)) The HyperStore Service's current JVM heap memory usage on the node, in bytes. This is measured each five minutes. Example:

```
"hyperStoreHeapUsed": {"timestamp": "1502799543355", "value": "139187600"}
```

#### *ioRx*

([MonitorStat](#)) The aggregate network bytes per second received by the node, for all types of network traffic including but not limited to S3 request traffic. This stat is recalculated each minute, based on the most recent minute of activity. Example:

```
"ioRx": {"timestamp": "1502799663530", "value": "17216"}
```

#### *ioTx*

([MonitorStat](#)) The aggregate network bytes per second transmitted by the node, for all types of network traffic including but not limited to S3 request traffic. This stat is recalculated each minute, based on the most recent minute of activity. Example:

```
"ioTx": {"timestamp": "1502799663530", "value": "28179"}
```

#### *s3GetLatency*

([MonitorStat](#)) On the node, the 95th percentile request latency value for S3 GET transactions, in milliseconds. New statistic values are calculated each five minutes, based on the most recent approximately 1000 GET transactions. The s3GetLatency value indicates that of the last 1000 GET transactions, 95% completed in that many milliseconds or less.

Example:

```
"s3GetLatency": null
```

**Note** HEAD transactions are counted toward this stat also.

#### *s3GetTPS*

([MonitorStat](#)) On the node, the number of S3 GET transactions processed per second. This stat is

recalculated each five minutes, based on the most recent five minutes of activity. HEAD transactions are counted toward this stat also. Example:

```
"s3GetTPS": null
```

#### *s3GetThruput*

([MonitorStat](#)) On the node, the data throughput for S3 GET transactions, expressed as MB per second. This stat is recalculated each five minutes, based on the most recent five minutes of transaction data. HEAD transactions are counted toward this stat also. Example:

```
"s3GetThruput": {"timestamp": "1502799543355", "value": "0"}
```

#### *s3HeapMax*

([MonitorStat](#)) The maximum JVM heap size allocated to the S3 Service, in bytes. Example:

```
"s3HeapMax": {"timestamp": "1502799543355", "value": "818020352"}
```

#### *s3HeapUsed*

([MonitorStat](#)) The S3 Service's current JVM heap memory usage on the node, in bytes. This is measured each five minutes. Example:

```
"s3HeapUsed": {"timestamp": "1502799543355", "value": "164786136"}
```

#### *s3PutLatency*

([MonitorStat](#)) On the node, the 95th percentile request latency value for S3 PUT transactions, in milliseconds. New statistic values are calculated each five minutes, based on the most recent approximately 1000 PUT transactions. The s3PutLatency value indicates that of the last 1000 PUT transactions, 95% completed in that many milliseconds or less.

Example:

```
"s3PutLatency": {"timestamp": "1502799543355", "value": "18.4"}
```

**Note** POST transactions are counted toward this stat also.

#### *s3PutTPS*

([MonitorStat](#)) On the node, the number of S3 PUT transactions processed per second. This stat is recalculated each five minutes, based on the most recent five minutes of transaction data. POST transactions are counted toward this stat also. Example:

```
"s3PutTPS": {"timestamp": "1502799543355", "value": "0.0"}
```

#### *s3PutThruput*

([MonitorStat](#)) On the node, the data throughput for S3 PUT transactions, expressed as MB per second. This stat is recalculated each five minutes, based on the most recent five minutes of transaction data. POST transactions are counted toward this stat also. Example:

```
"s3PutThruput": {"timestamp": "1502799543355", "value": "0"}
```

#### *status*

(ServiceStatus) Overall status of the node.

Formatted as `{"ipaddr": "<empty>", "status": [<list of one or more of "OK", "SVC_DOWN", "LOG_WARN", or "LOG_ERR">], "timestamp": "<UTCMilliseconds>", "value": "<statusValueFormattedAsString>"}`.

The "ipaddr" value will be empty or null here; an IP address is specified only in the service-specific status attributes (such as "svcCassandra") described below.

Possible values within the "status" list are:

- "OK" — All HyperStore services are up and running on the node, and the node has no unacknowledged events.
- "SVC\_DOWN" — One or more HyperStore services (Admin, Cassandra, HyperStore, Redis QoS, Redis Credentials, Redis Monitor, or S3) is down on the node
- "LOG\_WARN" — There are unacknowledged warnings in an application log on the node (such as the S3 Service application log or the Cassandra application log).
- "LOG\_ERR" — There are unacknowledged errors in an application service log.

The "value" attribute will be identical to the "status" attribute, except formatted as a single string rather than a list of strings.

Example:

```
"status": {"ipaddr": "", "status": ["SVC_DOWN", "LOG_WARN"], "timestamp":
"1502799663530", "value": "[SVC_DOWN, LOG_WARN]"}
```

#### svcAdmin

([ServiceStatus](#)) Admin service status on the node.

Formatted as `{"ipaddr": "<nodeIPAddress>", "status": [<list of one or more of "OK", "SVC_DOWN", "LOG_WARN", or "LOG_ERR">], "timestamp": "<UTCMilliseconds>", "value": "<statusValueFormattedAsString>"}`.

Example:

```
"svcAdmin": {"ipaddr": "10.10.2.91", "status": ["OK"], "timestamp":
"1502799663530", "value": "[OK]"}
```

The other service status attributes that follow below (svcCassandra and so on) are formatted in the same way.

#### svcCassandra

([ServiceStatus](#)) Cassandra service status on the node. Example:

```
"svcCassandra": {"ipaddr": "10.10.2.91", "status": ["OK"], "timestamp":
"1502799663530", "value": "[OK]"}
```

#### svcHyperstore

([ServiceStatus](#)) HyperStore service status on the node. Example:

```
"svcHyperstore": {"ipaddr": "10.10.2.91", "status": ["OK"], "timestamp":
"1502799663530", "value": "[OK]"}
```

#### svcRedisCred

([ServiceStatus](#)) Redis Credentials service status on the node. Example:

```
"svcRedisCred": {"ipaddr": "10.10.2.91", "status": ["OK"], "timestamp":
"1502799663530", "value": "[OK]"}
```

*svcRedisMon*

([ServiceStatus](#)) Redis Monitor service status on the node. Example:

```
"svcRedisMon": {"ipaddr": "10.10.2.91", "status": ["OK"], "timestamp":  
"1502799663530", "value": " [OK] "}
```

*svcRedisQos*

([ServiceStatus](#)) Redis QoS service status on the node. Example:

```
"svcRedisQos": {"ipaddr": "10.10.2.91", "status": ["OK"], "timestamp":  
"1502799663530", "value": " [OK] "}
```

*svcS3*

([ServiceStatus](#)) S3 service status on the node. Example:

```
"svcS3": {"ipaddr": "10.10.2.91", "status": ["OK"], "timestamp":  
"1502799663530", "value": " [OK] "}
```

#### 12.6.13.4. MonitorSystemInfo Object

The *MonitorSystemInfo* object consists of the following attributes:

**Note** Within the *MonitorSystemInfo* object, all statistics for which the data type (in the descriptions below) is *MonitorStat* are formatted as "*<statName>:{'timestamp': '<UTCMilliseconds>', 'value': '<statValue>'}*".

*diskAvailKb*

([MonitorStat](#)) Across the whole service region, the total mounted disk space that's still available for S3 object storage (HyperStore data directories) or Cassandra metadata storage (Cassandra data directory). Reported as a number of kilobytes.

The following are deducted from the total amount of unused disk space to arrive at the "available" disk space amount that is reported by the *diskAvailKb* statistic:

- Each disk's "reserved-blocks-percentage" (the portion of the disk that's reserved for privileged processes)
- Each HyperStore data disk's "stop-write" buffer (10% of capacity by default). For more information on the stop-write feature see "**Automatic Stop of Writes to a Disk at 90% Usage**" (page 157).

Example:

```
"diskAvailKb": {"timestamp": "1502799843254", "value": "61855592"}
```

**Note** Because the reserved-blocks-percentage and the stop-write buffer percentage are not counted as available space, the *diskUsedKb* and *diskAvailKb* will add up to less than the *diskTotalKb*.

**Note** The *diskAvailKb* value can potentially be larger than a 64 bit integer can hold.

*diskTotalKb*

([MonitorStat](#)) Across the whole service region, the total size of the disks mounted for S3 object storage (HyperStore data directories) or Cassandra metadata storage (Cassandra data directory). Reported as a number of kilobytes.

Example:

```
"diskTotalKb": {"timestamp": "1502799843254", "value": "88115680"}
```

**Note** This value can potentially be larger than a 64 bit integer can hold.

*diskUsedKb*

([MonitorStat](#)) Across the whole service region, on disks that are mounted for S3 object storage (HyperStore data directories) or Cassandra metadata storage (Cassandra data directory), the total disk space that's used. Reported as a number of kilobytes.

Example:

```
"diskUsedKb": {"timestamp": "1502799843254", "value": "23814368"}
```

**Note** This value can potentially be larger than a 64 bit integer can hold.

*nodeStatuses*

(List<*NodeStatus*>) List of *NodeStatus* objects, one for each node in the service region. Each nested *NodeStatus* object consists of the following attributes:

*hostname*

(String) Hostname of the node. Example:

```
"hostname": "store1"
```

*ipaddr*

(String) This attribute will have value *null*. Example:

```
"ipaddr": null
```

*status*

(List<string>) Status of the node. A list of one or more of the following strings: "OK", "SVC\_DOWN", "LOG\_WARN", or "LOG\_ERR". The meanings are:

- "OK" — All HyperStore services are up and running on the node, and the node has no unacknowledged events.
- "SVC\_DOWN" — One or more HyperStore services (Admin, Cassandra, HyperStore, Redis QoS, Redis Credentials, Redis Monitor, or S3) is down on the node
- "LOG\_WARN" — There are unacknowledged warnings in an application log on the node (such as the S3 Service application log or the Cassandra application log).
- "LOG\_ERR" — There are unacknowledged errors in an application service log.

Example:

```
"status": ["SVC_DOWN", "LOG_WARN"]
```

*timestamp*

(String) Status timestamp in UTC milliseconds. Example:

```
"timestamp": "1502799843254"
```

*value*

(String) The "value" attribute will be the same as the "status" attribute, except formatted as a single string rather than a list of strings. Example:

```
"value": "[SVC_DOWN, LOG_WARN]"
```

*s3GetLatency*

([MonitorStat](#)) Across the whole service region, the 95th percentile request latency value for S3 GET transactions, in milliseconds. New statistic values are calculated each five minutes, based on the most recent approximately 1000 GET transactions. The s3GetLatency value indicates that of the last 1000 GET transactions, 95% completed in that many milliseconds or less.

Example:

```
"s3GetLatency": null
```

**Note** HEAD transactions are counted toward this stat also.

*s3GetTPS*

([MonitorStat](#)) Across the whole service region, the number of S3 GET transactions processed per second. This stat is recalculated each five minutes, based on the most recent five minutes of transaction data. HEAD transactions are counted toward this stat also.

Example:

```
"s3GetTPS": null
```

*s3GetThruput*

([MonitorStat](#)) Across the whole service region, the data throughput for S3 GET transactions, expressed as MB per second. This stat is recalculated each five minutes, based on the most recent five minutes of transaction data. HEAD transactions are counted toward this stat also.

Example:

```
"s3GetThruput": {"timestamp": "1502799843254", "value": "0"}
```

*s3PutLatency*

([MonitorStat](#)) Across the whole service region, the 95th percentile request latency value for S3 PUT transactions, in milliseconds. New statistic values are calculated each five minutes, based on the most recent approximately 1000 PUT transactions. The s3PutLatency value indicates that of the last 1000 PUT transactions, 95% completed in that many milliseconds or less.

Example:

```
"s3PutLatency": {"timestamp": "1502799843254", "value": "130.1"}
```

**Note** POST transactions are counted toward this stat also.

*s3PutTPS*

([MonitorStat](#)) Across the whole service region, the number of S3 PUT transactions processed per second. This stat is recalculated each five minutes, based on the most recent five minutes of transaction data. POST transactions are counted toward this stat also.

Example:

```
"s3PutTPS": {"timestamp": "1502799843254", "value": "0.0"}
```

*s3PutThruput*

([MonitorStat](#)) Across the whole service region, the data throughput for S3 PUT transactions, expressed as MB per second. This stat is recalculated each five minutes, based on the most recent five minutes of transaction data. POST transactions are counted toward this stat also.

Example:

```
"s3PutThruput": {"timestamp": "1502799843254", "value": "0"}
```

*status*

(ServiceStatus) High-level service status for the system as a whole. Formatted as {"ipaddr": "<empty>", "status": [<one of "OK" or "SVC\_DOWN">], "timestamp": "<UTCMilliseconds>", "value": "<statusValueFormattedAsString>"}.

The "ipaddr" value will be empty or null.

The "status" will be formatted as a list but with just one member string. Status string meanings are:

- "OK" — All HyperStore services are up and running on all nodes in the service region.
- "SVC\_DOWN" — A HyperStore service (Admin, Cassandra, HyperStore, Redis QoS, Redis Credentials, Redis Monitor, or S3) is down on one or more nodes in the service region.

The "value" attribute will be identical to the "status" attribute, except formatted as a string rather than as a list.

Example:

```
"status": {"ipaddr": "", "status": ["OK"], "timestamp": "1502799843254",  
"value": "[OK]"}
```

### 12.6.13.5. NotificationRule Object

The *NotificationRule* object consists of the following attributes:

*condition (mandatory)*

(String) Comparator used in defining this rule: can be ">", "<", or "=". Example:

```
"condition": ">"
```

*conditionVal (mandatory)*

(String) Value against which to compare. The value of the statistic specified by statId will be compared to the conditionVal to determine whether a notification is called for. This statistic monitoring occurs on each node.

For example for a rule that triggers notifications if CPU utilization on any individual node exceeds 90%, the "statId" would be "cpu", the "condition" would be ">", and the "conditionVal" would be ".9".

Example:

```
"conditionVal": "0.9"
```

*email (optional)*

(String) Comma-separated list of email addresses to receive notifications. Or to use the default email address list (as configured on the CMC's **"Configuration Settings"** (page 337) page), set this to the string "default".

To not have email notifications as part of the rule (for instance, if the rule is only meant to trigger SNMP traps), set the "email" attribute to empty ("").

This defaults to empty (") if the attribute is omitted in a *PUT /monitor/notificationrule* request.

Example:

```
"email": "default"
```

*enabled (mandatory)*

(Boolean) Whether the rule is enabled, *true* or *false*.

This attribute defaults to *false* if the attribute is omitted in a *PUT /monitor/notificationrule* request.

Example:

```
"enabled": true
```

*region (optional)*

(String) In a *PUT /monitor/notificationrule* request, use this attribute to specify the service region in which to create the notification rule. To create the rule in the default region set this attribute to the default region name or to empty ("). If you omit the "region" attribute in a PUT, then by default the notification rule is created in the default region.

In a *GET /monitor/notificationrules* response, the "region" attribute will be present but set to empty. The client application will be aware of what region the retrieved rules are from because the desired region is specified in the GET request line.

Example:

```
"region": ""
```

*ruleId (mandatory)*

(String) System-generated unique ID for this rule. For the default notification rules that come packaged with the HyperStore system this will be a simple integer like "1", "2", or "14". For rules that you create yourself the system will generate a ruleId in the form of a UUID string like "8e4cc533-360a-4dd5-bfe4-6b5f5b6c40da".

In a PUT (when you are creating a new rule), include the "ruleId" attribute and set it to empty ("). The system will subsequently generate a ruleId upon rule creation.

In a POST (when you are updating an existing rule), set the "ruleId" attribute to the ruleId of the rule you want to update. To find out what the ruleId is for a particular rule, use the *GET /monitor/notificationrules* method.

Example:

```
"ruleId": "12"
```

*severityLevel* (**mandatory**)

(Integer) Severity level to assign to the event. This is an integer with meaning as follows:

- 0 = Low
- 1 = Medium
- 2 = High
- 3 = Critical

Example:

```
"severityLevel": 1
```

*snmpTrap* (**optional**)

(Boolean) Whether to transmit an SNMP trap as part of the notification when this rule is triggered, *true* or *false*. If a trap is sent it is sent to the destination configured on the CMC's **"Configuration Settings"** (page 337) page.

This defaults to false if the attribute is omitted in a *PUT /monitor/notificationrule* request.

Example:

```
"snmpTrap": false
```

*statId* (**mandatory**)

(String) ID of the node statistic being checked for this rule. The table below lists statistics for which notification rules can be defined. These statistics are monitored on **each node**. Note that the sample "conditionVal" column is not intended to suggest suitable values upon which to base notification rules but simply to illustrate the applicable data format. The right-most column indicates whether a rule for that *statId* already exists, as part of the default set of notification rules that come pre-packaged with the HyperStore system.

Example:

```
"statId": "cpu"
```

statId	Description	Appropriate "condition"	Sample "conditionVal"	Pre-Packaged Rule Exist?
s3GetTPS	Number of S3 GET transactions per second on the node.	">"	"100"	No
s3PutTPS	Number of S3 PUT transactions per second on the node.	">"	"100"	No
s3PutThruput	Number of bytes of throughput per second for S3 PUT operations on the node.	">"	"100000"	No
s3GetThruput	Number of bytes of throughput per second for S3 GET operations on the node.	">"	"100000"	No
s3PutLatency	Recent average latency for S3 PUT operations on the	">"	"100"	No

statId	Description	Appropriate "condition"	Sample "conditionVal"	Pre-Packaged Rule Exist?
	node, in number of milliseconds.			
s3GetLatency	Recent average latency for S3 GET operations on the node, in number of milliseconds.	">"	"100"	No
diskAvail	Of the node's total disk space allocated to HyperStore data storage, the portion of disk space that's still free. Expressed as a decimal value.	"<"	".1"	Yes, for < .1
diskInfo	On each individual disk that is allocated to HyperStore data storage, the portion of disk space that's still free. Expressed as a decimal. This rule triggers a notification if any individual disk on the node crosses the defined threshold.	"<"	".15"	Yes, for < .15
repairCompletionStatus	When this type of rule is set, a notification is triggered any time that an auto-repair completes. The notification includes the auto-repair's final status: COMPLETED, FAILED, or TERMINATED.	Set to empty ("")	Set to empty ("")	Yes
cpu	Current CPU utilization level as a decimal value.	">"	".9"	Yes, for > .9
ioRx	Total network bytes per second received by the node (S3 traffic plus any other network traffic to the node).	">"	"100000000"	No
ioTx	Total network bytes per second transmitted by the node (S3 traffic plus any other network traffic from the node).	">"	"100000000"	No
diskIORead	Bytes per second for disk reads on the node.	">"	"1000000"	No
diskIOWrite	Bytes per second for disk writes on the node.	">"	"1000000"	No
svcAdmin	Admin service status. One of	"="	"SVC_DOWN"	Yes, one

statId	Description	Appropriate "condition"	Sample "conditionVal"	Pre-Packaged Rule Exist?
	{SVC_DOWN, LOG_WARN, LOG_ERR}. Note that for this and the other "svc<ServiceType>" statIds, you have the option of creating multiple rules — for example, one rule for status "SVC_DOWN" and a second separate rule for status "LOG_ERR". Do not specify multiple service values in a single notification rule.			rule for SVC_DOWN and one rule for LOG_ERR
svcCassandra	Cassandra service status. One of {SVC_DOWN, LOG_WARN, LOG_ERR}.	"="	"SVC_DOWN"	Yes, one rule for SVC_DOWN and one rule for LOG_ERR
svcHyperStore	HyperStore service status. One of {SVC_DOWN, LOG_WARN, LOG_ERR}.	"="	"SVC_DOWN"	Yes, one rule for SVC_DOWN and one rule for LOG_ERR
svcRedisCred	Redis Credentials service status. One of {SVC_DOWN, LOG_WARN}.	"="	"SVC_DOWN"	Yes, one rule for SVC_DOWN and one rule for LOG_WARN
svcRedisQos	Redis QoS service status. One of {SVC_DOWN, LOG_WARN}.	"="	"SVC_DOWN"	Yes, one rule for SVC_DOWN and one rule for LOG_WARN
svcRedisMon	Redis Monitor service status.	"="	"SVC_DOWN"	Yes, for

statId	Description	Appropriate "condition"	Sample "conditionVal"	Pre-Packaged Rule Exist?
	Only supported value is "SVC_DOWN".			SVC_DOWN
svcS3	<p>S3 service status. One of {SVC_DOWN, LOG_WARN, LOG_ERR}.</p> <div> <p><b>Note</b> Along with log warnings and errors pertaining to providing S3 service to clients, the S3 service alerts category includes log warnings and errors pertaining to auto-tiering and cross-region replication.</p> </div>	"="	"SVC_DOWN"	Yes, one rule for SVC_DOWN and one rule for LOG_ERR

#### 12.6.13.6. NotificationRulesEnable Object

The *NotificationRulesEnable* object consists of the following attributes:

##### *enable*

(Mandatory, boolean) Set to *true* to enable the rule(s). Set to *false* to disable the rule(s). Example:

```
"enable":false
```

##### *regionId*

(Optional, string) Service region in which the rules are configured. If you do not specify a region, the default region is assumed. Example:

```
"regionId":""
```

##### *ruleList*

(Mandatory, list<string>) List of ruleId(s) of the notification rule(s) to enable or disable.

If you do not know the ruleIds of the rules that you want to enable/disable, you can retrieve them by using the [GET /monitor/notificationrules](#) method. That method returns a list of rules, which includes each rule's ruleId.

Example:

```
"ruleList":["836da4bf-c6cc-4f73-afa3-9854ce407ca6",
"8ef63b63-4961-4e17-88c7-d53c966557db"]
```

#### 12.6.13.7. MonitorStat

Data type with the following format:

```
"<statName>": {"timestamp": "<UTCMilliseconds>", "value": "<statValue>"}
```

### 12.6.13.8. ServiceStatus

Data type with the following format:

```
{"ipaddr": "<nodeIPAdress>", "status": [<list of one or more of "OK", "SVC_DOWN", "LOG_WARN", or "LOG_ERR">], "timestamp": "<UTCMilliseconds>", "value": "<statusValueFormattedAsString>"}
```

The meanings of the possible "status" values are:

- "OK" — The service is up and running on the node, and there are no unacknowledged events associated with the service.
- "SVC\_DOWN" — The service is down on the node.
- "LOG\_WARN" — There are unacknowledged warnings in the service application log on the node.
- "LOG\_ERR" — There are unacknowledged errors in the service application log on the node.

The "value" attribute will be identical to the "status" attribute, except formatted as a single string rather than a list of strings. For example:

```
"status": ["SVC_DOWN", "LOG_WARN"], "value": "[SVC_DOWN, LOG_WARN]"
```

## 12.7. permissions

The Admin API methods built around the **permissions** resource are for creating, changing, or retrieving public URL permissions for an object that's stored in the HyperStore system. When you create a public URL for an object, the object can then be accessed at that URL by a regular web browser.

### 12.7.1. GET /permissions/publicUrl

**GET /permissions/publicUrl** Get public URL permissions for an object

The request line syntax for this method is as follows.

```
POST /permissions/publicUrl?userId=xxx&groupId=xxx&bucketName=xxx&objectName=xxx
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"permissions Query Parameters"** (page 821).

There is no request payload.

Use this method to retrieve existing public URL permissions for an object (public URL permissions that have already been created with the [POST /permissions/publicUrl](#) method).

#### 12.7.1.0.1. Example Using cURL

The [example](#) below retrieves an existing public URL for an object named *Cloudian.pdf*.

```
curl -X GET -k -u sysadmin:public \
'https://
localhost
:19443/permissions/publicUrl?userId=
jim&groupId=Pubs&bucketName=bkt1&objectName=Cloudian.pdf' \
| python -mjson.tool
```

The response payload is a JSON-formatted *PublicUrlAccess* object, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"PublicUrlAccess Object"** (page 822).

```
{
  "allowRead": true,
  "currentDownloads": 0,
  "expiryTime": "1517385600000",
  "maxDownloadNum": 1000,
  "secure": true,
  "url": "https://s3-region1.mycloudianhyperstore.com/bkt1/Cloudian.pdf?
AWSAccessKeyId=00b3ec480eb5c844fb88&Expires=1517385600&Signature=
rxxJnEWoUusrj1kQ02A9PMcFQ4U%3D&x-amz-pt=MDAzMTE1NjIxNTE2MTE4NzIxMDc1"
}
```

### 12.7.1.0.2. Response Format

The response payload is a JSON-formatted *PublicUrlAccess* object (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameter : {userId, groupId, bucketName, objectName}
400	User does not exist

## 12.7.2. POST /permissions/publicUrl

**POST /permissions/publicUrl** Create or change public URL permissions for an object

The request line syntax for this method is as follows.

```
POST /permissions/publicUrl?userId=xxx&groupId=xxx&bucketName=xxx&objectName=xxx
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"permissions Query Parameters"** (page 821).

The required request payload is a JSON-formatted *PublicUrlAccess* object. See example below.

Use this method to create or update public URL permissions for an object that's stored in the HyperStore system. See the Example section below for the distinction between creating a new public URL and updating an existing one.

For this method to work, the object owner must also be the bucket owner. Also, the method will not work if the object has been encrypted using a user-managed encryption key (SSE-C).

The public URL for an object will have the following format:

```
http[s]://<bucketName>.<S3Domain>/<objectName>?AWSAccessKeyId=<accessKeyOfObjectOwner>
&Expires=<expiryTime>&Signature=<SignatureString>&x-amz-pt=<>
```

This format follows the AWS specification for signed URLs.

### 12.7.2.0.1. Example Using cURL

The [example](#) below creates a public URL for an object named *Cloudian.pdf*. In this example the JSON-formatted *PublicUrlAccess* object is specified in a text file named *postPublicUrlAccess.txt* which is then referenced as the data input to the cURL command.

```
curl -X POST -H "Content-Type: application/json" -k -u sysadmin:public \
-d @postPublicUrlAccess.txt \
'https://
localhost
:19443/permissions/publicUrl?userId=
jim&groupId=Pubs&bucketName=bkt1&objectName=Cloudian.pdf'
```

The *postPublicUrlAccess.txt* file content in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"PublicUrlAccess Object"** (page 822).

```
{
  "allowRead": true,
  "expiryTime": "1517385600000",
  "maxDownloadNum": 1000,
  "secure": true
}
```

**Note** If the *PublicUrlAccess* JSON object supplied in the POST request body does not include a "url" attribute -- as it does not in the example above -- the POST request is processed as a request to generate a **new** public URL. If a "url" attribute value is included in the *PublicUrlAccess* object and set to equal an existing public URL, the POST request is processed as an update to the permissions associated with the existing public URL (such as an update of the expiration date-time or the maximum allowed downloads limit).

To retrieve a public URL that you've just created or that you've created previously, use the [GET /permissions/publicUrl](#) method.

### 12.7.2.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameter : {userId, groupId, bucketName, objectName}
400	User does not exist
400	Invalid JSON object

## 12.7.3. permissions Query Parameters

*userId*

(Mandatory, string) User ID for user who owns the object for which a public URL is being generated.

*groupId*

(Mandatory, string) Group ID for user who owns the object.

*bucketName*

(Mandatory, bucketname) S3 bucket that contains the object. Note that the bucket's owner must be the same as the object owner or the request will be rejected with a 400 error response.

*objectName*

(Mandatory, string) Name of the object for which a public URL is being generated.

## 12.7.4. permissions Objects

This section describes the JSON-formatted data objects that constitute the request or response payloads for the Permissions related Admin API methods.

**Note** For examples of these objects see the API method request and response examples.

- **"PublicUrlAccess Object"** (page 822)

### 12.7.4.1. PublicUrlAccess Object

*allowRead*

(Optional, boolean) Whether a public URL is enabled for the associated object, *true* or *false*. Defaults to *true*. Example:

```
"allowRead": true
```

*currentDownloads*

(Optional, number) Current total number of times that the object has been downloaded via public URL. This value is set by the system, not by the client. Starts at 0 for a new public URL. Example:

```
"currentDownloads": 0
```

*expiryTime*

(Mandatory, string) Expiration date-time of the public URL in UTC milliseconds. After this date-time the public URL will no longer work. Example:

```
"expiryTime": "1517385600000"
```

*maxDownloadNum*

(Optional, number) Maximum number of times that the system will allow the object to be downloaded via public URL, by all users in total. To allow unlimited downloads, set this to "-1". Defaults to 1000. Example:

```
"maxDownloadNum": 1000
```

*secure*

(Optional, boolean) Whether the object's public URL should use HTTPS rather than HTTP, *true* or *false*. Defaults to *true*. Example:

```
"secure": true
```

*url*

(Optional, string) System-generated public URL for accessing the object.

With a POST request:

- To create a new public URL for an object do not include the "url" attribute in the request body.
- To change permission attributes for an existing public URL, use the "url" attribute in the request body to specify the existing public URL.

The public URL for an object will have the following format:

```
http[s]://<bucketName>.<S3Domain>/<objectName>?AWSAccessKeyId=
<accessKeyOfObjectOwner>&Expires=<expiryTime>&Signature=<signatureString>&
x-amz-pt=<internalTrackingCode>
```

This format follows the AWS specification for signed URLs.

Example:

```
"url": "https://s3-region1.mycloudianhyperstore.com/bkt1/Cloudian.pdf?
AWSAccessKeyId=00b3ec480eb5c844fb88&Expires=1517385600&Signature=
rxxJnEWoUusrj1kQ02A9PMcFQ4U%3D&x-amz-pt=MDAzMTE1NjIxNTE2MTE4NzIxMDc1"
```

## 12.8. qos

The Admin API methods built around the **qos** resource are for managing HyperStore quality of service (QoS) controls. These controls set limits on service usage by user groups and by individual users. There are API methods for assigning, retrieving, or deleting QoS settings for specified users or groups.

For an overview of the HyperStore quality of service feature, see "**Quality of Service (QoS) Feature Overview**" (page 135).

### 12.8.1. DELETE /qos/limits

#### DELETE /qos/limits Delete QoS settings for a user or group

The request line syntax for this method is as follows.

```
DELETE /qos/limits?userId=xxx&groupId=xxx [&region=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see "**qos Query Parameters**" (page 827).

There is no request payload.

Use this method to:

- Delete QoS limits that have been assigned to a specific user. If you delete user-specific QoS limits, the system will automatically assign the user the default user-level QoS limits associated with the group to which the user belongs.
- Delete QoS limits that have been assigned to a specific group. If you delete group-specific QoS limits, the system will automatically assign the group the regional default QoS limits.

Essentially, this method allows you to clear user-specific or group-specific QoS overrides so that default QoS settings are used instead.

### 12.8.1.0.1. Example Using cURL

The [example](#) below deletes QoS settings for the "Dev" group. With these group-specific settings deleted, the default group QoS settings for the service region will be applied to the Dev group.

```
curl -X DELETE -k -u sysadmin:public \
'https://localhost:19443/qos/limits?userId=* & groupId=Dev'
```

### 12.8.1.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameters : {userId, groupId}
400	Region {region} is invalid

## 12.8.2. GET /qos/limits

**GET /qos/limits** Get QoS settings for a user or group

The request line syntax for this method is as follows.

```
GET /qos/limits?userId=xxx&groupId=xxx [&region=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"qos Query Parameters"** (page 827).

There is no request payload.

**Note** The HyperStore IAM Service supports a role-based access version of this API method as described in **"RBAC Version of this Method"** (page 825).

### 12.8.2.0.1. Example Using cURL

The [example](#) below retrieves current QoS settings for the user "cody" in the "Dev" group.

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/qos/limits?userId=cody & groupId=Dev' \
| python -mjson.tool
```

The response payload is a JSON-formatted *QosLimitSettings* object, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"QosLimitSettings Object"** (page 829).

```
{
  "groupId": "Dev",
  "labelId": "qos.userQosOverrides.title",
  "qosLimitList": [
    {
      "type": "STORAGE_QUOTA_KBYTES",
      "value": 5000000
    }
  ]
}
```

```

    },
    {
      "type": "REQUEST_RATE_LW",
      "value": -1
    },
    {
      "type": "REQUEST_RATE_LH",
      "value": -1
    },
    {
      "type": "DATAKBYTES_IN_LW",
      "value": -1
    },
    {
      "type": "DATAKBYTES_IN_LH",
      "value": -1
    },
    {
      "type": "DATAKBYTES_OUT_LW",
      "value": -1
    },
    {
      "type": "DATAKBYTES_OUT_LH",
      "value": -1
    },
    {
      "type": "STORAGE_QUOTA_COUNT",
      "value": -1
    }
  ],
  "userId": "cody"
}

```

#### 12.8.2.0.2. Response Format

The response payload is a JSON-formatted *QosLimitSettings* object (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	User does not exist
400	Missing required parameter : {userId, groupId}
400	Region {region} is invalid

#### 12.8.2.1. RBAC Version of this Method

HyperStore's IAM Service supports an RBAC version of this API method. For an overview of the RBAC feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- Action name: *GetCloudianQosLimits*
- Parameters: Same as for *GET /qos/limits*, except parameter names start with an upper case letter rather than lower case
- Response body: Same response data as for *GET /qos/limits* except the data is formatted in XML rather than JSON
- Role-based restrictions:
  - HyperStore system admin user can get QoS limits for any group or user
  - HyperStore group admin user can only get QoS limits for own group or users within own group
  - HyperStore regular user can only get own QoS limits
  - IAM user can only use this method if granted *admin:GetCloudianQosLimits* permission by an IAM policy, and subject to the same restriction as the parent HyperStore user.

**Note** The "GetCloudianQosLimits" action retrieves QoS limit information for Cloudian HyperStore user accounts, not for subsidiary IAM users. The system does not maintain QoS limits per IAM user. For example, if a HyperStore group administrator grants *admin:GetCloudianQosLimits* permission to an IAM user, the IAM user will be able to retrieve QoS limits for **any HyperStore user in the group administrator's group**. And if a HyperStore regular user grants *admin:GetCloudianQosLimits* permission to an IAM user, the IAM user will be able to retrieve QoS limits for the **parent HyperStore user**.

- Sample request and response (abridged):

```
REQUEST

http://localhost:16080/?Action=GetCloudianQosLimits&UserId=cody&GroupId=Dev

<request headers including authorization info>

RESPONSE

200 OK

<?xml version="1.0" encoding="utf-8"?>
<GetCloudianQosLimitsResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">
  <ResponseMetadata>
    <RequestId>system-generated-request-id</RequestId>
  </ResponseMetadata>
  <UserQosOverrides>
    <groupId>Pubs</groupId>
    etc...
    ...
    ...
  </UserQosOverrides>
</GetCloudianQosLimitsResponse>
```

### 12.8.3. POST /qos/limits

**POST /qos/limits**    Create QoS settings for a user or group

The request line syntax for this method is as follows.

```
POST /qos/limits?userId=xxx&groupId=xxx&storageQuotaKBytes=xxx&storageQuotaCount=xxx
&wIRequestRate=xxx&hIRequestRate=xxx&wIDataBytesIn=xxx&hIDataBytesIn=xxx
&wIDataBytesOut=xxx&hIDataBytesOut=xxx [ &region=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"qos Query Parameters"** (page 827).

You must include each of the QoS type query parameters, even those types for which you do not want to set a limit. To disable a type set it to "-1". There is no request body payload.

This method creates user-level or group-level QoS settings. User-level QoS settings place an upper limit on the storage utilization and transaction activities of individual users, while group-level QoS settings place such limits on entire user groups.

In a multi-region HyperStore deployment, you must establish QoS limits separately for each region. The QoS limits that you establish for a region will be applied only to activity in that region.

**Note** For the system to **enforce** QoS limits that you have assigned to users or groups, the QoS feature must be enabled in your system configuration. By default it is disabled. You can enable it in the CMC's [Configuration Settings](#) page. Note that you can enable QoS enforcement just for storage utilization limits, or for storage utilization limits and also request traffic limits.

#### 12.8.3.0.1. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameters : {userId, groupId, storageQuotaBytes, storageQuotaCount, wIRequestRate, hIRequestRate, wIDataBytesIn, hIDataBytesIn, wIDataBytesOut, hIDataBytesOut}
400	Invalid parameter
400	Region {region} is invalid

### 12.8.4. qos Query Parameters

#### *userId*

(Mandatory, string) User ID of the user to whom the QoS settings apply. Supported options are a specific user ID, "ALL", or "".

See the [groupId](#) description below for information about how to use the "userId" and "groupId" parameters in combination to designate different levels of QoS settings.

#### *groupId*

(Mandatory, string) Group ID of the group to which the QoS settings apply. Supported options are a specific group ID, "ALL", or "".

You can use the "userId" and "groupId" parameters in combination to designate different levels of QoS settings:

- User-level QoS for a specific user (userId=<userId>&groupId=<groupId>)
- Default user-level QoS for a specific group (userId=ALL&groupId=<groupId>)
- Default user-level QoS for the whole region (userId=ALL&groupId=\*)
- Group-level QoS for a specific group (userId=\*&groupId=<groupId>)
- Default group-level QoS for the whole region (userId=\*&groupId=ALL)

#### *region*

(Optional, string) Service region to which the QoS settings apply. If not specified, the default region is assumed.

#### *storageQuotaKBytes*

(Mandatory, number) With a *POST /qos/limits* request: Maximum allowed storage, in number of kilobytes.

Storage overhead associated with replication or erasure coding does not count toward this limit. For example a 100KB object that's replicated three times (in accordance with a 3X replication storage policy) would count as 100KB toward this limit, not 300KB.

#### *storageQuotaCount*

(Mandatory, number) With a *POST /qos/limits* request: Maximum allowed storage, in number of objects.

#### *wlRequestRate*

(Mandatory, number) With a *POST /qos/limits* request: Warning level for number of HTTP requests per minute. Exceeding this level triggers the writing of a message to the S3 Service application log (it does not return a warning to the S3 client).

#### *hlRequestRate*

(Mandatory, number) With a *POST /qos/limits* request: Maximum allowed number of HTTP requests per minute.

#### *wlDataKBytesIn*

(Mandatory, number) With a *POST /qos/limits* request: Warning level for number of uploaded kilobytes per minute. Exceeding this level triggers the writing of a message to the S3 Service application log (it does not return a warning to the S3 client).

#### *hlDataKBytesIn*

(Mandatory, number) With a *POST /qos/limits* request: Maximum allowed number of uploaded kilobytes per minute.

#### *wlDataKBytesOut*

(Mandatory, number) With a *POST /qos/limits* request: Warning level for number of downloaded kilobytes per minute. Exceeding this level triggers the writing of a message to the S3 Service application log (it does not return a warning to the S3 client).

#### *hlDataKBytesOut*

(Mandatory, number) With a *POST /qos/limits* request: Maximum allowed number of downloaded

kilobytes per minute.

## 12.8.5. qos Objects

This section describes the JSON-formatted data objects that constitute the request or response payloads for the QoS related Admin API methods.

**Note** For examples of these objects see the API method request and response examples.

- **"QosLimitSettings Object"** (page 829)

### 12.8.5.1. QosLimitSettings Object

The *QosLimitSettings* object consists of the following attributes and nested objects:

#### *groupId*

(String) Group ID. This will be either a specific group ID, or "ALL", or "\*". For details of how this is used, see **"qos Query Parameters"** (page 827).

Example:

```
"groupId": "Dev"
```

#### *labelId*

(String) This attribute is used by the CMC to display the correct title on a group or user QoS configuration screen. Example:

```
"labelId": "qos.userQosOverrides.title"
```

#### *qosLimitList*

(List<*QosLimit*>) List of *QosLimit* objects. There will be one *QosLimit* object for each of the eight QoS limit types. Each *QosLimit* object consists of the following attributes:

#### *type*

(String) One of the following QoS limit types:

- STORAGE\_QUOTA\_KBYTES
- STORAGE\_QUOTA\_COUNT
- REQUEST\_RATE\_LW
- REQUEST\_RATE\_LH
- DATAKBYTES\_IN\_LW
- DATAKBYTES\_IN\_LH
- DATAKBYTES\_OUT\_LW
- DATAKBYTES\_OUT\_LH

For descriptions of these types see **"QoS Limit Type Descriptions"** (page 830).

Example:

```
"type": "STORAGE_QUOTA_KBYTES"
```

#### *value*

(Number) The value assigned to this QoS limit type. A value of -1 indicates that the limit is disabled. Example:

```
"value": 5000000
```

*userId*

(String) User ID. This will be either a specific user ID, or "ALL", or "". For details of how this is used, see **"qos Query Parameters"** (page 827). Example:

```
"userId": "cody"
```

### 12.8.5.1.1. QoS Limit Type Descriptions

The table below describes each QoS limit type, as it applies to the individual user level and to the group level.

**Note** 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.

**Note** Requests rejected due to QoS limits **are** counted toward usage tracking, for purposes of request volume based billing. For example, if a user has reached her storage quota and tries to do a PUT of more data, the system rejects the PUT request but counts the request toward the user's bill (specifically, toward the part of her rating plan that charges per volume of HTTP PUT requests).

QoS Limit Type	Description	Implementation
STORAGE_QUOTA_KBYTES	Storage quota limit, in number of KBs	<ul style="list-style-type: none"> <li>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.</li> <li>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.</li> </ul>
STORAGE_QUOTA_COUNT	Storage quota limit, in total number of objects. Note that folders count as objects, as well as files	<ul style="list-style-type: none"> <li>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.</li> <li>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.</li> </ul>

QoS Limit Type	Description	Implementation
REQUEST_RATE_LW	Request rate warning limit, in total number of HTTP requests per minute.	<ul style="list-style-type: none"> <li>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, an informational message is written to the S3 Server's application log. 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.</li> <li>For group QoS — The implementation is the same as for user QoS, except that it applies to requests from all users in the group. 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.</li> </ul> <p>Note that HTTP DELETE requests are not counted toward Request Rate controls.</p>
REQUEST_RATE_LH	Request rate maximum, in total number of HTTP requests per minute.	<ul style="list-style-type: none"> <li>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. 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.</li> <li>For group QoS — The imple-</li> </ul>

QoS Limit Type	Description	Implementation
		mentation is the same as for user QoS, except that it applies to requests from all users in the group. 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.
DATAKBYTES_IN_LW	Inbound data rate warning limit, in KBs 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 kilobytes of data.
DATAKBYTES_IN_LH	Inbound data rate high limit, in KBs 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 kilobytes of data. Note that if a block is triggered by the Data Bytes IN (KB) High Limit being reached, the block applies to all HTTP request types (not just PUTs.)
DATAKBYTES_OUT_LW	Outbound data rate warning limit, in KBs 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 kilobytes of data.
DATAKBYTES_OUT_LH	Outbound data rate high limit, in KBs 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 kilobytes of data. Note that if a block is triggered by the Data Bytes OUT (KB) High Limit being reached, the block applies to all HTTP request types (not just GETs.)

## 12.9. ratingPlan

The Admin API methods built around the **ratingPlan** resource are for managing HyperStore rating plans. Rating plans assigning pricing to various types and levels of service usage, in support of billing users or charging back to an organization's business units. There are methods for creating, changing, and deleting rating plans.

For an overview of the HyperStore billing feature, see **"Usage Reporting and Billing Feature Overview"** (page 138).

**Note** By default the system only supports billing based on number of stored bytes. If you want your rating plans to include billing based on request rates or data transfer rates you must enable the "Track-/Report Usage for Request Rates and Data Transfer Rates" setting in the CMC's [Configuration Settings](#) page.

## 12.9.1. DELETE /ratingPlan

### DELETE /ratingPlan Delete a rating plan

The request line syntax for this method is as follows.

```
DELETE /ratingPlan?ratingPlanId=xxx
```

For parameter description click on the parameter name or see **"ratingPlan Query Parameters"** (page 839).

There is no request payload.

#### 12.9.1.0.1. Example Using cURL

The [example](#) below deletes a rating plan with ID "Plan-6".

```
curl -X DELETE -k -u sysadmin:public \
https://localhost:19443/ratingPlan?ratingPlanId=Plan-6
```

#### 12.9.1.0.2. Response Format

There is no response payload. For response status code this method will return either one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Rating Plan does not exist
400	Missing required parameter : {ratingPlanId}

## 12.9.2. GET /ratingPlan

### GET /ratingPlan Get a rating plan

The request line syntax is as follows.

```
GET /ratingPlan?ratingPlanId=xxx
```

For parameter description click on the parameter name or see **"ratingPlan Query Parameters"** (page 839).

There is no request payload.

#### 12.9.2.0.1. Example Using cURL

The [example](#) below retrieves the system default rating plan, which has ID "Default-RP".

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/ratingPlan?ratingPlanId=Default-RP python -mjson.tool
```

The response payload is a JSON-formatted *RatingPlan* object, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see "**RatingPlan Object**" (page 839).

```
{
  "currency": "USD",
  "id": "Default-RP",
  "mapRules": {
    "BI": {
      "ruleclassType": "BYTES_IN",
      "rules": [
        {
          "first": "1",
          "second": "0.20"
        },
        {
          "first": "0",
          "second": "0.10"
        }
      ]
    },
    "BO": {
      "ruleclassType": "BYTES_OUT",
      "rules": [
        {
          "first": "1",
          "second": "0.20"
        },
        {
          "first": "0",
          "second": "0.10"
        }
      ]
    },
    "HD": {
      "ruleclassType": "HTTP_DELETE",
      "rules": [
        {
          "first": "0",
          "second": "0"
        }
      ]
    },
    "HG": {
      "ruleclassType": "HTTP_GET",
      "rules": [
        {
          "first": "10",
          "second": "0.02"
        },
        {
```

```

        "first": "0",
        "second": "0.01"
    }
]
},
"HP": {
    "ruleclassType": "HTTP_PUT",
    "rules": [
        {
            "first": "0",
            "second": "0.02"
        }
    ]
},
"SB": {
    "ruleclassType": "STORAGE_BYTE",
    "rules": [
        {
            "first": "1",
            "second": "0.14"
        },
        {
            "first": "5",
            "second": "0.12"
        },
        {
            "first": "0",
            "second": "0.10"
        }
    ]
}
},
"name": "Default Rating Plan"
}

```

#### 12.9.2.0.2. Response Format

The response payload is a JSON-formatted *RatingPlan* object (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
204	Rating Plan does not exist
400	Missing required parameter: {ratingPlanId}

### 12.9.3. GET /ratingPlan/list

**GET /ratingPlan/list** Get the list of rating plans in the system

The request line syntax for this method is as follows.

```
GET /ratingPlan/list
```

There is no request payload.

#### 12.9.3.0.1. Example Using cURL

The [example](#) below retrieves the list of rating plans that are in the system. Note that this method does not return the full content of the rating plans -- just the ID, name, and currency for each plan.

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/ratingPlan/list | python -mjson.tool
```

The response payload in this example is as follows.

```
[
  {
    "currency": "USD",
    "encodedId": "Default-RP",
    "id": "Default-RP",
    "name": "Default Rating Plan"
  },
  {
    "currency": "USD",
    "encodedId": "Whitelist-RP",
    "id": "Whitelist-RP",
    "name": "Whitelist Rating Plan"
  }
]
```

#### 12.9.3.0.2. Response Format

The response payload is in format `List<Map<string,string,string,string>>` (see example above). Strings are: {id,-name,encodedid,currency}. "encodedId" value is a URL-encoding of the "id" value. For response status code this method will return one of the **"Common Response Status Codes"** (page 746).

### 12.9.4. POST /ratingPlan

#### POST /ratingPlan    Change a rating plan

The request line syntax for this method is as follows.

```
POST /ratingPlan
```

The required request payload is a JSON-formatted *RatingPlan* object.

#### 12.9.4.0.1. Example Using cURL

The [example](#) below modifies the rating plan that was created in the [PUT /ratingPlan](#) example. Again the *RatingPlan* object is specified in a text file named *ratingStorageOnly.txt* which is then referenced as the data input to the cURL command.

```
curl -X POST -H "Content-Type: application/json" -k -u sysadmin:public \
-d @ratingStorageOnly.txt https://localhost:19443/ratingPlan
```

Note that in editing the *RatingPlan* object in the *ratingStorageOnly.txt* file before doing the POST operation you could edit any attribute except for the "id" attribute. The "id" attribute must remain the same, so that you're modifying an existing rating plan rather than creating a new one. For an example *RatingPlan* object see [PUT /ratingPlan](#).

#### 12.9.4.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Rating Plan does not exist
400	Missing required parameter : {id, name}
400	Invalid JSON Object

### 12.9.5. PUT /ratingPlan

#### PUT /ratingPlan Create a new rating plan

The request line syntax for this method is as follows.

```
PUT /ratingPlan
```

The required request payload is a JSON-formatted *RatingPlan* object. See example below.

#### 12.9.5.0.1. Example Using cURL

The [example](#) below creates a new rating plan that charges users based only on storage level, with no charges for traffic. In this example the JSON-formatted *RatingPlan* object is specified in a text file named *ratingStorageOnly.txt* which is then referenced as the data input to the cURL command.

```
curl -X PUT -H "Content-Type: application/json" -k -u sysadmin:public \
-d @ratingStorageOnly.txt https://localhost:19443/ratingPlan
```

The *ratingStorageOnly.txt* file content in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"RatingPlan Object"** (page 839).

```
{
  "currency": "USD",
  "id": "Storage-Only",
  "mapRules": {
    "BI": {
      "ruleclassType": "BYTES_IN",
      "rules": [
        {
          "first": "0",
          "second": "0"
        }
      ]
    },
    "BO": {
      "ruleclassType": "BYTES_OUT",
```

```
    "rules": [
      {
        "first": "0",
        "second": "0"
      }
    ]
  },
  "HD": {
    "ruleclassType": "HTTP_DELETE",
    "rules": [
      {
        "first": "0",
        "second": "0"
      }
    ]
  },
  "HG": {
    "ruleclassType": "HTTP_GET",
    "rules": [
      {
        "first": "0",
        "second": "0"
      }
    ]
  },
  "HP": {
    "ruleclassType": "HTTP_PUT",
    "rules": [
      {
        "first": "0",
        "second": "0"
      }
    ]
  },
  "SB": {
    "ruleclassType": "STORAGE_BYTE",
    "rules": [
      {
        "first": "100",
        "second": "0.2"
      },
      {
        "first": "0",
        "second": "0.15"
      }
    ]
  }
},
"name": "Storage Bytes Only"
}
```

### 12.9.5.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameter: {id, name}
400	Invalid JSON Object
409	Unique Constraint Violation : {id}

### 12.9.6. ratingPlan Query Parameters

*ratingPlanId*

(Mandatory, string) Unique identifier of the rating plan.

### 12.9.7. ratingPlan Objects

This section describes the JSON-formatted data objects that constitute the request or response payloads for the Rating Plan related Admin API methods.

**Note** For examples of this object see the API method request and response examples.

- **"RatingPlan Object"** (page 839)

#### 12.9.7.1. RatingPlan Object

The *RatingPlan* object consists of the following attributes and nested objects:

*currency*

(Optional, string) An international currency code (such as "USD", "JPY", or "EUR"). Defaults to "USD".

Example:

```
"currency": "USD"
```

*id*

(Mandatory, string) Unique identifier that you assign to the rating plan. Example:

```
"id": "Default-RP"
```

*mapRules*

(Optional, map<string,*RuleClass*>) Map of rating rules per service dimension. The string is the service dimension acronym. For each <string,*RuleClass*> combination the string is one of "BI" (bytes in), "BO" (bytes out), "HG" (HTTP GETs), "HP" (HTTP PUTs), "HD" (HTTP DELETES), or "SB" (storage bytes). For each service dimension there is a corresponding *RuleClass* object that expresses the rating rules for that service dimension.

Example:

```
"BI": {RuleClass}
```

**Note** You can omit the *mapRules* entirely in the unlikely event that you want to create a rating plan that does not charge for anything. But if you do include a *mapRules* map you must set a `<string,RuleClass>` combination for each of the six service dimensions, including dimensions for which you do not want to charge.

The *RuleClass* object consists of the following attributes and nested objects:

#### *ruleclassType*

(String) Type of rating rule: one of {STORAGE\_BYTE, BYTES\_IN, BYTES\_OUT, HTTP\_GET, HTTP\_PUT, HTTP\_DELETE}. These are the service usage dimensions for which pricing can be set. Example:

```
"ruleclassType": "BYTES_IN"
```

#### *rules*

(List<*Pair*>) List of rating rules to apply to the rule class type. There is one rule (one *Pair* object) per pricing tier.

Each *Pair* object consists of the following attributes:

#### *first*

(String) Rating tier size, as a number of units. In the first *Pair* within a list of *Pair* objects, the "first" attribute specifies the N in the rating rule "The first N units are to be priced at X per unit". (The specific units follow from the service usage type. See **"Service Usage Units"** (page 841) below). In the next *Pair* object in the list, the "first" attribute specifies the N in "The next N units are to be priced at X per unit"; and so on. For your final tier — for pricing additional units above and beyond the already defined tiers — use "0" as the value for the "first" attribute. For an example see the description of "second" below.

#### *second*

(String) For the rating tier specified by the "first" attribute, the rate per unit. The rate is specified as an integer or decimal. (The currency is as specified in the parent *RatingPlan* object.) For example, suppose the currency is set as dollars in the *RatingPlan* object, and you want the first 10 units to be charged at \$2 per unit, the next 10 units to be priced at \$1.50 per unit, and any additional units to be charged at \$1 per unit. Your first *Pair* would be:

```
{"first": "10", "second": "2.00"}
```

Your next *Pair* would be:

```
{"first": "10", "second": "1.50"}
```

Your third and final *Pair* would be:

```
{"first": "0", "second": "1.00"}
```

**Note** For each service dimension that you do not want to charge for, specify just one *Pair* object with both "first" and "second" set to "0".

### 12.9.7.1.1. Service Usage Units

In a *Pair* object, the "first" attribute indicates a number of units constituting a pricing tier, and the "second" attribute indicates the price per unit within that pricing tier. What constitutes a unit depends on the service usage type that the rating rule is being applied to. For illustration suppose that in the examples below, the currency (as specified in the parent *RatingPlan* object) is dollars.

- For **storage bytes (SB)**, the unit is **GB-month** — the average number of GBs of data stored for the month (which is calculated by summing the month's hourly readings of stored bytes, converting to GB, then dividing by the number of hours in the month). So if in a *Pair* object the "first" attribute is set to "5" and the "second" attribute is set to "2", this means that within this pricing tier which spans 5 GB-month, the charge is \$2 per GB-month.
- For **data transfer bytes in or out (BI or BO)**, the unit is number of GBs. So if in a *Pair* object the "first" attribute is set to "5" and the "second" attribute is set to "2", this means that within this pricing tier which spans 5 GBs, the charge is \$2 per GB.
- For **HTTP GETs, PUTs, or DELETEs (HG, HP, or HD)**, the unit is blocks of 10,000 requests. So if in a *Pair* object the "first" attribute is set to "5" and the "second" attribute is set to "2", this means that within this pricing tier which spans 50,000 requests, the charge is \$2 per 10,000 requests.

*name*

(Mandatory, string) Name of rating plan. Example:

```
"name": "Default Rating Plan"
```

## 12.10. system

The Admin API methods built around the **system** resource are for retrieving system information or performing certain system maintenance tasks.

### 12.10.1. GET /system/audit

**GET /system/audit** Get summary counts for system

The request line syntax for this method is as follows.

```
GET /system/audit?[region=xxx]
```

There is no request payload.

**Note** Audit data is automatically updated within the system at the top of each hour. When you call the *GET /system/audit* method you are retrieving the audit data from the most recent hourly update. If you want up-to-the-minute counts -- rather than the counts as of the top of the last hour -- first call the method *POST /system/audit*, with no request body. This updates the counts. Then, you can retrieve the freshly updated counts using the *GET /system/audit* method. If you have a multi-region system and want to update each region's audit data, you would need to submit a separate *POST /system/audit* request to one Admin host in each region. Again, this is necessary only if you want audit data that is fresher than the automatic update done (in every region) at the top of each hour.

12.10.1.0.1. Example Using cURL

The [example](#) below retrieves the summary counts for the system.

```
curl -X GET -k -u sysadmin:public https://localhost:19443/system/audit \
| python -mjson.tool
```

The response payload is a JSON-formatted *AuditData* object, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"AuditData Object"** (page 855).

```
{
  "byteCount": 647687490,
  "bytesInCount": 0,
  "bytesOutCount": 0,
  "licenseExpiration": 1590094491952,
  "nodes": [
    {
      "name": "10.50.50.201"
    },
    {
      "name": "10.50.50.202"
    },
    {
      "name": "10.50.50.203"
    }
  ],
  "objectCount": 13,
  "os": "Linux 3.10.0-957.1.3.el7.x86_64 amd64",
  "tieredBytesCount": 0,
  "timestamp": 1563724800000,
  "userCount": 3
}
```

12.10.1.0.2. Response Format

The response payload is a JSON-formatted *AuditData* object (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or this method-specific status code:

Status Code	Description
400	Region {region} is not valid

12.10.2. GET /system/bucketcount

**GET /system/bucketcount** Get count of buckets owned by a group's members

The request line syntax for this method is as follows.

```
GET /system/bucketcount?groupId=xxx
```

There is no request payload.

### 12.10.2.0.1. Example Using cURL

The [example](#) below retrieves the count of buckets owned by users within the group "testgroup1".

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/system/bucketcount?groupId=testgroup1'
```

The response payload is a text string, which in this example is as follows:

```
5
```

### 12.10.2.0.2. Response Format

The response payload is as shown in the example above. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or this method-specific status code:

Status Code	Description
204	No content

## 12.10.3. GET /system/bucketlist

**GET /system/bucketlist** Get list of buckets owned by a group's members

The request line syntax for this method is as follows.

```
GET /system/bucketlist?groupId=xxx
```

There is no request payload.

### 12.10.3.0.1. Example Using cURL

The [example](#) below retrieves the list of buckets owned by users within the group "testgroup1".

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/system/bucketlist?groupId=testgroup1'
| python -mjson.tool
```

The response payload in this example is as follows:

```
[
  {
    "userId": "testuser1",
    "buckets": [
      {
        "bucketName": "bucket1",
        "createTime": "1554755537223"
      },
      {
        "bucketName": "bucket2",
        "createTime": "1554755542554"
      },
      {
        "bucketName": "bucket3",
        "createTime": "1554755548227"
      }
    ]
  }
]
```

```

    ]
  },
  {
    "userId": "testuser2",
    "buckets": [
      {
        "bucketName": "testbucket4",
        "createTime": "1554755580759"
      },
      {
        "bucketName": "testbucket5",
        "createTime": "1554755587516"
      }
    ]
  }
]

```

#### 12.10.3.0.2. Response Format

The response payload is as shown in the example above. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or this method-specific status code:

Status Code	Description
204	No content

### 12.10.4. GET /system/bytecount

**GET /system/bytecount** Get stored byte count for the system, a group, or a user

The request line syntax for this method is as follows.

```
GET /system/bytecount?groupId=xxx&userId=xxx
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"system Query Parameters"** (page 854).

There is no request payload.

**Note** The byte count is for "net" bytes. Overhead due to replication or erasure coding does not count toward this figure. For example, if a 1MB object is replicated three times in the system (as part of a replication storage policy), this counts as 1MB toward the total byte count -- not as 3MB.

**Note** If you want to retrieve the current byte count for a particular **bucket**, use the [POST /usage/repair/bucket](#) method.

#### 12.10.4.0.1. Examples Using cURL

The [example](#) below retrieves the byte count for the system as a whole (all users in all groups).

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/system/bytecount?groupId=ALL&userId=*
```

The response payload is the byte count in plain text, which in this example is as follows:

```
73836232
```

This next example retrieves the byte count for the "Pubs" group as a whole (all users in the Pubs group).

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/system/bytecount?groupId=Pubs&userId=*
```

The response payload is:

```
542348
```

This next example retrieves the byte count for the user "PubsUser1".

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/system/bytecount?groupId=Pubs&userId=PubsUser1'
```

The response payload is:

```
66712
```

#### 12.10.4.0.2. Response Format

The response payload is plain text (see examples above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or this method-specific status code:

Status Code	Description
400	Missing required parameters: {groupId}, {userId}

### 12.10.5. GET /system/bytestiered

**GET /system/bytestiered** Get tiered byte count for the system, a group, or a user

The request line syntax for this method is as follows.

```
GET /system/bytestiered?groupId=xxx&userId=xxx
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"system Query Parameters"** (page 854).

There is no request payload.

**Note** The response is the total current volume of tiered storage in destination systems **other than HyperStore**. Data auto-tiered from one region to another within a HyperStore system, or from one HyperStore system to another HyperStore system, does not count toward this figure.

#### 12.10.5.0.1. Example Using cURL

The [example](#) below retrieves the tiered volume for the system as a whole (all users in all groups).

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/system/bytestiered?groupId=ALL&userId=*
```

The response payload is the tiered volume as a plain text string, which in this example is as follows:

```
"62G"
```

The tiered volume is expressed as "<n>G" (for number of GBs) or "<n>T" (for number of TBs) or "<n>P" (for number of PBs). If the tiered volume is currently less than 1GB then "0" is returned. If the tiered volume is

12.10.5.0.2. Response Format

The response payload is a plain text string. For response status code this method will return one of the "Common Response Status Codes" (page 746) or this method-specific status code:

Status Code	Description
400	Missing required parameters: {groupId}, {userId}

12.10.6. GET /system/groupbytecount

GET /system/groupbytecount    Get stored byte counts for all of a group's users

The request line syntax for this method is as follows.

```
GET /system/groupbytecount?groupId=xxx [&limit=xxx] [&offset=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see "system Query Parameters" (page 854).

There is no request payload.

**Note** The byte counts are for "net" bytes. Overhead due to replication or erasure coding does not count toward these figures. For example, if a 1MB object is replicated three times in the system (as part of a replication storage policy), this counts as 1MB toward the byte count -- not as 3MB.

12.10.6.0.1. Example Using cURL

The [example](#) below retrieves the stored byte counts for each of the users in the "Pubs" group.

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/system/groupbytecount?groupId=Pubs' | python -mjson.tool
```

The response payload is a JSON-formatted *UserUsage* object, which in this example is as follows. The group has three users in it. If a user has multiple buckets, the user's byte count value is the sum total across all of the user's buckets. For example the user "brady" has a total of 220508 stored bytes. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see "UserUsage Object" (page 860).

```
[
{
  "canonicalUserId": "da870acdd136d60789fb5c761fef4a4a",
  "groupId": "Pubs",
  "usageVal": 220508,
  "userId": "brady"
```

```

},
{
  "canonicalUserId": "9bdcdd44ce1f9266adb9f22a8313feb4",
  "groupId": "Pubs",
  "usageVal": 225365,
  "userId": "gilmore"
},
{
  "canonicalUserId": "9a00529cdfb6496a09c5105913b486ac",
  "groupId": "Pubs",
  "usageVal": 76744,
  "userId": "gronk"
}
]

```

### Response Format

The response payload is a JSON-formatted *UserUsage* object (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or this method-specific status code:

Status Code	Description
400	Missing required parameter: {groupId}
400	Limit should be greater than zero.

## 12.10.7. GET /system/groupobjectcount

**GET /system/groupobjectcount** Get stored object counts for all of a group's users

The request line syntax for this method is as follows.

```
GET /system/groupobjectcount?groupId=xxx [&limit=xxx] [&offset=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"system Query Parameters"** (page 854).

There is no request payload.

### 12.10.7.0.1. Example Using cURL

The [example](#) below retrieves the stored object counts for each of the users in the "Pubs" group.

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/system/groupobjectcount?groupId=Pubs' | python -mjson.tool
```

The response payload is a JSON-formatted *UserUsage* object, which in this example is as follows. The group has three users in it. If a user has multiple buckets, the user's object count value is the sum total across all of the user's buckets. For example the user "brady" has a total of 5 stored objects. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"UserUsage Object"** (page 860).

```
[
  {
    "canonicalUserId": "da870acdd136d60789fb5c761fef4a4a",
    "groupId": "Pubs",
    "usageVal": 5,
    "userId": "brady"
  },
  {
    "canonicalUserId": "9bdcdd44ce1f9266adb9f22a8313feb4",
    "groupId": "Pubs",
    "usageVal": 5,
    "userId": "gilmore"
  },
  {
    "canonicalUserId": "9a00529cdfb6496a09c5105913b486ac",
    "groupId": "Pubs",
    "usageVal": 2,
    "userId": "gronk"
  }
]
```

#### 12.10.7.0.2. Response Format

The response payload is a JSON-formatted *UserUsage* object (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or this method-specific status code:

Status Code	Description
400	Missing required parameter: {groupId}
400	Limit should be greater than zero.

### 12.10.8. GET /system/license

#### GET /system/license Get HyperStore license terms

The request line syntax for this method is as follows.

```
GET /system/license
```

There is no request payload.

**Note** The HyperStore IAM Service supports a role-based access version of this API method as described in **"RBAC Version of this Method"** (page 849).

**Note** For background information about HyperStore licensing, see **"Licensing and Auditing"** (page 15).

### 12.10.8.0.1. Example Using cURL

The [example](#) below retrieves license data for the HyperStore system in which the call is submitted.

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/system/license | python -mjson.tool
```

The response payload is a JSON-formatted *LicenseData* object, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"LicenseData Object"** (page 857).

```
{
  "appliances": null,
  "enforcing": true,
  "expiration": 1621915200000,
  "gracePeriod": 0,
  "hyperIQ": "90",
  "issued": 1586751232344,
  "maxNetStorage": "100T",
  "maxRawStorage": null,
  "maxTieredStorage": "-1"
  "objectLockMode": "DISABLED",
  "storageExceeded": false,
  "storageMode": "NET",
  "tieringExceeded": false,
  "warnPeriod": 30
}
```

### 12.10.8.0.2. Response Format

The response payload is a JSON-formatted *LicenseData* object (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746).

### 12.10.8.1. RBAC Version of this Method

HyperStore's IAM Service supports an RBAC version of this API method. For an overview of the RBAC feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- Action name: *GetCloudianSystemLicense*
- Parameters: Same as for *GET /system/license* (no parameters)
- Response body: Same response data as for *GET /system/license* except the data is formatted in XML rather than JSON
- Role-based restrictions:
  - HyperStore system admin user can use this method
  - HyperStore group admin user cannot use this method
  - HyperStore regular user cannot use this method
  - IAM user can only use this method if granted *admin:GetCloudianSystemLicense* permission by policy, and subject to the same restriction as the parent HyperStore user
- Sample request and response (abridged):

```
REQUEST
```

```
http://localhost:16080/?Action=GetCloudianSystemLicense

<request headers including authorization info>

RESPONSE

200 OK

<?xml version="1.0" encoding="utf-8"?>
<GetCloudianSystemLicenseResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">
  <ResponseMetadata>
    <RequestId>system-generated-request-id</RequestId>
  </ResponseMetadata>
  <LicenseData>
    <expiration>2020-05-21T13:54:51.952-07:00</expiration>
    etc...
    ...
    ...
  </LicenseData>
</GetCloudianSystemLicenseResponse>
```

### 12.10.9. GET system/objectcount

**GET system/objectcount** Get stored object count for the system, a group, or a user

The request line syntax for this method is as follows.

```
GET /system/objectcount?groupId=xxx&userId=xxx
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"system Query Parameters"** (page 854).

There is no request payload.

**Note** If you want to retrieve the current object count for a particular **bucket**, use the [POST /usage/repair/bucket](#) method.

#### 12.10.9.0.1. Examples Using cURL

The [example](#) below retrieves the object count for the system as a whole (all users in all groups).

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/system/objectcount?groupId=ALL&userId=*
```

The response payload is the object count in plain text, which in this example is as follows:

```
1023
```

This next example retrieves the object count for the "Pubs" group as a whole (all users in the Pubs group).

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/system/objectcount?groupId=Pubs&userId=*
```

The response payload is:

215

This next example retrieves the object count for the user "PubsUser1".

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/system/objectcount?groupId=Pubs&userId=PubsUser1'
```

The response payload is:

54

### 12.10.9.0.2. Response Format

The response payload is plain text (see examples above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or this method-specific status code:

Status Code	Description
400	Missing required parameters: {groupId}, {userId}

## 12.10.10. GET /system/objectlockenabled

### GET /system/objectlockenabled    Get Object Lock feature status

The request line syntax for this method is as follows.

```
GET /system/objectlockenabled
```

There is no request payload.

The response will be:

- *true* if the Object Lock feature is enabled in the system -- which means that **all** of the following criteria for enabling Object Lock have been met:
  - Your HyperStore license supports the [Object Lock feature](#)
  - You have [enabled the HyperStore Shell](#)
  - You have [disabled root password access to HyperStore nodes](#)
- *false* if the Object Lock feature is not enabled in the system -- which means that one or more of the three criteria above have not been met

### 12.10.10.0.1. Example Using cURL

The [example](#) below retrieves the Object Lock feature status.

```
curl -X GET -k -u sysadmin:public https://localhost:19443/system/objectlockenabled
```

The response payload is the Object Lock feature status in plain text, which in this example is as follows.

```
false
```

## 12.10.11. GET /system/version

### GET /system/version    Get HyperStore system version

The request line syntax for this method is as follows.

```
GET /system/version
```

There is no request payload.

**Note** The HyperStore IAM Service supports a role-based access version of this API method as described in **"RBAC Version of this Method"** (page 852).

#### 12.10.11.0.1. Example Using cURL

The [example](#) below retrieves the HyperStore system version.

```
curl -X GET -k -u sysadmin:public https://localhost:19443/system/version
```

The response payload is the system version information in plain text, which in this example is as follows.

```
7.2.3 Compiled: 2020-09-04 12:24
```

#### 12.10.11.0.2. Response Format

The response payload is plain text (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746).

#### 12.10.11.1. RBAC Version of this Method

HyperStore's IAM Service supports an RBAC version of this API method. For an overview of the RBAC feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- Action name: *GetCloudianSystemVersion*
- Parameters: Same as for *GET /system/version* (no parameters)
- Response body: Same response data as for *GET /system/version* except the data is formatted in XML rather than JSON
- Role-based restrictions:
  - HyperStore system admin user, group admin user, and regular user can all use this method
  - IAM user can only use this method if granted *admin:GetCloudianSystemVersion* permission by policy
- Sample request and response:

REQUEST

```
http://localhost:16080/?Action=GetCloudianSystemVersion
```

<request headers including authorization info>

RESPONSE

```
200 OK
```

```
<?xml version="1.0" encoding="utf-8"?>
<GetCloudianSystemVersionResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">
  <ResponseMetadata>
    <RequestId>system-generated-request-id</RequestId>
  </ResponseMetadata>
  <String>
    7.1 Compiled: 2018-08-16 16:32
  </String>
</GetSystemVersionResponse>
```

## 12.10.12. POST /system/processProtectionPolicy

### POST /system/processProtectionPolicy    Process pending storage policy deletion or creation jobs

The request line syntax for this method is as follows.

```
POST /system/processProtectionPolicy
```

There is no request payload.

This method processes any pending storage policy deletion 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. This operator action marks the policy with a "DELETED" flag and makes it immediately unavailable to service users.

However, the full process of deleting the storage policy from the system is not completed until the *POST /system/processProtectionPolicy* method is run.

This method 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.

**Note** This method is invoked once a day by a HyperStore "Storage Policy Deletion or Creation Processing" (page 476) cron job.

#### 12.10.12.0.1. Example Using cURL

The [example](#) below triggers the processing of any pending storage policy deletion or creation jobs.

```
curl -X POST -k -u sysadmin:public \
https://localhost:19443/system/processProtectionPolicy
```

#### 12.10.12.0.2. Response Format

There is no response payload. For response status code this method will return one of the "Common Response Status Codes" (page 746).

### 12.10.13. POST /system/repairusercount

#### POST /system/repairusercount    Reconcile user counts in Redis and Cassandra

The request line syntax for this method is as follows.

```
POST /system/repairusercount
```

Use this method if you have reason to suspect that user counts in your audit data are inaccurate. This method will synchronize the user counts in Redis (which are used in audit data) to the metadata in the Cassandra User-Info table.

There is no request payload.

##### 12.10.13.0.1. Example Using cURL

The [example](#) below syncs the user counts in Redis with the Cassandra metadata.

```
curl -X POST -k -u sysadmin:public https://localhost:19443/system/repairusercount
```

##### 12.10.13.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746).

### 12.10.14. system Query Parameters

*groupId, userId*

(Mandatory, string) For the *GET /system/bytecount*, *GET /system/bytestiered*, and *GET /system/objectcount* methods: Use the *groupId* and *userId* parameters to specify whether you want to retrieve a count for the whole system, for one whole group, or for one particular user:

- Whole system: *groupId=ALL&userId=\**
- One whole group: *groupId=<groupId>&userId=\** (example: *groupId=Dev&userId=\**)
- One particular user : *groupId=<groupId>&userId=<userId>* (example: *groupId=Dev&userId=Cody*)

*groupId*

(Mandatory, string) For the *GET /system/groupbytecount* and *GET /system/groupobjectcount* methods: Use the *groupId* parameter to specify the group for which you want to retrieve counts for all users in the group.

*limit*

(Optional, integer) For the *GET /system/groupbytecount* and *GET /system/groupobjectcount* methods: For purposes of pagination, the optional *limit* parameter specifies the maximum number of users to return in one response. In the response the users are sorted alphanumerically and if there are more than "limit" users in the group, then the number of users returned will be "limit plus 1" (for example, 101 users if the limit is 100). The last, extra returned user — the "plus 1" — is an indicator that there are more users than could be returned in the current response (given the specified "limit" value). That last user's ID can then be used as the "offset" value in a subsequent request that retrieves additional users.

**Note** If the offset user happens to be the last user in the entire set of users, the subsequent query using the offset will return no users.

Defaults to 100.

#### *offset*

(Optional, string) For the *GET /system/groupbytecount* and *GET /system/groupobjectcount* methods: The user ID with which to start the response list of users for the current request, sorted alpha-numerically. The optional "offset" parameter can be used for purposes of pagination within a large result set that is being retrieved via multiple sequential requests. See the description of "limit" above for more information.

If "offset" is not specified, the first user in the response list will be the alphanumerically first user from the entire set of users in the group.

#### *region*

(Optional, string) For the *GET /system/audit* method: The service region for which to retrieve audit data. If the region is not specified in the request, then the returned audit data will be for the whole system (all regions), combined.

## 12.10.15. system Objects

This section describes the JSON-formatted data objects that constitute the request or response payloads for the System related Admin API methods.

**Note** For examples of these objects see the API method request and response examples.

- **"AuditData Object"** (page 855)
- **"LicenseData Object"** (page 857)
- **"UserUsage Object"** (page 860)

### 12.10.15.1. AuditData Object

The *AuditData* object consists of the following attributes.

#### *byteCount*

(number) Net bytes of object data stored in the system. This measure excludes overhead from replication and erasure coding.

Example:

```
"byteCount": 647687490
```

#### *bytesInCount*

(number) This measure is not implemented currently and its value will always be "0".

Example:

```
"bytesInCount": 0
```

*bytesOutCount*

(number) This measure is not implemented currently and its value will always be "0".

Example:

```
"bytesOutCount": 0
```

*licenseExpiration*

(number) License expiration date-time, in UTC milliseconds.

Example:

```
"licenseExpiration": 1590094491952
```

*nodes*

(set) The list of nodes that comprise the HyperStore system, identified by IP address.

Example:

```
"nodes": [
  {
    "name": "10.50.50.201"
  },
  {
    "name": "10.50.50.202"
  },
  {
    "name": "10.50.50.203"
  }
],
```

*objectCount*

(number) Number of objects currently stored in the system.

Example:

```
"objectCount": 13
```

*os*

(string) Operating system version being used by HyperStore hosts.

Example:

```
"os": "Linux 3.10.0-957.1.3.el7.x86_64 amd64"
```

*tieredBytesCount*

(number) Number of bytes of object data that has been auto-tiered to a remote destination or destinations.

Example:

```
"tieredBytesCount": 0
```

*timestamp*

(number) Date-time when audit data was automatically updated at the top of the most recently

completed hour. In UTC milliseconds. Note that this timestamp is not affected by calling the *POST /system/audit* method (this method updates the counts, but not the timestamp).

Example:

```
"timestamp": 1563724800000
```

#### *userCount*

(number) Number of active users in the system. This includes administrators as well as regular users.

Example:

```
"userCount": 3
```

### 12.10.15.2. LicenseData Object

The *LicenseData* object consists of the following attributes.

#### *appliances*

(JSON object) Information about each HyperStore appliance in the cluster, if any. Information for each appliance consists of:

- *nodeId*
- *maxStorage* -- This is the amount of raw storage capacity usage licensed for this individual appliance machine.
- *productName*

If there are no HyperStore appliances in the cluster, the *appliances* attribute is set to 0.

Example:

```
"appliances": null
```

#### *enforcing*

(Boolean) If *true*, then the system will enforce the licensed storage maximum by rejecting S3 PUTs and POSTs if the cluster stored volume reaches 110% of the licensed maximum. If this happens, then support for S3 PUTs and POSTs will resume again after the cluster stored volume is less than 100% of the licensed maximum (either because data has been deleted, or because a new license with higher maximum cluster stored volume has been installed).

Also, if this attribute is set to *true*, then the system will enforce the licensed tiering maximum by no longer auto-tiering data if the tiered volume reaches 110% of the licensed tiering maximum. If this happens, then support for auto-tiering will resume again after the tiered volume is less than 100% of the licensed tiering maximum (either because tiered data has been deleted through HyperStore interfaces, or because a new license with higher maximum tiered volume has been installed).

For more information on license enforcement see **"Licensing and Auditing"** (page 15).

Example:

```
"enforcing": true
```

#### *expiration*

(Number) License expiration date-time in UTC milliseconds. Example:

```
"expiration": 1621915200000
```

*gracePeriod*

(Number) After the license expiration date passes, the number of days until the HyperStore system is automatically disabled. Example:

```
"gracePeriod": 0
```

*hyperIQ*

(String) Your HyperStore license's level of support for Cloudian HyperIQ. Cloudian HyperIQ is a solution for dynamic visualization and analysis of HyperStore monitoring 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* attribute in the LicenseData object indicates what level of HyperIQ functionality will be available to you **if you acquire and set up HyperIQ**.

- basic -- HyperIQ dashboards for OS and service status monitoring are supported indefinitely.
- <number of days> -- HyperIQ dashboards for OS and service status monitoring are supported indefinitely, and also an S3 analytics dashboard is supported for <number of days> duration from the HyperStore license issuance. This HyperStore license type has "Enterprise HyperIQ" support, with the distinction (as versus only the "basic" support level) being the availability of the S3 analytics dashboard in HyperIQ.

Example

```
"hyperIQ": "90"
```

*issued*

(Number) License issuance date-time in UTC milliseconds. Example:

```
"issued": 1586751232344
```

*maxNetStorage*

(String) Applicable only if "storageMode" is "NET". If "storageMode" is "RAW" then this attribute will be null.

Maximum allowed Net storage volume for your entire HyperStore system. This is expressed as "<n>G" (for number of GiBs) or "<n>T" (for number of TiBs), or so on. For example, "100T" for one hundred TiBs. This value does not use decimals and will be expressed in GiBs unless it's exact number of TiBs (that is, an exact multiple of 1024 GiBs). For example, 1024 GiBs would be expressed as "1T" but 1030 GiBs would be expressed as "1030G".

"Net" storage volume usage **excludes overhead from replication or erasure coding**. For example a 1GiB object protected by 3X replication counts as 1GiB toward the "maxNetStorage" limit — not as 3GiB.

Example:

```
"maxNetStorage": "100T"
```

*maxRawStorage*

(String) Applicable only if "storageMode" is "RAW". If "storageMode" is "NET" then this attribute will be null.

If "storageMode" is "RAW", the "maxRawStorage" attribute indicates any additional raw storage licensed for your cluster above and beyond the raw storage licensed to each of your appliance nodes (as

indicated by the "maxStorage" child attributes within the "appliances" attribute). Typically the "maxRawStorage" attribute would have a non-zero value only if your cluster has a mix of appliance nodes and software-only nodes. In such an environment, total licensed raw storage for the cluster is the sum of each of the individual appliance machine raw storage licenses **plus** the "maxRawStorage".

In a cluster consisting purely of appliance nodes, the "maxRawStorage" value would typically be 0. In such an environment, total licensed raw storage for the cluster is the sum of each of the individual appliance machine raw storage licenses.

When non-zero, "maxRawStorage" is expressed as "<n>G" (for number of GiBs) or "<n>T" (for number of TiBs), or so on. For example, "100T" for one hundred TiBs. This value does not use decimals and will be expressed in GiBs unless it's exact number of TiBs (that is, an exact multiple of 1024 GiBs). For example, 1024 GiBs would be expressed as "1T" but 1030 GiBs would be expressed as "1030G".

Raw storage volume usage counts **all stored data**, including overhead from replication or erasure coding. For example a 1GB object protected by 3X replication counts as 3GB toward a raw storage license limit. Also, stored metadata counts toward the limit as well.

Example:

```
"maxRawStorage": null
```

#### *maxTieredStorage*

(String) Maximum allowed volume of auto-tiered data stored in external systems other than HyperStore, after having been transitioned to those systems from HyperStore. This is expressed as "<n>G" (for number of GiBs) or "<n>T" (for number of TiBs), or so on. For example, "100T" for one hundred TiBs. This value does not use decimals and will be expressed in GiBs unless it's exact number of TiBs (that is, an exact multiple of 1024 GiBs). For example, 1024 GiBs would be expressed as "1T" but 1030 GiBs would be expressed as "1030G".

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.

This attribute may have the value "-1" to indicate "unlimited" (i.e. the license places no limit on tiered data volume).

Example:

```
"maxTieredStorage": "-1"
```

#### *objectLockMode*

(String) The license's support or non-support of the HyperStore WORM (Object Lock) feature:

- DISABLED -- Object Lock is not supported.
- ENABLED -- Object Lock is supported.

**Note** To use the Object Lock feature, along with having a license that supports this feature you must enable the HyperStore Shell (HSH) and disable the root password on your HyperStore nodes. For more information see **"Enabling the HSH and Managing HSH Users"** (page 90) and **"WORM (Object Lock)"** (page 121).

Example:

```
"objectLockMode": "DISABLED"
```

*storageExceeded*

(Boolean) This flag sets to *true* if the cluster storage volume reaches 110% of licensed maximum storage. It sets back to *false* when the cluster storage volume is less than 100% of licensed maximum storage (either because data has been deleted, or because a new license with higher maximum storage volume has been installed).

Example:

```
"storageExceeded": false
```

*storageMode*

(String) The type of storage volume licensing applied by this license: either "NET" or "RAW". See "maxNetStorage" and "maxRawStorage" for more detail. Example:

```
"storageMode": "NET"
```

*tieringExceeded*

(Boolean) This flag sets to *true* if the tiered storage volume reaches 110% of the licensed maximum tiered volume. It sets back to *false* when the tiered storage volume is less than 100% of the licensed maximum tiered volume (either because tiered data has been deleted through HyperStore interfaces, or because a new license with higher maximum tiered volume has been installed).

Example:

```
"tieringExceeded": false
```

*warnPeriod*

(Number) Starting this many days before the license expiration date, an expiration warning message will display at the top of the Clouddian Management Console. Example:

```
"warnPeriod": 30
```

### 12.10.15.3. UserUsage Object

The *UserUsage* object consists of the following attributes.

*canonicalUserId*

(Number) The user's system-generated canonical user ID. Example:

```
"canonicalUserId": "da870acdd136d60789fb5c761fef4a4a"
```

*groupId*

(Number) The ID of the group to which the user belongs. Example:

```
"groupId": "Pubs"
```

*usageVal*

(Number) Either the user's current stored byte count (in response to a *GET /system/groupbytecount* request) or the user's current stored object count (in response to a *GET /system/groupobjectcount* request). If the user has multiple buckets, the count is a combined total across all of the user's buckets.

Example:

```
"usageVal": 220508
```

*userId*

(String) The user's user ID. Example:

```
"userId": "brady"
```

## 12.11. tiering

The Admin API methods built around the **tiering** resource are for managing account credentials to use for accessing auto-tiering destination systems. You can post tiering credentials to associate with specific HyperStore source buckets, and the system will securely store the credentials and use them when implementing auto-tiering for those buckets. For S3-compliant tiering destinations you also have the option of posting a system default tiering credential, which can be made available for all bucket owners to use for auto-tiering to a system default tiering destination.

**Note** Having a system default tiering credential is only supported for S3-compliant tiering destinations -- not for Azure or Spectra.

### 12.11.1. DELETE /tiering/credentials

**DELETE /tiering/credentials** Delete a tiering credential for Amazon, Google, or other S3-compliant destination

The request line syntax for this method is as follows.

```
DELETE /tiering/credentials[?bucketName=xxx]
```

For parameter description click on the parameter name or see "**tiering Query Parameters**" (page 868).

There is no request payload.

#### 12.11.1.0.1. Example Using cURL

The [example](#) below deletes the S3 auto-tiering credential currently associated with a HyperStore source bucket named "bucket1".

```
curl -X DELETE -k -u sysadmin:public \
https://localhost:19443/tiering/credentials?bucketName=bucket1
```

#### 12.11.1.0.2. Response Format

There is no response payload. For response status code this method will return one of the "**Common Response Status Codes**" (page 746).

### 12.11.2. DELETE /tiering/azure/credentials

**DELETE /tiering/azure/credentials** Delete a tiering credential for Azure

The request line syntax for this method is as follows.

```
DELETE /tiering/azure/credentials?bucketName=xxx
```

For parameter description click on the parameter name or see **"tiering Query Parameters"** (page 868).

There is no request payload.

#### 12.11.2.0.1. Example Using cURL

The [example](#) below deletes the Azure auto-tiering credential currently associated with a HyperStore source bucket named "bucket2".

```
curl -X DELETE -k -u sysadmin:public \
https://localhost:19443/tiering/azure/credentials?bucketName=bucket2
```

#### 12.11.2.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746).

### 12.11.3. DELETE /tiering/spectra/credentials

**DELETE /tiering/spectra/credentials** Delete a tiering credential for Spectra

The request line syntax for this method is as follows.

```
DELETE /tiering/spectra/credentials?bucketName=xxx
```

For parameter description click on the parameter name or see **"tiering Query Parameters"** (page 868).

There is no request payload.

#### 12.11.3.0.1. Example Using cURL

The [example](#) below deletes the Spectra auto-tiering credential currently associated with a HyperStore source bucket named "bucket1".

```
curl -X DELETE -k -u sysadmin:public \
https://localhost:19443/tiering/spectra/credentials?bucketName=bucket1
```

#### 12.11.3.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746).

### 12.11.4. GET /tiering/credentials

**GET /tiering/credentials** Get a tiering credential for Amazon, Google, or other S3-compliant destination

The request line syntax for this method is as follows.

```
GET /tiering/credentials[?bucketName=xxx]
```

For parameter description click on the parameter name or see **"tiering Query Parameters"** (page 868).

There is no request payload.

### 12.11.4.0.1. Example Using cURL

The [example](#) below retrieves the S3 auto-tiering credential currently associated with a HyperStore source bucket named "bucket1".

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/tiering/credentials?bucketName=bucket1
```

The response payload is the S3 access key in plain text, which in this example is as follows. The secret key is not returned.

```
00cc33c4blef9f50282a
```

### 12.11.4.0.2. Response Format

The response payload is a plain text string (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
204	No Tiering Credentials found.

## 12.11.5. GET /tiering/credentials/src

**GET /tiering/credentials/src** Check whether a bucket uses a bucket-specific or system default tiering credential

The request line syntax for this method is as follows.

```
GET /tiering/credentials/src[?bucketName=xxx]
```

For parameter description click on the parameter name or see **"tiering Query Parameters"** (page 868).

For buckets that auto-tier to Amazon, Google, or other S3-compliant destinations, you can use this method to check whether the bucket is using a bucket-specific tiering credential or the system default tiering credential (or no credential, if the bucket has not yet been configured for auto-tiering). You can omit the "bucketName" parameter if you want to check whether or not the system default tiering credential has been created for the system. The method responds with a plain text string -- either "BUCKET" (bucket-specific credential), "SYSTEM" (system default credential), or NONE (no credential has been set).

There is no request payload.

**Note** This method is not supported for buckets that tier to Azure or Specta.

### 12.11.5.0.1. Example Using cURL

The [example](#) below checks the S3 auto-tiering credential type for a HyperStore source bucket named "bucket1".

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/tiering/credentials/src?bucketName=bucket1
```

In this example the response payload is BUCKET, indicating that "bucket1" uses a bucket-specific tiering credential in its S3 auto-tiering configuration.

BUCKET

### 12.11.5.0.2. Response Format

The response payload is plain text string (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746).

## 12.11.6. GET /tiering/azure/credentials

### GET /tiering/azure/credentials    Get a tiering credential for Azure

The request line syntax for this method is as follows.

```
GET /tiering/azure/credentials?bucketName=xxx
```

For parameter description click on the parameter name or see **"tiering Query Parameters"** (page 868).

There is no request payload.

#### 12.11.6.0.1. Example Using cURL

The [example](#) below retrieves the Azure auto-tiering credential currently associated with a HyperStore source bucket named "bucket2".

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/tiering/azure/credentials?bucketName=bucket1
```

The response payload is the Azure account name and account key in plain text with comma-separation, which in this example is as follows.

```
123456,Oy1wMUklsF81331LIGY5RlVqa8Rg+iWT6zEFt6I1
```

#### 12.11.6.0.2. Response Format

The response payload is a plain text string (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
204	No Tiering Credentials found.

## 12.11.7. GET /tiering/spectra/credentials

### GET /tiering/spectra/credentials    Get a tiering credential for Spectra

The request line syntax for this method is as follows.

```
GET /tiering/spectra/credentials?bucketName=xxx
```

For parameter description click on the parameter name or see **"tiering Query Parameters"** (page 868).

There is no request payload.

### 12.11.7.0.1. Example Using cURL

The [example](#) below retrieves the Spectra auto-tiering credential currently associated with a HyperStore source bucket named "bucket1".

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/tiering/spectra/credentials?bucketName=bucket1
```

The response payload is the access key in plain text, which in this example is as follows. The secret key is not returned.

```
00d5dc27224f9d529257
```

### 12.11.7.0.2. Response Format

The response payload is a plain text string (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
204	No Tiering Credentials found.

## 12.11.8. POST /tiering/credentials

**POST /tiering/credentials** Post a tiering credential for Amazon, Google, or other S3-compliant destination

The request line syntax for this method is as follows.

```
POST /tiering/credentials?accessKey=urlencode (xxx) &secretKey=urlencode (xxx)
[&bucketName=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"tiering Query Parameters"** (page 868).

There is no request payload.

### 12.11.8.0.1. Example Using cURL

The [example](#) below posts S3 auto-tiering credentials for a HyperStore source bucket named "bucket1".

```
curl -X POST -k -u sysadmin:public \
'https://
localhost:19443/tiering/credentials?accessKey=00cc33c4b1e&secretKey=YuaOJ7l0Fqc&bucketName=bucket1'
```

When implementing auto-tiering from this source bucket to an S3-compatible destination system (as configured by the bucket lifecycle configuration), the HyperStore system will use this credential.

**Note** In the example above, the access key and secret key are truncated so that the 'https://...' segment can be shown on one line.

**Note** If an access key or secret key includes a non-alphanumeric character and you do not URL-encode the key, the API server will return a 403 error.

### 12.11.8.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
204	Bucket does not exist.
400	Missing required attributes : {accessKey, secretKey}

## 12.11.9. POST /tiering/azure/credentials

### POST /tiering/azure/credentials Post a tiering credential for Azure

The request line syntax for this method is as follows.

```
POST /tiering/azure/credentials?<a href="#">accountName=urlencode (xxx)
&<a href="#">accountKey=urlencode (xxx) &<a href="#">bucketName=xxx
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"tiering Query Parameters"** (page 868).

There is no request payload.

#### 12.11.9.0.1. Example Using cURL

The [example](#) below posts Azure auto-tiering credentials for a HyperStore source bucket named "bucket2".

```
curl -X POST -k -u sysadmin:public \
'https://localhost:19443/tiering/azure/credentials?accountName=
123&accountKey=Oy1wMU&bucketName=bucket2'
```

When implementing auto-tiering from this source bucket to an Azure destination system (as configured by the bucket lifecycle configuration), the HyperStore system will use this credential.

**Note** In the example above, the account name and key are truncated so that the 'https://...' segment can be shown on one line.

**Note** If an account name or account key includes a non-alphanumeric character and you do not URL-encode the key, the API server will return a 403 error.

#### 12.11.9.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
204	Bucket does not exist.
400	Missing required attributes : {accountName, accountKey}

### 12.11.10. POST /tiering/spectra/credentials

#### POST /tiering/spectra/credentials Post a tiering credential for Spectra

The request line syntax for this method is as follows.

```
POST /tiering/spectra/credentials?accessKey=urlencode (xxx)
&secretKey=urlencode (xxx) &bucketName=xxx
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"tiering Query Parameters"** (page 868).

There is no request payload.

#### 12.11.10.0.1. Example Using cURL

The [example](#) below posts Spectra auto-tiering credentials for a HyperStore source bucket named "bucket1".

```
curl -X POST -k -u sysadmin:public \
'https://
localhost:19443/tiering/spectra/credentials?accessKey=00d5d&secretKey=PxvAH6Ks&bucketName=bucket1'
```

When implementing auto-tiering from this source bucket to a Spectra destination system (as configured by the bucket lifecycle configuration), the HyperStore system will use this credential.

**Note** In the example above, the access key and secret key are truncated so that the 'https://...' segment can be shown on one line.

**Note** If an access key or secret key includes a non-alphanumeric character and you do not URL-encode the key, the API server will return a 403 error.

#### 12.11.10.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
204	Bucket does not exist.
400	Missing required attributes : {accessKey, secretKey}

### 12.11.11. tiering Query Parameters

*bucketName*

(Optional for S3-compliant tiering, mandatory for Azure or Spectra tiering. String) Name of the **Hyper-Store source bucket** that uses the credential for auto-tiering to a destination system.

For tiering to Amazon, Google, or other S3-compliant destinations, if the *bucketName* parameter is omitted then the request applies to the system default auto-tiering credential. For example, with the POST method for S3 tiering credentials, if you omit the bucket name then you are POSTing a system default credential for tiering to an S3 destination. With GET or DELETE methods if you omit the bucket name then you are retrieving or deleting the system default S3 tiering credential.

**Note** Having a system default tiering credential is only supported for S3-compliant tiering destinations -- not for Azure or Spectra.

*accessKey*

(Mandatory, string) Access key for the tiering destination account.

*secretKey*

(Mandatory, string) Secret key for the tiering destination account.

*accountName*

(Mandatory, string) Name of the Azure tiering destination account.

*accountKey*

(Mandatory, string) Account key for the Azure tiering destination account.

## 12.12. usage

The Admin API methods built around the **usage** resource are for managing HyperStore usage reporting. This includes support for retrieving service usage data for specified users, groups, or buckets. There are also methods for aggregating usage data and ensuring its accuracy — many of these methods are invoked regularly by HyperStore system cron jobs.

For an overview of the HyperStore usage reporting feature, see "**Usage Reporting and Billing Feature Overview**" (page 138).

### 12.12.1. DELETE /usage

**DELETE /usage** Delete usage data

The request line syntax for this method is as follows.

```
DELETE /usage?granularity=xxx&startTime=xxx [&unitCount=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see "**usage Query Parameters**" (page 880).

There is no request payload.

This method deletes service usage data from the Reports keyspace in Cassandra. Separate data exists for the raw, hourly roll-up, daily roll-up, and monthly roll-up levels. Note that when you delete usage data, usage data for **all** groups and users will be deleted for your specified granularity and time period.

Apart from using this API method, usage data deletion is also managed by configurable retention periods after which the system automatically deletes the data. See **"Setting Usage Data Retention Periods"** (page 144).

**IMPORTANT !** The HyperStore system calculates monthly bills for service users by aggregating hourly roll-up data. Once hourly data is deleted, you will not be able to generate bills for the service period covered by that data.

**Note** If you have [enabled the per-bucket usage data feature](#), this API method does not delete per-bucket usage data. It deletes only per-group and per-user usage data. Deletion of per-bucket usage data is managed exclusively by the configuration retention periods.

#### 12.12.1.0.1. Example Using cURL

The [example](#) below deletes daily roll-up usage data from the day of May 1st, 2017.

```
curl -X DELETE -k -u sysadmin:public \
'https://localhost:19443/usage?granularity=day&startTime=20170501&unitCount=1'
```

#### 12.12.1.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing or invalid parameters

### 12.12.2. GET /usage

**GET /usage** Get usage data for group, user, or bucket

The request line syntax for this method is as follows.

```
GET /usage? [id=xxx | canonicalUserId=xxx | bucket=xxx] &operation=xxx &startTime=xxx
&endTime=xxx &granularity=xxx &reversed=xxx [ &limit=xxx ] [ &pageSize=xxx ]
[ &offset=xxx ] [ &region=xxx ] [ &regionOffset=xxx ]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"usage Query Parameters"** (page 880).

There is no request payload.

**Note** The HyperStore IAM Service supports a role-based access version of this API method as described in **"RBAC Version of this Method"** (page 872).

**Note** The `GET /usage?bucket=xxx...` option is supported only if bucket usage statistics are enabled in the system. Bucket usage statistics are disabled by default. For information on enabling this feature see **"Enabling Advanced Usage Reporting Features"** (page 142).

### 12.12.2.0.1. Examples Using cURL

The first [example](#) below retrieves the monthly stored bytes usage data for the "QA" group, from July 2017.

```
curl -X GET -k -u sysadmin:public \
'https://
localhost
:19443/usage?id=
QA|*&operation=SB&startTime=201707010000&endTime=201708010000&granularity=month' \
| python -mjson.tool
```

The response payload is a JSON-formatted list of *UsageData* objects, which in this example is as follows. Note that in this case we are retrieving monthly roll-up data from a time interval that spans just one month, so here there is just one *UsageData* object in the list. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"UsageData Object"** (page 886).

```
[
{
  "averageValue": "107956",
  "bucket": null,
  "count": "744",
  "groupId": "QA",
  "ip": "",
  "maxValue": "305443",
  "operation": "SB",
  "region": "taoyuan",
  "timestamp": "1498867200000",
  "uri": "",
  "userId": "*",
  "value": "80319535",
  "whitelistAverageValue": "0",
  "whitelistCount": "0",
  "whitelistMaxValue": "0",
  "whitelistValue": "0"
}
```

The next example below retrieves the total bytes count for a bucket named "bucket1" as of the specified hour interval. Note that to support retrieving the total bytes (TB) count or total objects (TO) count for a bucket as of a specified time interval, the [POST /usage/repair/bucket](#) method must have been executed for that bucket some-time during that time interval (since that method generates the TB and TO counts). If that method has not been executed for a bucket during a given time interval -- such as a particular hour or day -- then you cannot subsequently retrieve a TB or TO count for that bucket from that interval.

```
curl -X GET -k -u sysadmin:public \
,
https://
localhost
```

```
:19443/usage?bucket=
bucket1&operation=TB&startTime=201712201400&endTime=201712201500&granularity=raw' \
| python -mjson.tool
```

The response payload is a JSON-formatted list of *UsageData* objects, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"UsageData Object"** (page 886).

```
[
  {
    "averageValue": "4242572",
    "bucket": "bucket1",
    "count": "0",
    "groupId": null,
    "ip": null,
    "maxValue": "0",
    "operation": "TB",
    "policyId": "880e7d065225009b481ff24ae8d893ce",
    "region": null,
    "timestamp": "1513781460000",
    "uri": null,
    "userId": null,
    "value": "4242572",
    "whitelistAverageValue": "0",
    "whitelistCount": "0",
    "whitelistMaxValue": "0",
    "whitelistValue": "0"
  }
]
```

**Note** If the *POST /usage/repair/bucket* method had been called multiple times during the time period specified in the *GET /usage?bucket* request, and the requested granularity is "raw", then multiple *UsageData* objects would be returned in the response, each with a TB value and each with a timestamp indicating when the *POST /usage/repair/bucket* call had generated that TB value. By contrast, if the requested granularity is a roll-up period such as "hour", then only most recent TB value generated during that roll-up period would be returned.

For example, suppose that you have been executing the *POST /usage/repair/bucket* call on a particular bucket at 11AM and 11PM every day. Subsequently, if the start and end times in a *GET /usage?bucket* request span one week and the requested granularity is "day", the response will return one *UsageData* object for each day of the week, and the TB count shown for each day will be the one generated by the *POST /usage/repair/bucket* call executed at 11PM on each day.

### 12.12.2.0.2. Response Format

The response payload is a JSON-formatted list of *UsageData* objects (see examples above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing or invalid parameters

Status Code	Description
400	Invalid parameter: region = {region}
400	Invalid parameter: regionOffset = {region}
400	Conflicting parameters: {canonicalUserId, id}

### 12.12.2.1. RBAC Version of this Method

HyperStore's IAM Service supports an RBAC version of this API method. For an overview of the RBAC feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- Action name: *GetCloudianUsage*
- Parameters: Same as for *GET /usage*, except all parameter names start with an upper case letter rather than lower case
- Response body: Same response data as for *GET /usage* except the data is formatted in XML rather than JSON
- Role-based restrictions:
  - HyperStore system admin user can get usage for any group, user, or bucket
  - HyperStore group admin user can only get usage for her own group, for users within her own group, or for buckets owned by users within her own group
  - HyperStore regular user can only get his own usage or usage for a bucket that he owns
  - IAM user can only use this method if granted *admin:GetCloudianUsage* permission by an IAM policy, and subject to the same restriction as the parent HyperStore user.

**Note** The "GetCloudianUsage" action retrieves usage data for Cloudian HyperStore user accounts, not for subsidiary IAM users. The system does not maintain usage data per IAM user. For example, if a HyperStore group administrator grants *admin:GetCloudianUsage* permission to an IAM user, the IAM user will be able to retrieve usage information for **any HyperStore user in the group administrator's group**. And if a HyperStore regular user grants *admin:GetCloudianUsage* permission to an IAM user, the IAM user will be able to retrieve usage information for the **parent HyperStore user**.

- Sample request and response (abridged):

```

REQUEST

http://localhost:16080/?Action=GetCloudianUsage&Id=QA|*&Operation=SB&StartTime=201807010000
&EndTime=201808010000&Granularity=month

<request headers including authorization info>

RESPONSE

200 OK

<?xml version="1.0" encoding="utf-8"?>
<GetCloudianUsageResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">
<ResponseMetadata>

```

```

<RequestId>system-generated-request-id</RequestId>
</ResponseMetadata>
<ListWrapper>
  <UsageData>
    etc...
    ...
    ...
  </UsageData>
  <UsageData>
    etc...
    ...
    ...
  </UsageData>
</ListWrapper>
</GetCloudianUsageResponse>

```

### 12.12.3. POST /usage/bucket

#### POST /usage/bucket Get raw usage data for multiple buckets

The request line syntax for this method is as follows.

```
POST /usage/bucket
```

The required request payload is a JSON-formatted *UsageBucketReq* object. See example below.

This method retrieves complete **raw** usage data for one or multiple specified buckets, from during a specified time period. This method does not support retrieving rolled up hourly, daily, or monthly usage data and it does not support filtering by the service operation type.

**Note** If you want to retrieve rolled up usage data for a bucket, or bucket usage data for just a particular service operation type, use the [GET /usage](#) method instead. Note however that with the [GET /usage](#) method you can only get usage data for one bucket at a time.

**Note** The *POST /usage/bucket* method is supported only if bucket usage statistics are enabled in the system. Bucket usage statistics are disabled by default. For information on enabling this feature see **"Enabling Advanced Usage Reporting Features"** (page 142).

#### 12.12.3.0.1. Example Using cURL

The [example](#) below retrieves raw usage data for two buckets named "b123" and "mybucket", for a one hour period. In this example the JSON-formatted *UsageBucketReq* object is specified in a text file named *buckets\_usage.txt* which is then referenced as the data input to the cURL command.

```
curl -X POST -H "Content-Type: application/json" -k -u sysadmin:public \
-d @buckets_usage.txt https://localhost:19443/usage/bucket | python -mjson.tool
```

The *buckets\_usage.txt* file content in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"UsageBucketReq Object"** (page 885).

```
{
  "buckets": [
    "b123",
    "mybucket"
  ],
  "endTime": "201611291900",
  "startTime": "201611291800"
}
```

The response payload is a JSON-formatted list of *UsageBucketRes* objects (with one such object for each bucket), which in this example is as follows. The response payload is truncated here. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"UsageBucketRes Object"** (page 885).

**Note** If during your specified start and end time interval there were no operations of a particular type in the bucket, then no data will be returned for that operation type. For example, if there were no deletes during the interval then no "HD" operation usage data will be returned.

```
[
  {
    "bucket": "b123",
    "data": [
      {
        "averageValue": "3222",
        "bucket": "b123",
        "count": "0",
        "groupId": null,
        "ip": "10.10.0.1",
        "maxValue": "0",
        "operation": "BO",
        "region": null,
        "timestamp": "1480442520000",
        "uri": null,
        "userId": null,
        "value": "3222",
        "whitelistAverageValue": "0",
        "whitelistCount": "0",
        "whitelistMaxValue": "0",
        "whitelistValue": "0"
      },
      ...
    ]
  },
  ...
]
```

### 12.12.3.0.2. Response Format

The response payload is a JSON-formatted list of *UsageBucketRes* objects (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required attributes : {buckets, startTime, endTime}
400	Invalid JSON Object

## 12.12.4. POST /usage/repair

### POST /usage/repair Repair storage usage data for group or system

The request line syntax for this method is as follows.

```
POST /usage/repair?groupId=xxx[&summarizeCountsOnly=xxx] [&region=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"usage Query Parameters"** (page 880).

There is no request payload.

This method checks and repairs storage usage data for specified user groups or for all groups in the system. For each repaired group the operation repairs the storage usage counts for individual users within the group as well as the aggregate counts for the group as a whole.

For background information on storage usage data repair, see **"Validating Storage Usage Data"** (page 143).

**Note** This is a resource-intensive operation if you have a large number of users in your system. Note that a more focused type of storage usage repair is run as a recurring HyperStore cron job -- see [POST /usage/repair/dirtyusers](#).

**Note** In a multi-region HyperStore system, this method can be applied to usage data in all regions by submitting the request to the Admin Service in the default region and omitting the "region" query parameter. You cannot directly run this method against Admin Service nodes in non-default regions.

#### 12.12.4.0.1. Example Using cURL

The [example](#) below checks and repairs storage usage data for the "engineering" group.

```
curl -X POST -k -u sysadmin:public \
https://localhost:19443/usage/repair?groupId=engineering
```

#### 12.12.4.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746).

## 12.12.5. POST /usage/repair/bucket

### POST /usage/repair/bucket Retrieve total bytes and total objects for a bucket

The request line syntax for this method is as follows.

```
POST /usage/repair/bucket?bucket=xxx
```

For parameter description click on the parameter name or see **"usage Query Parameters"** (page 880).

There is no request payload.

This method calculates and returns the current counts for total bytes stored and number of objects stored in a specified bucket. The calculation entails reading Cassandra metadata for objects in the bucket.

**Note** This is potentially a resource-intensive operation, depending on how many objects are in the bucket.

**Note** This API method is supported even if bucket usage statistics are disabled in the system. Bucket usage statistics are disabled in the system by default. For more information on bucket statistics see **"Bucket Usage Statistics"** (page 139).

#### 12.12.5.0.1. Example Using cURL

The [example](#) below calculates and returns the current total bytes stored (TB) and total objects stored (TO) for a bucket named "testbucket1".

```
curl -X POST -k -u sysadmin:public \
https://localhost:19443/usage/repair/bucket?bucket=testbucket1 | python -mjson.tool
```

The response payload is the JSON-formatted TB and TO values.

```
{
  "TB": 305360,
  "TO": 9
}
```

#### 12.12.5.0.2. Response Format

The response payload is the TB and TO values in JSON (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746).

### 12.12.6. POST /usage/repair/dirtyusers

**POST /usage/repair/dirtyusers**    Repair storage usage data for users with recent activity

The request line syntax for this method is as follows.

```
POST /usage/repair/dirtyusers[?summarizeCounts=xxx]
```

For parameter description click on the parameter name or see **"usage Query Parameters"** (page 880).

There is no request payload.

This method checks and repairs storage usage data for users whose storage bytes and/or storage object counts in the Redis QoS database have changed since the last time those users' counts were subjected to a usage repair. This method selects users at random from among this set of "dirty" users, and performs usage repair for a configurable maximum number of those users per method execution (*mts.properties.erb: usage.repair.maxdirtyusers*; default = 1000).

For background information on storage usage data repair, see **"Validating Storage Usage Data"** (page 143).

**Note** This method is invoked once every 12 hours by a HyperStore **"Usage Data Processing"** (page 474) cron job. In a multi-region system, a separate cron job is run from within each region.

**Note** At the conclusion of this method's run, in *cloudian-admin.log* there will be an INFO level message from the `CassandraUsageAccess::repairDirtyUsers` component that indicates "1000 users processed. N remaining", where N is the number of remaining dirty users for whom usage repair was not performed.

Also in *cloudian-admin.log*, the `CassandraUsageAccess::repairDirtyUsers` component writes two INFO messages for each user processed — one message indicating the start of processing the user and one message indicating the completion of processing the user. If a correction was made to the user's Redis QoS counts for stored bytes and/or stored objects, a third INFO message is sandwiched between the other two, indicating "Processed storage update: " and the correct counts.

### 12.12.6.0.1. Example Using cURL

The [example](#) below checks and repairs storage usage data for "dirty" users. It will also update group-level and system-level storage usage counts based on the repaired user-level counts.

```
curl -X POST -k -u sysadmin:public https://localhost:19443/usage/repair/dirtyusers
```

### 12.12.6.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing or invalid parameters

## 12.12.7. POST /usage/repair/user

### POST /usage/repair/user Repair storage usage data for a user

The request line syntax for this method is as follows.

```
POST /usage/repair/user?groupId=xxx&userId=xxx[&region=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"usage Query Parameters"** (page 880).

There is no request payload.

This method checks and repairs storage usage data for a single specified user. For background information on storage usage data repair, see **"Validating Storage Usage Data"** (page 143).

**Note** This operation does not update the group-level usage counters for the group to which the user belongs. For information about doing the latter, see [POST /usage/repair](#) — particularly the "summarizeCountsOnly" option. This is relevant especially when you are repairing multiple individual users

within a group, one at a time, using the `POST /usage/repair/user` method. In that case you should subsequently update the group-level usage counters for the group, using the [POST /usage/repair](#) method with the "summarizeCountsOnly" option.

#### 12.12.7.0.1. Example Using cURL

The [example](#) below checks and repair storage usage data for the user "gladdes" in the "engineering" group.

```
curl -X POST -k -u sysadmin:public \
'https://localhost:19443/usage/repair/user?groupId=engineering&userId=gladdes'
```

#### 12.12.7.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing or invalid parameters

### 12.12.8. POST /usage/rollup

#### POST /usage/rollup Roll up usage data

The request line syntax for this method is as follows.

```
POST /usage/rollup?granularity=xxx&startTime=xxx&unitCount=xxx
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"usage Query Parameters"** (page 880).

There is no request payload.

This method triggers the generation of "rollup" (aggregated across a time interval) service usage data from more granular data. Hourly rollup data is derived from "raw" transactional data. Daily rollup data and monthly rollup data are derived from hourly rollup data.

This method does not return the rolled up service usage data in the response, it only generates the rollup data and stores it in the system. To retrieve raw or rolled-up service usage data use the [GET /usage](#) method.

**Note** The `POST /usage/rollup` method is called regularly by HyperStore **"Usage Data Processing"** (page 474) cron jobs. The cron job to create hourly rollup data runs each hour; the cron job to create daily rollup data runs once per day; and the cron job to create monthly rollup data runs once per month.

In a multi-region system the rollup operations act only on usage data in the local service region. Consequently, cron jobs that trigger these operations are configured in each region.

#### 12.12.8.0.1. Example Using cURL

The [example](#) below creates hour roll-up usage data for the hour from midnight to 1AM on August 15, 2017.

```
curl -X POST -k -u sysadmin:public \
'https://localhost:19443/usage/rollup?granularity=hour&startTime=2017081500&unitCount=1'
```

### 12.12.8.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing or invalid parameters

## 12.12.9. POST /usage/storage

**POST /usage/storage** Post raw storage usage data for users with recent activity

The request line syntax for this method is as follows.

```
POST /usage/storage
```

There is no request payload.

The Redis QoS database maintains per-user and per-group counters for stored bytes and number of stored objects, based on transaction data that it receives from the S3 Service. This Admin API method writes these Redis-based stored bytes and stored object counts to the "Raw" column family in the Cassandra "Reports" key-space. Subsequently the [POST /usage/rollup](#) method can be used to roll up this "Raw" data into hourly, daily, and monthly aggregate data in Cassandra.

This API method applies only to users who have uploaded or deleted objects since the last time this method was executed.

**Note** This API method is triggered every 5 minutes by a HyperStore **"Usage Data Processing"** (page 474) cron job. The method acts only on usage data in the local service region. Consequently, in a multi-region system, cron jobs that trigger this method are automatically configured in each region.

### 12.12.9.0.1. Example Using cURL

The [example](#) below triggers the writing of raw stored bytes and stored objects counts into Cassandra, for users who have been active since the last running of this API method.

```
curl -X POST -k -u sysadmin:public https://localhost:19443/usage/storage
```

### 12.12.9.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746).

## 12.12.10. POST /usage/storageall

### POST /usage/storageall Post raw storage usage data for all users

The request line syntax for this method is as follows.

```
POST /usage/storageall
```

There is no request payload.

This method performs the same operation as described for [POST /usage/storage](#) except it applies to **all** users, not just recently active users.

**Note** This API method is triggered once each day by a HyperStore **"Usage Data Processing"** (page 474) cron job. The method acts only on usage data in the local service region. Consequently, in a multi-region system, cron jobs that trigger this method are automatically configured in each region.

#### 12.12.10.0.1. Example Using cURL

The [example](#) below triggers the writing of raw stored bytes and stored objects counts into Cassandra, for all users.

```
curl -X POST -k -u sysadmin:public https://localhost:19443/usage/storageall
```

#### 12.12.10.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746).

## 12.12.11. usage Query Parameters

### *granularity*

(Mandatory, string) With a *GET /usage* or *POST /usage/rollup* or *DELETE /usage* request: The time period granularity of the usage data to retrieve or generate or delete. Supported values are:

- *hour* — Hourly rollup data
- *day* — Daily rollup data
- *month* — Monthly rollup data
- *raw* — Raw transactional data (not rolled up). This is supported only for a GET or DELETE.

**Note** For a GET with granularity "raw", the interval between "startTime" and "endTime" must not exceed 24 hours. If the interval is larger than this, a 400 Bad Request response will be returned.

### *startTime*

(Mandatory, string)

The start time in **GMT**.

With a *GET /usage* request this is the start time of the interval for which to retrieve usage data. Format is *yyyyMMddHHmm*.

**Note** For retrieving **bucket** usage data, the start time's "*mm*" -- the minutes -- must be 00.

With a *POST /usage/rollup* or *DELETE /usage* request the format depends on the granularity of the usage data that you are generating or deleting:

- For hourly rollup data use format *yyyyMMddHH*. The start time will be the start of the hour that you specify.
- For daily rollup data use format *yyyyMMdd*. The start time will be the start of the day that you specify.
- For monthly rollup data use format *yyyyMM*. The start time will be the start of the month that you specify.
- For raw data use format *yyyyMMddHHmm*. The start time will be the start of the minute that you specify. This level of granularity is not supported for *POST /usage/rollup*.

#### *unitCount*

(Optional, integer) With a *POST /usage/rollup* or *DELETE /usage* request: The number of units of the specified "granularity" to generate or delete. Supported range is [1,100].

For example, with "granularity" = hour and "unitCount" = 24, a *DELETE /usage* operation will delete 24 hours worth of hourly rollup data, starting from your specified "startTime". In the case of "granularity" = raw, a *DELETE /usage* operation will delete "unitCount" minutes worth of raw transactional data -- for example 10 minutes worth of raw transactional data if "unitCount" = 10.

Defaults to 1 unit if not specified.

#### *endTime*

(Mandatory, string) With a *GET /usage* request: The end time **in GMT** of the interval for which to retrieve usage data. Format is *yyyyMMddHHmm*.

**Note** For retrieving **bucket** usage data, the end time's "*mm*" -- the minutes -- must be 00.

#### *id*

(Optional, string) With a *GET /usage* request: The identifier of a user or group for which to retrieve usage data, in format "<groupId>|<userId>" (for example "Dev|dstone", where Dev is the group ID and dstone is the user ID). To retrieve usage data for a whole group rather than a single user, use "<groupId>|\*" (for example "Dev|\*").

Do not use the "id" parameter for users who have been deleted from the system. For deleted users, use the "canonicalUserId" parameter described below.

With a *GET /usage* request you must use **either** "id" **or** "canonicalUserId" **or** "bucket". Do not use more than one of these query parameters.

#### *canonicalUserId*

(Optional, string) With a *GET /usage* request: The system-generated canonical ID of a user for which to retrieve usage data. Use this parameter if you want to retrieve usage data for a user who has been

deleted from the system. If you don't know the user's canonical ID, you can obtain it by using the *GET /user/list* method (this method can retrieve user profile information — including canonical ID — for all deleted users within a specified group).

With a *GET /usage* request you must use **either** "id" **or** "canonicalUserId" **or** "bucket". Do not use more than one of these query parameters.

#### *bucket*

With a *GET /usage* request (Optional, string): The bucket name. Use this parameter if you want to retrieve usage data for a specific bucket (rather than for a user or group). With a *GET /usage* request you must use **either** "id" **or** "canonicalUserId" **or** "bucket". Do not use more than one of these query parameters. Note that bucket names are globally unique within a HyperStore system, so specifying a bucket name is sufficient to uniquely identify a bucket.

With *POST /usage/repair/bucket* (Mandatory, string): The bucket name.

**Note** With the exception of the *POST /usage/repair/bucket* method, bucket usage statistics are disabled by default. For information on enabling this feature see "**Enabling Advanced Usage Reporting Features**" (page 142).

#### *operation*

(Mandatory, string) With a *GET /usage* request: The type of service usage data to retrieve. Supported values are:

- *SB* — Number of stored bytes
- *SO* — Number of stored objects
- *HG* — Number of S3 HTTP GET requests (includes HEADs also). The returned usage data also includes information about bytes downloaded.
- *HP* — Number of S3 HTTP PUT requests (includes POSTs also). The returned usage data also includes information about bytes uploaded.
- *HD* — Number of S3 HTTP DELETE requests

**Note** Usage tracking and reporting for the HG, HP, and HD metrics is disabled by default. For information on enabling these metrics see "**Enabling Advanced Usage Reporting Features**" (page 142).

- *BI* — For bucket usage only, the number of data transfer bytes IN (bytes of data uploaded).
- *BO* — For bucket usage only, the number of data transfer bytes OUT (bytes of data downloaded).

**Note** The BI and BO operation types are supported only for *GET /usage?bucket* requests. For user and group level usage statistics, the inbound and outbound data transfer size information is included within the HP and HG operation type usage data.

- *TB* — For bucket usage only, the total bytes count for the bucket.
- *TO* — For bucket usage only, the total objects count for the bucket.

**Note** The TB and TO operation types are supported only for *GET /usage?bucket* requests. The TB and TO counts for a bucket for a specified time period (from *startTime* to *endTime*) will exist only if you previously executed the *POST /usage/repair/bucket* method during that time period. That method generates the TB and TO counts for the bucket which the system then stores along with a timestamp indicating when the count was generated.

For *GET /usage?bucket* requests the SB and SO operation types are also supported, but these will return the **change** in the stored bytes and stored object counts during the specified time interval -- for example, the total increase in a bucket's stored bytes total during a specified day, rather than the total number of bytes in the bucket on that day. For the latter you would use the TB operation type.

#### *reversed*

(Optional, boolean) With a *GET /usage* request: If this is set to "false", the retrieved usage data results will be listed in chronological order. If this is set to "true", results will be listed in reverse chronological order. Defaults to "false" if not specified.

**Note** This parameter is not supported if you are retrieving **bucket** usage data.

#### *limit*

(Optional, integer) With a *GET /usage* request: The maximum number of results to return — that is, the maximum number of *<UsageData>* objects to return in the response body — if pagination is not used (if no "pageSize" value is specified).

Defaults to 10,000 if not specified.

#### *pageSize*

(Optional, integer) With a *GET /usage* request: For pagination, the maximum number of results to return per request. If a "pageSize" is specified, this supersedes the "limit" value.

Defaults to 0 if not specified.

**Note** This parameter is not supported if you are retrieving **bucket** usage data.

#### *offset*

(Optional, integer) With a *GET /usage* request: If you use the "pageSize" parameter in support of paginating a large result set, in the response the system will return one additional result beyond your specified "pageSize" value (for example, if you specify "pageSize=25", the system will return 26 results). From the extra result (listed last in the response body), use the result's timestamp as the "offset" parameter value in your next request. That result will then be the first of the results returned for that request.

For each request you submit, the last of the returned results will be an extra result from which you can use the timestamp as the "offset" value for the next request. If there is no extra result in the response, that indicates that the result set has been exhausted.

Defaults to 0 if not specified.

**Note** This parameter is not supported if you are retrieving **bucket** usage data.

#### *region*

(Optional, string)

With a *GET /usage* request: The region for which to retrieve usage data. To retrieve usage data for all regions, specify the string "ALL". If no "region" value is specified, the default region is assumed. This parameter is not supported if the "bucket" parameter is used (for retrieving data for a specified bucket).

With a *POST /usage/repair* or *POST /usage/repair/user* request: The region for which to perform the usage data repair. If the region parameter is not specified, the repair is performed for all service regions.

**Note** *GET /usage* requests for user or group level statistics should be submitted only to the Admin Service in the default region. Use the "region" query parameter to specify the region for which you want to retrieve usage data.

*GET /usage* requests for **bucket** usage data can be submitted to the Admin Service any region, and the results will be from that region. Do not specify the "region" parameter for bucket usage data requests.

#### *regionOffset*

(Optional, string) With a *GET /usage* request: If you use a "region" value of "ALL", use the "regionOffset" parameter to specify the region name of your local region. This helps with pagination of the result set.

**Note** This parameter is not supported if you are retrieving **bucket** usage data.

#### *groupId*

(Mandatory, string)

With a *POST /usage/repair* request: The group for which to repair user-level and group-level storage usage counts. If groupId is "ALL", repair is performed for all groups.

With a *POST /usage/repair/user* request: The ID of the group to which the target user belongs.

#### *summarizeCountsOnly*

(Optional, boolean) With a *POST /usage/repair* request: If set to "true" while "groupId" = a specific group, then the operation will not validate or repair usage data counters for individual users within the specified group. Instead, it will presume the user-level counters to be correct, and will only sum up the user-level counters in order to update the counters for the group as a whole. This option is useful after you have been running *POST /usage/repair/user* operations (which validate and repair usage counters for individual users without updating the group-level counters for the groups that those users belong to).

If set to "true" while "groupId" = ALL, then the operation will only sum up the existing group-level usage counters to update the counters for the system as a whole.

If set to "false", then the operation runs in the normal manner, by first validating and repairing user-level usage counters within the specified group and then using that repaired data to update the group-level counters for the group.

Defaults to "false" if the "summarizeCountsOnly" parameter is omitted.

*summarizeCounts*

(Optional, boolean) With a *POST /usage/repair/dirtyusers* request: If set to "true", then the *POST /usage/repair/dirtyusers* operation, after repairing usage counters for individual users, will update the group-level usage counters for the groups to which those repaired users belong. It will then also update system-level usage counts, based on the updated group counters.

If set to "false", then the operation will repair only user-level counters, and will not update the group or whole-system counters.

Defaults to "true".

*userId*

(Mandatory, string) With a *POST /usage/repair/user* request: The ID of the user for whom usage data repair is to be performed.

## 12.12.12. usage Objects

This section describes the JSON-formatted data objects that constitute the request or response payloads for the Usage related Admin API methods.

**Note** For examples of these objects see the API method request and response examples.

- **"UsageBucketReq Object"** (page 885)
- **"UsageBucketRes Object"** (page 885)
- **"UsageData Object"** (page 886)

### 12.12.12.1. UsageBucketReq Object

The *UsageBucketReq* object consists of the following attributes:

*buckets*

(Mandatory, list<string>) List of the buckets for which to retrieve raw usage data. Example:

```
"buckets": ["b123", "mybucket"]
```

*endTime*

(Mandatory, string) End time (in GMT) of the interval for which to retrieve raw usage data. Format is *yyyyMMddHHmm*. Example:

```
"endTime": "201611291900"
```

*startTime*

(Mandatory, string) Start time (in GMT) of the interval for which to retrieve raw usage data. Format is *yyyyMMddHHmm*. Example:

```
"startTime": "201611291800"
```

### 12.12.12.2. UsageBucketRes Object

The *UsageBucketRes* object consists of the following attributes:

*bucket*

(String) Bucket with which the usage data is associated. Example:

```
"bucket": "b123"
```

*data*

(List<UsageData>) List of [UsageData](#) objects. Note that in the context of a *UsageBucketRes* object, the *UsageData* objects will always be for "raw" granularity. Example:

```
"data": [  
  {  
    "averageValue": "3222",  
    "bucket": "b123",  
    "count": "0",  
    "groupId": null,  
    "ip": "10.10.0.1",  
    "maxValue": "0",  
    "operation": "BO",  
    "region": null,  
    "timestamp": "1480442520000",  
    "uri": null,  
    "userId": null,  
    "value": "3222",  
    "whitelistAverageValue": "0",  
    "whitelistCount": "0",  
    "whitelistMaxValue": "0",  
    "whitelistValue": "0"  
  }  
]
```

### 12.12.12.3. UsageData Object

The *UsageData* object consists of the following attributes:

*averageValue*

(String) Average value of the usage statistic during the granularity interval.

For user level or group level statistics, when usage report granularity = **hour**, **day**, or **month**, the "averageValue" will equal the "value" divided by the "count".

When usage report granularity = **raw** or for bucket usage statistics of any granularity, the "averageValue" will equal the "value".

Example:

```
"averageValue": "107956"
```

*bucket*

(String) Name of the bucket with which the usage data is associated. This attribute will have a value only for bucket usage data. For user level or group level usage data this attribute will have *null* value.

Example:

```
"bucket": null
```

*count*

(String) Data count. The specific meaning of this attribute depends on the usage reporting granularity and operation type.

When usage report granularity = **raw** or for bucket usage statistics of any granularity, "count" is not relevant and always returns a "0".

For user level or group level statistics, when usage report granularity = **hour**, **day**, or **month**:

- For operation type SB or SO:
  - For granularity **hour**, the "count" will always be "1".
  - For granularity **day**, the "count" will be the number of hourly data points recorded by the system within the day. For a past day, this will be "24"; for the current day, this will be the number of hours that have completed so far within the day.
  - For granularity **month**, the "count" will be the number of hourly data points recorded by the system within the month. For a past month, this will be the total number of days in that month X 24 hours-per-day; for the current month, this will be the number of hours that have completed so far in the month.

**Note** For SB and SO, the "count" is relevant only insofar as it is used as the denominator in the calculation of an average storage value for the granularity interval (the numerator in the calculation is the "value" attribute).

- For operation type HG, HP, or HD, the "count" is the count of requests within the granularity interval (within the hour, day, or month). For example, if the operation type is HD and the granularity is hour, this is the count of HTTP Delete requests during the hour. Requests from whitelisted source IP addresses are excluded from HG, HP, or HD counts (unless the usage data is for a specific bucket, in which case the whitelist feature does not apply and whitelisted source addresses are not treated any differently than other source addresses in regard to usage tracking.)

Example:

```
"count": "744"
```

#### *groupId*

(String) Group ID with which the usage data is associated.

For bucket usage this attribute will have *null* value.

Example:

```
"groupId": "QA"
```

#### *ip*

(String) IP address of the client that submitted an S3 request. Applicable only if the usage reporting granularity is **raw** and the operation type is HG, HP, HD, BI, or BO. Otherwise this attribute will have *null* value.

Example:

```
"ip": ""
```

#### *maxValue*

(string) Maximum value recorded during the granularity interval. For example, for operation type SB this

would be the largest storage byte level reached during the granularity interval. The "maxValue" is reported only for rollup granularities (hour, day, month). For raw granularity and for bucket usage data of any granularity it will have a value of "0".

Requests from whitelisted source addresses are excluded from HG, HP, HD "maxValue".

Example:

```
"maxValue": "305443"
```

#### *operation*

(String) Operation type for which the usage statistics are reported:

- SB = Storage Bytes
- SO = Storage Objects
- HG = S3 HTTP GETs (and HEADs)
- HP = S3 HTTP PUTs (and POSTs)
- HD = S3 HTTP DELETEs
- BI = For bucket usage only, the data transfer IN bytes
- BO = For bucket usage only, the data transfer OUT bytes
- TB = For bucket usage only, the total bytes count for the bucket.
- TO = For bucket usage only, the total objects count for the bucket.

**Note** The TB and TO operation types are supported only for bucket usage statistics. The TB and TO counts for a bucket for a specified time period (from `startTime` to `endTime`) will exist only if you previously executed the `POST /usage/repair/bucket` method one or more times during that time period. That method generates the TB and TO counts for the bucket, which the system then stores along with a timestamp indicating when the counts were generated. It's these saved TB and TO counts that are returned by `GET /usage` for the bucket. In the case of rolled-up usage data, the most recent TB and TO counts from within the roll-up period are used.

The SB and SO operation types are also supported for bucket usage statistics, but these will return the **change** in the stored bytes and stored object counts during the specified time interval -- for example, the total increase in a bucket's stored bytes total during each day in the interval (if you are using granularity "day"), rather than the total number of bytes in the bucket on each day. The latter is captured by the TB operation type.

Example:

```
"operation": "SB"
```

#### *policyId*

(String) System-generated unique ID of the storage policy used by the bucket.

This attribute is relevant only to bucket usage data and will be *null* for group or user-level usage data.

Example:

```
"policyId": "880e7d065225009b481ff24ae8d893ce"
```

#### *region*

(String) Service region in which the usage occurred.

For bucket usage this attribute will have *null* value.

Example:

```
"region": "taoyuan"
```

#### *timestamp*

(String) Timestamp for creation of this usage data, in UTC milliseconds. The specific meaning of the timestamp depends on the usage reporting granularity and operation type.

When usage report granularity = **raw**:

- For operation type SB or SO, the "timestamp" is the time when the storage level was recorded by the `/usage/storage` API call (which is run by cron job every five minutes)
- For operation type HG, HP, HD, BI, or BO, the "timestamp" is the time when the transaction occurred.
- For operation type TB or TO (supported for bucket usage only), the "timestamp" is the time when the `POST /usage/repair/bucket` call that calculated the TB and TO counts was executed.

When usage report granularity = **hour**, **day**, or **month**:

- The "timestamp" is the time that the granularity interval started (the start of the hour, day, or month for which data is encapsulated in the *UsageData* object).

Example:

```
"timestamp": "1498867200000"
```

#### *uri*

(String) URI of the data object. Applicable only for user and group level usage data and only if the usage reporting granularity is "raw". For user and group level usage data with granularity other than "raw", this attribute will have an empty value.

For bucket usage this attribute will have a *null* value.

Example:

```
"uri": ""
```

#### *userId*

(String) User ID with which the usage data is associated. For group level usage data the *userId* attribute will be `"*"`.

For bucket usage this attribute will have *null* value.

Example:

```
"userId": "*"
```

#### *value*

(String) Data value. The specific meaning of this attribute depends on the usage reporting granularity and operation type.

When usage report granularity = **raw**:

- For operation type SB or SO (or TB or TO in the case of bucket usage statistics), the "value" is the current storage bytes or current number of stored objects.
- For operation type HG, HP, HD, BI, or BO, the "value" is the data transfer size for the single transaction, in bytes.

**Note** With Multipart Upload operations (for large objects), each part upload counts as a separate transaction toward the HP and BI statistics.

When usage report granularity = **hour**, **day**, or **month**:

- For operation type SB or SO for user or group level usage data, the "value" is the sum of the storage level measures recorded by the system during the granularity interval, in bytes. For example, for granularity day, a current SB measure is recorded for each hour during the day, and the sum of those hourly measures is the SB "value" for the day. For SB and SO, this aggregate "value" is relevant only insofar as it is used as the numerator in the calculation of an average storage value for the granularity interval (the denominator in the calculation is the "count" attribute).

**Note** In the case of bucket usage data the hour, day, or month "rollup" value for SB or SO is the **change** to the stored byte or stored object count in the bucket during the rollup period.

- For operation type HG, HP, HD, BI, or BO, the "value" is the sum data transfer size for the granularity interval, in bytes. For example, if operation type is HP and the granularity is hour, the "value" is the aggregated data transfer size of all HTTP PUT and POST requests during the hour.

Requests from whitelisted source IP addresses are excluded from HG, HP, and HD values (unless the usage data is for a specific bucket, in which case the whitelist feature does not apply and whitelisted source addresses are not treated any differently than other source addresses in regard to usage tracking.)

Example:

```
"value": "80319535"
```

#### *whitelistAverageValue*

(String) Same as "averageValue" above, except this is exclusively for traffic from whitelisted source addresses.

**Note** For bucket usage data, traffic from whitelisted sources is bundled in with the main usage statistics rather than being separated out. For bucket usage all "whitelist\*" attributes will have "0" as their value.

Example:

```
"whitelistAverageValue": "0"
```

#### *whitelistCount*

(String) Same as "count", except this is exclusively for traffic from whitelisted source addresses.

Example:

```
"whitelistCount": "0"
```

*whitelistMaxValue*

(String) Same as "maxValue" above, except this is exclusively for traffic from whitelisted source addresses. Example:

```
"whitelistMaxValue": "0"
```

*whitelistValue*

(String) Same as "value", except this is exclusively for traffic from whitelisted source addresses. Example:

```
"whitelistValue": "0"
```

**12.12.12.3.1. Usage Data Topics****How Particular S3 Operations Impact Usage Data Counts**

To support usage reporting, billing, and the implementation of Quality of Service (QoS) limits, the following counters are maintained for individual users and for groups:

- Storage bytes
- Storage objects
- Number of requests
- Data bytes IN
- Data bytes OUT

When calculating size for storage byte tracking, the size of the object metadata is included as well as the size of the object itself. If compression is used for storage of S3 objects, the uncompressed object size is counted toward storage byte tracking.

When calculating size for data transfer byte tracking (IN and OUT), the size of the HTTP headers is included as well as the size of the object itself.

The table below shows how particular S3 operations (the left-most column) impact the various service usage counters (shown in the remaining columns).

Operation	Storage Bytes	Storage Objects	Num Requests	Bytes IN	Bytes OUT
DELETE (Add delete marker)	Add Total-Size which is same as size of object path including bucketname (i.e., <bucketname>/<objectname>), unless replacing existing delete marker, then no change	Incremented by 1, unless replacing existing DM, then no change	No change	No change	No change
DELETE (No delete marker added) object, bucket	If object is successfully deleted, decremented by Total-Size of deleted object. If request is to region where bucket is not located, no change.	If object is successfully deleted, decremented by 1. If request is to region where bucket is not located, no	No change	No change	No change

Operation	Storage Bytes	Storage Objects	Num Requests	Bytes IN	Bytes OUT
		change.			
DELETE object tagging	Decrementd by size of old tagging string	No change	No change	No change	No change
DELETE policy	No change	No change	No change	No change	No change
DELETE uploadId (MP Abort)	If successfully deleted, decremented by Total-Size of uploaded parts and 1V value added in MP initiate	If successfully deleted, decremented by 1	No change	No change	No change
GET bucket, service, policy, location, acl, bucketlogging, versioning, list uploads, list parts	No change	No change	Incremented by 1	Transfer-Size of request	Transfer-Size of response
GET object	No change	No change	Incremented by 1	Transfer-Size of request	Transfer-Size of response
GET object tagging	No change	No change	Incremented by 1	Transfer-Size of request	Transfer-Size of response
HEAD object	No change	No change	Incremented by 1	Transfer-Size of request	Transfer-Size of response
POST MP initiate	Add object name size and metadata size.	Incremented by 1	Incremented by 1	Add Transfer-Size of request	Add Transfer-Size of response
POST MP complete	If replacement object, decrement by Total-Size of old object. Total size of completed object metadata is set to total size of MP parts and initiate request.	If replacement object, decrement 1	Incremented by 1	Add Transfer-Size of request	Add Transfer-Size of response
POST object	Incremented by Total-size minus Total-size of old object, if any	Incremented by 1 if new object	Incremented by 1	Transfer-Size of request	Transfer-Size of response
PUT bucket	Incremented by bucketname size if bucket created in region, otherwise 0	Incremented by 1 if bucket created in region, otherwise 0	Incremented by 1	Transfer-Size of request	Transfer-Size of response
PUT bucket log-	Incremented by Total-Size of log	Incremented	No change	No	No

Operation	Storage Bytes	Storage Objects	Num Requests	Bytes IN	Bytes OUT
ging object	object	by 1		change	change
PUT part	Add Content-Length of part body. If replacing an existing part, subtract Content-Length of old part body.	No change	Incremented by 1	Add Transfer-Size of request	Add Transfer-Size of response
PUT object	Incremented by Total-size minus Total-size of old object, if any	Incremented by 1 if new object	Incremented by 1	Transfer-Size of request	Transfer-Size of response
PUT object CRR ( <a href="#">cross-region replication</a> )	Incremented by Total-size of original object and replica object combined, plus 51 bytes of metadata associated with implementing CRR	Incremented by one if new object	Incremented by 1	Transfer-Size of request	Transfer-Size of response
PUT object copy	Two cases: (1) Metadata COPY. Increment by source total-size + difference between new and old object name. (2) Metadata REPLACE. Increment by source content-length + new objectname + new meta headers. In both cases, if replacement object, then Total-Size of replacement object is subtracted.	Incremented by one if new object	Incremented by 1	Transfer-Size of request	Transfer-Size of response
PUT policy, logging, acl, versioning	No change	No change	Incremented by 1	Transfer-Size of request	Transfer-Size of response
PUT object tagging	Incremented by size of new tagging string minus size of old tagging string (if any)	No change	Incremented by 1	Transfer-Size of request	Transfer-Size of response
Upload Part Copy	Increment by source content-length. If replacement object, then Total-Size of replacement object is subtracted.	No change	Incremented by 1	Transfer-Size of request	Transfer-Size of response

## How Request Processing Errors Impact Usage Counts

If an S3 request for uploading or downloading data fails to complete due to a processing error within the HyperStore system, the request still counts towards the data transfer bytes total for usage tracking and QoS implementation. For example, if a user tries to upload a 1MB object and the request fails to complete, the 1MB is still added to the user's total for Data Bytes In. It would not impact the user's Stored Bytes or Stored Objects counts.

## How Auto-Tiering Impacts Usage Counts

The HyperStore system supports an S3 **"PutBucketLifecycle"** (page 969) API extension whereby objects can, on a specified scheduled, be auto-tiered to Amazon S3, Amazon Glacier, a different Cloudian HyperStore service region, or a third party Cloudian HyperStore system. When an object is transitioned to Amazon or a

different HyperStore region or system, its size is removed from the Storage Bytes count in the local HyperStore region. At the same time, a reference to the transitioned object is created and the size of this reference — 8KB, regardless of the transitioned object size — is added to the local Storage Bytes count. For example, if a 100KB object is auto-tiered to Amazon or a different HyperStore region or system, the net local effect is a 92KB reduction in the local Storage Bytes count.

If the object is temporarily restored to local HyperStore storage (through the S3 API method POST Object restore), then while the object is locally restored the object's size is added to the local Storage Bytes count and the 8KB for the reference is subtracted from the count. After the restore interval ends, the object size is once again subtracted from Storage Bytes and the 8KB for the reference is added back.

Auto-tiering does **not** impact the Storage Objects count.

**Note** In regard to the maximum stored bytes that your license permits you — for objects that have been auto-tiered to Amazon, the size of the tiered objects does not count toward your maximum allowed storage capacity. However, the 8KB per tiered object (described above) **does** count toward your licensed maximum storage capacity.

## How Server-Side Encryption Impacts Usage Counts

For [Server-Side Encryption](#) where the objects are encrypted in storage, "Bytes In" and "Bytes Out" reflect the original, unencrypted object size. The "Storage Bytes" value uses the encrypted object size. Headers are not encrypted, and thus not included. The increase of size of the encrypted object, i.e., the "padding size", depends on the AES block size and the amount of padding required.

The padding formula for AES/CBC/PKCS5 padding is as described below.

```
AES block size = 16
```

```
In the PKCS5 padding always a pad block is added at the end. So the padding
bytes vary from 1 to 16.
```

```
*Non-chunked objects*
```

```
Cipher size = (plain text size / 16 + 1) * 16.
```

```
Padding size = cipher size - plain text size
```

```
For example:
```

```
20 bytes object: total cipher size = (20/16 + 1) * 16 = 32 bytes
```

```
11 bytes object: total cipher size = (11/16 + 1) * 16 = 16 bytes
```

```
*Chunked objects*
```

```
Number of full chunks = plain text size / max chunk size
```

```
Last (partial) chunk size = plain text size % max chunk size
```

```
Cipher chunk size = (max chunk size / 16 + 1) * 16
```

```
- If last (partial) chunk size == 0
  last chunk padding size = 0
```

```
- If last (partial) chunk size > 0
  cipher last (partial) chunk size = (last (partial) chunk size / 16 + 1) * 16
```

```

last chunk padding size = cipher last (partial) chunk size - last (partial) chunk size

padding size = number of full chunks * (cipher chunk size - max chunk size)
               + last chunk padding size

```

For example:

max chunk size=1024

```

1024 bytes object: total cipher size = plain text size + padding size
                                   = 1024 + 1*(1040-1024) + 0
                                   = 1024 + 16
                                   = 1040

```

```

1025 bytes object: total cipher size = plain text size + padding size
                                   = 1025 + 1*(1040-1024) + 15
                                   = 1025 + 16 + 15
                                   = 1056

```

## How Compression Impacts Usage Counts

For S3 service usage tracking (for purposes of QoS enforcement and billing), the uncompressed size of objects is always used, even if you [enable compression](#) for all or some of your storage policies.

## 12.13. user

The Admin API methods built around the **user** resource are for managing HyperStore service user accounts. This includes support for creating, changing, and deleting user accounts. These methods also support management of users' security credentials and the assignment of rating plans to users.

### 12.13.1. DELETE /user

#### DELETE /user Delete a user

The request line syntax for this method is as follows.

```
DELETE /user? [userId=xxx&groupId=xxx|canonicalUserId=xxx]
```

Either use *userId* together with *groupId*, or use *canonicalUserId*.

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"user Query Parameters"** (page 919).

There is no request payload.

The user will be removed from his group; his security credentials will be deleted; and his S3 buckets and stored objects will be deleted. **Once deleted, a user's buckets and objects will not be recoverable.**

The operations associated with deleting a user are performed asynchronously. If you receive an OK response to a *DELETE /user* request, this indicates that the user's status has successfully transitioned to "deleting", and the associated operations are underway. You can use the [GET /user/list](#) method to check on which users within a group are in "deleting" status or "deleted" status ("deleted" status indicates that all associated operations have completed, including deletion of the user's stored S3 buckets and objects).

**Note** Service usage report data for a deleted user is retained for a period of time as configured by the *reports.rollup.ttl* setting in *mts.properties.erb*. You can retrieve usage data for a recently deleted user via the [GET /usage](#) method.

**Note** You cannot delete the default system administrator account. This is not allowed.

#### 12.13.1.0.1. Example Using cURL

The [example](#) below deletes a user with ID "John" who is in the "QA" group.

```
curl -X DELETE -k -u sysadmin:public \
'https://localhost:19443/user?userId=John&groupId=QA'
```

#### 12.13.1.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing Required parameters : {userId, groupId}
400	User does not exist

### 12.13.2. DELETE /user/credentials

**DELETE /user/credentials** Delete a user's S3 security credential

The request line syntax for this method is as follows.

```
DELETE /user/credentials?accessKey=xxx
```

For parameter description click on the parameter name or see **"user Query Parameters"** (page 919).

There is no request payload.

#### 12.13.2.0.1. Example Using cURL

The [example](#) below deletes a user's S3 credential as specified by the access key. Note that since each S3 access key is unique in the system, you do not need to specify the user to whom the key is assigned.

```
curl -X DELETE -k -u sysadmin:public \
https://localhost:19443/user/credentials?accessKey=21289bab1738ffdc792a
```

#### 12.13.2.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameters : {accessKey}

Status Code	Description
400	Invalid Access Key

### 12.13.3. DELETE /user/deleted

#### DELETE /user/deleted Purge profile data of a deleted user or users

The request line syntax for this method is as follows.

```
DELETE /user/deleted[?canonicalUserId=xxx|groupId=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"user Query Parameters"** (page 919).

After deleting a user or users, you can use this Admin API method if you want to purge the deleted users' profile information from the Cassandra database. Otherwise, the deleted users' profile information is retained in Cassandra indefinitely.

Use the *canonicalUserId* parameter to specify just a single user for whom to purge profile data, **or** use the *groupId* parameter to purge profile data for all deleted users in the specified group. Do not use both a *canonicalUserId* and a *groupId* together.

There is no request payload.

**Note** If you purge a deleted user's profile information, you will no longer be able to retrieve that user's profile information via the [GET /user/list](#) method. This means that you will no longer be able to retrieve the deleted user's canonical user ID. Without a deleted user's canonical user ID, you will not be able to retrieve usage history for the user. Consequently, you should purge a deleted user's profile information **only if** you have some independent record of the user's canonical user ID (outside of the Cassandra database); or if you are confident that you will no longer require access to the deleted user's usage history.

#### 12.13.3.0.1. Example Using cURL

The [example](#) below purges a single deleted user's profile data.

```
curl -X DELETE -k -u sysadmin:public \
https://localhost:19443/user/deleted?canonicalUserId=bd0796cd9746ef9cc4ef656ddaacf4
```

#### 12.13.3.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Conflicting or missing parameters : {canonicalUserId, groupId}
400	User does not exist or is not in a deleted state.

### 12.13.4. GET /user

#### GET /user    Get a user's profile

The request line syntax for this method is as follows.

```
GET /user?[userId=xxx&groupId=xxx|canonicalUserId=xxx]
```

Either use *userId* together with *groupId*, or use *canonicalUserId*.

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"user Query Parameters"** (page 919).

There is no request payload.

**Note** The HyperStore IAM Service supports a role-based access version of this API method as described in **"RBAC Version of this Method"** (page 899).

#### 12.13.4.0.1. Example Using cURL

The [example](#) below retrieves a user with ID "John" who is in the "QA" group.

```
curl -X GET -k -u sysadmin:public \  
'https://localhost:19443/user?userId=John&groupId=QA' | python -mjson.tool
```

The response payload is a JSON-formatted *UserInfo* object, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"UserInfo Object"** (page 922).

```
{  
  "active": "true",  
  "address1": "",  
  "address2": "",  
  "canonicalUserId": "bd0796cd9746ef9cc4ef656ddaacfac4",  
  "city": "",  
  "country": "",  
  "emailAddr": "",  
  "fullName": "John Thompson",  
  "groupId": "QA",  
  "ldapEnabled": false,  
  "phone": "",  
  "state": "",  
  "userId": "John",  
  "userType": "User",  
  "website": "",  
  "zip": ""  
}
```

#### 12.13.4.0.2. Response Format

The response payload is a JSON-formatted *UserInfo* object (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
204	User does not exist
400	Missing Required parameters : {userId, groupId}

#### 12.13.4.1. RBAC Version of this Method

HyperStore's IAM Service supports an RBAC version of this API method. For an overview of the RBAC feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- Action name: *GetCloudianUser*
- Parameters: Same as for *GET /user*, except all parameter names start with an upper case letter rather than lower case
- Response body: Same response data as for *GET /user* except the data is formatted in XML rather than JSON
- Role-based restrictions:
  - HyperStore system admin user can get any user's profile
  - HyperStore group admin user can only get the profiles of users within her own group
  - HyperStore regular user can only get own profile
  - IAM user can only use this method if granted *admin:GetCloudianUser* permission by an IAM policy, and subject to the same restriction as the parent HyperStore user.

**Note** The "GetCloudianUser" action retrieves profile data for Cloudian HyperStore user accounts, not for subsidiary IAM users. For example, if a HyperStore group administrator grants *admin:GetCloudianUser* permission to an IAM user, the IAM user will be able to retrieve profile information for **any HyperStore user in the group administrator's group**. And if a HyperStore regular user grants *admin:GetCloudianUser* permission to an IAM user, the IAM user will be able to retrieve profile information for the **parent HyperStore user**.

- Sample request and response (abridged):

##### REQUEST

```
http://localhost:16080/?Action=GetCloudianUser&UserId=John&GroupId=QA
```

<request headers including authorization info>

##### RESPONSE

200 OK

```
<?xml version="1.0" encoding="utf-8"?>
<GetCloudianUserResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">
  <ResponseMetadata>
    <RequestId>system-generated-request-id</RequestId>
  </ResponseMetadata>
  <CassandraUserInfo>
    <active>Active</active>
```

```

etc...
...
...
</CassandraUserInfo>
</GetCloudianUserResponse>

```

### 12.13.5. GET /user/credentials

#### GET /user/credentials Get a user's S3 security credential

The request line syntax for this method is as follows.

```
GET /user/credentials?accessKey=xxx
```

For parameter description click on the parameter name or see **"user Query Parameters"** (page 919).

There is no request payload.

**Note** The HyperStore IAM Service supports a role-based access version of this API method as described in **"RBAC Version of this Method"** (page 901).

#### 12.13.5.0.1. Example Using cURL

The [example](#) below retrieves the S3 credentials object corresponding to a specified S3 access key. Note that since each S3 access key is unique in the system, you do not need to specify the user to whom the key is assigned.

```

curl -X GET -k -u sysadmin:public \
https://localhost:19443/user/credentials?accessKey=009c156c79e64e0e4928 \
| python -mjson.tool

```

The response payload is a JSON-formatted *SecurityInfo* object, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"SecurityInfo Object"** (page 921).

```

{
  "accessKey": "009c156c79e64e0e4928",
  "active": true,
  "createDate": 1502279336024,
  "expireDate": null,
  "secretKey": "wVHk2nA0M03RWSMIrFHFAtuhow6S1DKN0gWjPhDG"
}

```

#### 12.13.5.0.2. Response Format

The response payload is a JSON-formatted *SecurityInfo* object (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
204	No Data Found
400	Missing required parameters : {accessKey}

### 12.13.5.1. RBAC Version of this Method

HyperStore's IAM Service supports an RBAC version of this API method. For an overview of the RBAC feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- Action name: *GetCloudianUserCredentials*
- Parameters: Same as for *GET /user/credentials*, except all parameter names start with an upper case letter rather than lower case
- Response body: Same response data as for *GET /user/credentials* except:
  - The data is formatted in XML rather than JSON
  - The *secretKey* is not included in the response
- Role-based restrictions:
  - HyperStore system admin user can get any user's credentials
  - HyperStore group admin user can only get the credentials of users within her own group
  - HyperStore regular user can only get own credentials
  - IAM user can only use this method if granted *admin:GetCloudianUserCredentials* permission by an IAM policy, and subject to the same restriction as the parent HyperStore user.

**Note** The "GetCloudianUserCredentials" action retrieves credentials for Cloudian HyperStore user accounts, not for subsidiary IAM users. For example, if a HyperStore group administrator grants *admin:GetCloudianUserCredentials* permission to an IAM user, the IAM user will be able to retrieve credentials for **any HyperStore user in the group administrator's group**. And if a HyperStore regular user grants *admin:GetCloudianUserCredentials* permission to an IAM user, the IAM user will be able to retrieve the **parent HyperStore user's** credentials.

- Sample request and response (abridged):

```
REQUEST

http://localhost:16080/?Action=GetCloudianUserCredentials&AccessKey=009c156c79e64e0e4928

<request headers including authorization info>

RESPONSE

200 OK

<?xml version="1.0" encoding="utf-8"?>
<GetCloudianUserCredentialsResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">
  <ResponseMetadata>
    <RequestId>system-generated-request-id</RequestId>
  </ResponseMetadata>
  <SecurityInfo>
    <accessKey>40602cdfaef3a676594d</accessKey>
    etc...
    ...
  </SecurityInfo>
```

```
</GetCloudianUserCredentialsResponse>
```

### 12.13.6. GET /user/credentials/list

#### GET /user/credentials/list    Get a user's list of S3 security credentials

The request line syntax for this method is as follows.

```
GET /user/credentials/list?[userId=xxx&groupId=xxx | canonicalUserId=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"user Query Parameters"** (page 919).

This retrieves all of the user's S3 credentials (active credentials as well as inactive [disabled] credentials).

Specify the user either by using *userId* and *groupId*, or by using *canonicalUserId*.

There is no request payload.

**Note** The HyperStore IAM Service supports a role-based access version of this API method as described in **"RBAC Version of this Method"** (page 903).

#### 12.13.6.0.1. Example Using cURL

The [example](#) below retrieves all of the S3 security credentials belonging to user "John" in the "QA" group.

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/user/credentials/list?userId=John&groupId=QA' \
| python -mjson.tool
```

The response payload is a JSON-formatted list of *SecurityInfo* objects, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"SecurityInfo Object"** (page 921).

```
[
  {
    "accessKey": "009c156c79e64e0e4928",
    "active": true,
    "createDate": 1502279336024,
    "expireDate": null,
    "secretKey": "wVHk2nA0M03RWSMIrFHFAtuhow6S1DKN0gWjPhDG"
  },
  {
    "accessKey": "21289bab1738ffdc792a",
    "active": false,
    "createDate": 1502283467021,
    "expireDate": null,
    "secretKey": "o5jqJtqV36+sENGLozEUg1EXEmQp9V6yfCHLFCJk"
  }
]
```

### 12.13.6.0.2. Response Format

The response payload is a JSON-formatted list of *SecurityInfo* objects (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
204	No Access Key found
400	Missing Required parameters : {userId, groupId}
400	User/Group does not exist

### 12.13.6.1. RBAC Version of this Method

HyperStore's IAM Service supports an RBAC version of this API method. For an overview of the RBAC feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- Action name: *GetCloudianUserCredentialsList*
- Parameters: Same as for *GET /user/credentials/list*, except all parameter names start with an upper case letter rather than lower case
- Response body: Same response data as for *GET /user/credentials/list* except:
  - The data is formatted in XML rather than JSON
  - The *secretKey* is not included in the response
- Role-based restrictions:
  - HyperStore system admin user can get any user's credentials list
  - HyperStore group admin user can only get the credentials list of users within her own group
  - HyperStore regular user can only get own credentials list
  - IAM user can only use this method if granted *admin:GetCloudianUserCredentialsList* permission by an IAM policy, and subject to the same restriction as the parent HyperStore user.

**Note** The "GetCloudianUserCredentialsList" action retrieves credentials for Cloudian HyperStore user accounts, not for subsidiary IAM users. For example, if a HyperStore group administrator grants *admin:GetCloudianUserCredentialsList* permission to an IAM user, the IAM user will be able to retrieve a credentials list for **any HyperStore user in the group administrator's group**. And if a HyperStore regular user grants *admin:GetCloudianUserCredentialsList* permission to an IAM user, the IAM user will be able to retrieve the **parent HyperStore user's** credentials list.

- Sample request and response (abridged):

```
REQUEST

http://localhost:16080/?Action=GetCloudianUserCredentialsList&UserId=John&GroupId=QA

<request headers including authorization info>

RESPONSE
```

```

200 OK

<?xml version="1.0" encoding="utf-8"?>
<GetCloudianUserCredentialsListResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">
  <ResponseMetadata>
    <RequestId>system-generated-request-id</RequestId>
  </ResponseMetadata>
  <ListWrapper>
    <SecurityInfo>
      <accessKey>40602cdfaef3a676594d</accessKey>
      etc...
    </SecurityInfo>
    <SecurityInfo>
      etc...
    </SecurityInfo>
  </ListWrapper>
</GetCloudianUserCredentialsListResponse>

```

### 12.13.7. GET /user/credentials/list/active

**GET /user/credentials/list/active** Get a user's list of active S3 security credentials

The request line syntax for this method is as follows.

```
GET /user/credentials/list/active?[userId=xxx&groupId=xxx | canonicalUserId=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"user Query Parameters"** (page 919).

This retrieves the user's active S3 credentials. Inactive (disabled) credentials are not returned.

Specify the user either by using *userId* and *groupId*, or by using *canonicalUserId*.

There is no request payload.

**Note** The HyperStore IAM Service supports a role-based access version of this API method as described in **"RBAC Version of this Method"** (page 905).

#### 12.13.7.0.1. Example Using cURL

The [example](#) below retrieves the active S3 credentials for user "John" in the "QA" group.

```

curl -X GET -k -u sysadmin:public \
'https://localhost:19443/user/credentials/list/active?userId=John&groupId=QA' \
| python -mjson.tool

```

The response payload is a JSON-formatted list of *SecurityInfo* objects, which in this example is as follows (note that this user has only one active credential). For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"SecurityInfo Object"** (page 921).

```

[
  {

```

```

    "accessToken": "009c156c79e64e0e4928",
    "active": true,
    "createDate": 1502279336024,
    "expireDate": null,
    "secretKey": "wVHk2nA0M03RWSMlrFHFAtuhow6S1DKN0gWjPhDG
  }
]

```

### 12.13.7.0.2. Response Format

The response payload is a JSON-formatted list of *SecurityInfo* objects (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
204	No Access Key found
400	Missing Required parameters : {userId, groupId}
400	User/Group does not exist

### 12.13.7.1. RBAC Version of this Method

HyperStore's IAM Service supports an RBAC version of this API method. For an overview of the RBAC feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- Action name: *GetCloudianUserCredentialsListActive*
- Parameters: Same as for *GET /user/credentials/list/active*, except all parameter names start with an upper case letter rather than lower case
- Response body: Same response data as for *GET /user/credentials/list/active* except:
  - The data is formatted in XML rather than JSON
  - The *secretKey* is not included in the response
- Role-based restrictions:
  - HyperStore system admin user can get any user's active credentials list
  - HyperStore group admin user can only get the active credentials list of users within her own group
  - HyperStore regular user can only get own active credentials list
  - IAM user can only use this method if granted *admin:GetCloudianUserCredentialsListActive* permission by an IAM policy, and subject to the same restriction as the parent HyperStore user.

**Note** The "GetCloudianUserCredentialsListActive" action retrieves credentials for Cloudian HyperStore user accounts, not for subsidiary IAM users. For example, if a HyperStore group administrator grants *admin:GetCloudianUserCredentialsListActive* permission to an IAM user, the IAM user will be able to retrieve an active credentials list for **any HyperStore user in the group administrator's group**. And if a HyperStore regular user grants *admin:GetCloudianUserCredentialsListActive* permission to an IAM user, the IAM user will be able to retrieve the **parent HyperStore user's** active credentials list.

- Sample request and response (abridged):

## REQUEST

```
http://localhost:16080/?Action=GetCloudianUserCredentialsListActive&UserId=John&GroupId=QA
```

*<request headers including authorization info>*

## RESPONSE

```
200 OK
```

```
<?xml version="1.0" encoding="utf-8"?>
<GetCloudianUserCredentialsListActiveResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">
  <ResponseMetadata>
    <RequestId>system-generated-request-id</RequestId>
  </ResponseMetadata>
  <ListWrapper>
    <SecurityInfo>
      <accessKey>40602cdfaef3a676594d</accessKey>
      etc...
    ...
  </SecurityInfo>
  <SecurityInfo>
    etc...
  ...
</SecurityInfo>
</ListWrapper>
</GetCloudianUserCredentialsListActiveResponse>
```

### 12.13.8. GET /user/list

#### GET /user/list    Get a list of user profiles

The request line syntax for this method is as follows.

```
GET /user/list?groupId=xxx&userType=xxx&userStatus=xxx [&prefix=xxx] [&limit=xxx]
[&offset=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"user Query Parameters"** (page 919).

There is no request payload.

**Note** The HyperStore IAM Service supports a role-based access version of this API method as described in **"RBAC Version of this Method"** (page 908).

#### 12.13.8.0.1. Example Using cURL

The [example](#) below retrieves a list of all active users in the "QA" group.

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/user/list?groupId=QA&userType=all&userStatus=active' \
| python -mjson.tool
```

The response payload is a JSON-formatted list of *UserInfo* objects, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"UserInfo Object"** (page 922).

```
[
  {
    "active": "true",
    "address1": "",
    "address2": "",
    "canonicalUserId": "fd221552ff4ddc857d7a9ca316bb8344",
    "city": "",
    "country": "",
    "emailAddr": "",
    "fullName": "Glory Bee",
    "groupId": "QA",
    "ldapEnabled": false,
    "phone": "",
    "state": "",
    "userId": "Glory",
    "userType": "User",
    "website": "",
    "zip": ""
  },
  {
    "active": "true",
    "address1": "",
    "address2": "",
    "canonicalUserId": "bd0796cd9746ef9cc4ef656ddaacfac4",
    "city": "",
    "country": "",
    "emailAddr": "",
    "fullName": "John Thompson",
    "groupId": "QA",
    "ldapEnabled": false,
    "phone": "",
    "state": "",
    "userId": "John",
    "userType": "User",
    "website": "",
    "zip": ""
  },
  {
    "active": "true",
    "address1": "",
    "address2": "",
    "canonicalUserId": "4dc9cd1c20c78eb6c84bb825110fddcb",
    "city": "",
    "country": "",
    "emailAddr": "",
    "fullName": "Xiao Li",
```

```

    "groupId": "QA",
    "ldapEnabled": false,
    "phone": "",
    "state": "",
    "userId": "Xiao",
    "userType": "GroupAdmin",
    "website": "",
    "zip": ""
  }
]

```

### 12.13.8.0.2. Response Format

The response payload is a JSON-formatted list of *UserInfo* objects (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameters : {groupId, userType, userStatus}
400	Invalid user type. Valid values {admin, user, all}
400	Invalid user status. Valid values {active, inactive, all}
400	Invalid limit

### 12.13.8.1. RBAC Version of this Method

HyperStore's IAM Service supports an RBAC version of this API method. For an overview of the RBAC feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

- Action name: *GetCloudianUserList*
- Parameters: Same as for *GET /user/list*, except all parameter names start with an upper case letter rather than lower case
- Response body: Same response data as for *GET /user/list* except the data is formatted in XML rather than JSON
- Role-based restrictions:
  - HyperStore system admin user can get any group's user list
  - HyperStore group admin user can only get the user list for her own group
  - HyperStore regular user cannot use this method
  - IAM user can only use this method if granted *admin:GetCloudianUserList* permission by an IAM policy, and subject to the same restriction as the parent HyperStore user.

**Note** The "GetCloudianUserList" action retrieves user profile data for Cloudian HyperStore user accounts, not for subsidiary IAM users. For example, if a HyperStore group administrator grants *admin:GetCloudianUserList* permission to an IAM user, the IAM user will be able to retrieve profile information for **any HyperStore user in the group administrator's group**.

- Sample request and response (abridged):

## REQUEST

```
http://localhost:16080/?Action=GetCloudianUserList&GroupId=QA&UserType=all&UserStatus=active
```

*<request headers including authorization info>*

## RESPONSE

200 OK

```
<?xml version="1.0" encoding="utf-8"?>
<GetCloudianUserListResponse xmlns="https://iam.amazonaws.com/doc/2010-05-08/">
  <ResponseMetadata>
    <RequestId>system-generated-request-id</RequestId>
  </ResponseMetadata>
  <ListWrapper>
    <CassandraUserInfo>
      <active>Active</active>
      etc...
    ...
  </CassandraUserInfo>
  <CassandraUserInfo>
    etc...
  ...
</CassandraUserInfo>
</ListWrapper>
</GetCloudianUserListResponse>
```

### 12.13.9. GET /user/password/verify

#### GET /user/password/verify    Verify a user's CMC password

The request line syntax for this method is as follows.

```
GET /user/password/verify?userId=xxx&groupId=xxx&password=xxx
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"user Query Parameters"** (page 919).

This verifies that the supplied CMC password is the correct password for the user.

There is no request payload.

#### 12.13.9.0.1. Example Using cURL

The [example](#) below verifies the supplied password for a user "John" in the "QA" group.

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/user/password/verify?userId=John&groupId=QA&password=Pla2s3s4!'
```

The response payload is a plain text value "true" or "false", which in this example is as follows.

```
true
```

The "true" response indicates that the supplied password is the correct password for the user.

### 12.13.9.0.2. Response Format

The response payload is a plain text string (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or this method-specific status code:

Status Code	Description
400	Missing Required parameters : {userId, groupId, password}

## 12.13.10. GET /user/ratingPlan

### GET /user/ratingPlan    Get a user's rating plan content

The request line syntax for this method is as follows.

```
GET /user/ratingPlan?userId=xxx&groupId=xxx[&region=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"user Query Parameters"** (page 919).

There is no request payload.

#### 12.13.10.0.1. Example Using cURL

The [example](#) below retrieves the content of the rating plan that is assigned to user "John" in group "QA".

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/user/ratingPlan?userId=John&groupId=QA' \
| python -mjson.tool
```

The response payload is a JSON-formatted *RatingPlan* object, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"RatingPlan Object"** (page 839).

```
{
  "currency": "USD",
  "id": "Gold",
  "mapRules": {
    "BI": {
      "ruleclassType": "BYTES_IN",
      "rules": [
        {
          "first": "0",
          "second": "0"
        }
      ]
    },
    "BO": {
      "ruleclassType": "BYTES_OUT",
      "rules": [
        {
          "first": "0",
          "second": "0"
        }
      ]
    }
  ]
}
```

```

    },
    "HD": {
      "ruleclassType": "HTTP_DELETE",
      "rules": [
        {
          "first": "0",
          "second": "0"
        }
      ]
    },
    "HG": {
      "ruleclassType": "HTTP_GET",
      "rules": [
        {
          "first": "0",
          "second": "0"
        }
      ]
    },
    "HP": {
      "ruleclassType": "HTTP_PUT",
      "rules": [
        {
          "first": "0",
          "second": "0"
        }
      ]
    },
    "SB": {
      "ruleclassType": "STORAGE_BYTE",
      "rules": [
        {
          "first": "100",
          "second": "0.25"
        },
        {
          "first": "0",
          "second": "0.15"
        }
      ]
    }
  },
  "name": "Gold Rating Plan"
}

```

#### 12.13.10.0.2. Response Format

The response payload is a JSON-formatted *RatingPlan* object (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
204	Rating Plan does not exist

Status Code	Description
400	Missing Required parameters : {userId, groupId}
400	Region {region} is not valid

### 12.13.11. GET /user/ratingPlanId

#### GET /user/ratingPlanId Get a user's rating plan ID

The request line syntax for this method is as follows.

```
GET /user/ratingPlanId?userId=xxx&groupId=xxx [&region=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"user Query Parameters"** (page 919).

There is no request payload.

#### 12.13.11.0.1. Example Using cURL

The [example](#) below retrieves the rating plan ID for user "John" in group "QA".

```
curl -X GET -k -u sysadmin:public \
'https://localhost:19443/user/ratingPlanId?userId=John&groupId=QA'
```

The response payload is the rating plan identifier in plain text, which in this example is as follows.

```
Gold
```

#### 12.13.11.0.2. Response Format

The response payload is a plain text string (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
204	Rating Plan does not exist
400	Missing Required parameters : {userId, groupId}
400	Region {region} is not valid

### 12.13.12. POST /user

#### POST /user Change a user's profile

The request line syntax for this method is as follows.

```
POST /user
```

The required request payload is a JSON-formatted *UserInfo* object.

### 12.13.12.0.1. Example Using cURL

The [example](#) below modifies the user profile that was created in the [PUT /user](#) example. Again the *UserInfo* object is specified in a text file named *user\_John.txt* which is then referenced as the data input to the cURL command.

```
curl -X POST -H "Content-Type: application/json" -k -u sysadmin:public \
-d @user_John.txt https://localhost:19443/user
```

Note that in editing the *UserInfo* object in the *user\_John.txt* file before doing the POST operation you could edit any attribute except for the "userId" or "canonicalUserId" attributes. For an example *UserInfo* object see [PUT /user](#).

**Note** You cannot change the userType of the default system administrator account. This is not allowed.

### 12.13.12.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	User does not exist
400	Missing Required parameters : {userId, groupId, userType}
400	Invalid JSON object
400	Invalid User Name

## 12.13.13. POST /user/credentials

### POST /user/credentials Post a user's supplied S3 credential

The request line syntax for this method is as follows.

```
POST /user/credentials?userId=xxx&groupId=xxx&accessKey=xxx&secretKey=xxx
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"user Query Parameters"** (page 919).

There is no request payload.

### 12.13.13.0.1. Example Using cURL

The [example](#) below posts a supplied S3 access key and secret key for user "John" in the "QA" group.

```
curl -X POST -k -u sysadmin:public \
'https://localhost:19443/user/credentials?userId=John&groupId=QA&accessKey=21289&secretKey=o5jqJtq'
```

**Note** To allow the single quote-enclosed 'https:...' segment in the above example to be shown on one line, the access key and secret key values are truncated.

### 12.13.13.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing Required parameters : {userId, groupId, accessKey, secretKey}
400	User does not exist
403	Reached maximum number of credentials allowed
409	Access Key already exists

### 12.13.14. POST /user/credentials/status

**POST /user/credentials/status** Deactivate or reactivate a user's S3 credential

The request line syntax for this method is as follows.

```
POST /user/credentials/status?accessKey=xxx&isActive=xxx
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"user Query Parameters"** (page 919).

There is no request payload.

#### 12.13.14.0.1. Example Using cURL

The [example](#) below deactivates a user's S3 credential. Note that since each S3 access key is unique in the system, you do not need to specify the user to whom the key is assigned.

```
curl -X POST -k -u sysadmin:public \
'https://localhost:19443/user/credentials/status?accessKey=21289bab1738ffdc792a&isActive=false'
```

#### 12.13.14.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required parameters : {accessKey}
400	Invalid Access Key

### 12.13.15. POST /user/password

**POST /user/password** Create or change a user's CMC password

The request line syntax for this method is as follows.

```
POST /user/password?userId=xxx&groupId=xxx&password=xxx
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"user Query Parameters"** (page 919).

Use this method to create or update a user's CMC login password.

If you are updating an existing password for a user, use the "password" parameter to specify the new password, not the existing password.

There is no request payload.

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 [common.csv](#), see **"user\_password\_min\_length"** (page 527) and the subsequent settings.

### 12.13.15.0.1. Example Using cURL

The [example](#) below posts a CMC password for the user "John" in the "QA" group.

```
curl -X POST -k -u sysadmin:public \
'https://localhost:19443/user/password?userId=John&groupId=QA&password=Pla2s3s4!'
```

### 12.13.15.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
204	User does not exist
400	Missing Required parameters : {userId, groupId, password}
400	Exceeded max password length
400	Password strength is too weak.

## 12.13.16. POST /user/ratingPlanId

**POST /user/ratingPlanId** Assign a rating plan to a user

The request line syntax for this method is as follows.

```
POST /user/ratingPlanId?userId=xxx&groupId=xxx&ratingPlanId=xxx [&region=xxx]
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"user Query Parameters"** (page 919).

There is no request payload.

### 12.13.16.0.1. Example Using cURL

The [example](#) below assigns the "Gold" rating plan to user "John" in the "QA" group.

```
curl -X POST -k -u sysadmin:public \
'https://localhost:19443/user/ratingPlanId?userId=John&groupId=QA&ratingPlanId=Gold'
```

### 12.13.16.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing Required parameters : {userId, groupId, ratingPlanId}
400	Region {region} is not valid

## 12.13.17. PUT /user

### PUT /user Create a new user

The request line syntax for this method is as follows.

```
PUT /user
```

The required request payload is a JSON-formatted *UserInfo* object. See example below.

**Note** This method does not create a CMC login password for the new user. After creating a new user with the *PUT /user* method, use the [POST /user/password](#) method to create a CMC password for the user.

### 12.13.17.0.1. Example Using cURL

The [example](#) below creates a new user "John" in the "QA" group. In this example the JSON-formatted *UserInfo* object is specified in a text file named *user\_John.txt* which is then referenced as the data input to the cURL command.

```
curl -X PUT -H "Content-Type: application/json" -k -u sysadmin:public \
-d @user_John.txt https://localhost:19443/user | python -mjson.tool
```

The response payload is a JSON-formatted *UserInfo* object.

Immediately below is the input file for this example (the *UserInfo* object submitted in the request). Below that is the response payload for this example (the *UserInfo* object returned in the response). The difference between the two is that the *UserInfo* object submitted in the request does not include a "canonicalUserId" attribute, whereas the *UserInfo* object returned in the response body does have this attribute. The system has generated

a canonical user ID for the new user. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"UserInfo Object"** (page 922).

Request payload:

```
{
  "active": "true",
  "address1": "",
  "address2": "",
  "city": "",
  "country": "",
  "emailAddr": "",
  "fullName": "John Thompson",
  "groupId": "QA",
  "ldapEnabled": false,
  "phone": "",
  "state": "",
  "userId": "John",
  "userType": "User",
  "website": "",
  "zip": ""
}
```

Response payload:

```
{
  "active": "true",
  "address1": "",
  "address2": "",
  "canonicalUserId": "bd0796cd9746ef9cc4ef656ddaacf4",
  "city": "",
  "country": "",
  "emailAddr": "",
  "fullName": "John Thompson",
  "groupId": "QA",
  "ldapEnabled": false,
  "phone": "",
  "state": "",
  "userId": "John",
  "userType": "User",
  "website": "",
  "zip": ""
}
```

#### 12.13.17.0.2. Response Format

The response payload is a JSON-formatted *UserInfo* object (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing required attributes : {userId, groupId, userType}
400	Invalid JSON object
400	Problem accessing /user

Status Code	Description
400	User Id is not allowed : {userId}
400	Invalid User Name
409	Problem accessing /user

### 12.13.18. PUT /user/credentials

#### PUT /user/credentials Create a new S3 credential for a user

The request line syntax for this method is as follows.

```
PUT /user/credentials?userId=xxx&groupId=xxx
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"user Query Parameters"** (page 919).

There is no request payload.

#### 12.13.18.0.1. Example Using cURL

The [example](#) below creates a new S3 credential for user "John" in the "QA" group.

```
curl -X PUT -k -u sysadmin:public \
'https://localhost:19443/user/credentials?userId=John&groupId=QA' \
| python -mjson.tool
```

The response payload is a JSON-formatted *SecurityInfo* object, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"SecurityInfo Object"** (page 921).

```
{
  "accessKey": "28d945de2a2623fc9483",
  "active": true,
  "createDate": 1502285593100,
  "expireDate": null,
  "secretKey": "j2OrPGHF69hp3YsZHRHOCWdAQDabppsBtD7kttr9"
}
```

#### 12.13.18.0.2. Response Format

The response payload is a JSON-formatted *SecurityInfo* object (see example above). For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Missing Required parameters : {userId, groupId}
400	User does not exist
403	Reached maximum number of credentials allowed

## 12.13.19. user Query Parameters

### *userId*

(Optional for methods that also support a *canonicalUserId* parameter and mandatory for methods that do not; string) Unique identifier of the user. This is the user ID that was supplied by the user (or whoever created the user) at the time of user creation -- not the *canonicalUserId* that is automatically generated by the system when a new user is created.

### *groupId*

(Optional for methods that also support a *canonicalUserId* parameter and mandatory for methods that do not; string) Unique identifier of the group to which the user belongs.

**Note** The group ID for system admins is "0".

### *canonicalUserId*

(Optional, string) System-generated canonical user ID of the user.

If you don't know the user's canonical ID you can retrieve it via the Admin API method *GET /user/list*.

### *accessKey*

(Mandatory, string) With a *POST*, *GET*, or *DELETE /user/credentials* request: The S3 access key.

**Note** An S3 security credential is a key pair consisting of an "access key" (public key) and a "secret key" (private key).

### *secretKey*

(Mandatory, string) With a *POST /user/credentials* request: The S3 secret key.

### *isActive*

(Optional, boolean) With a *POST /user/credentials/status* request: The status to apply to the credentials — *true* for active or *false* for inactive. Defaults to *false* if this query parameter is not supplied in the request.

### *userType*

(Mandatory, string) With a *GET /user/list* request: Retrieve users of this type. Options are:

- *admin* — Administrators. If the group ID is "0" this would be system admins; for any other group this would be group admins.
- *user* — Regular users who lack administrative privileges.
- *all* — Retrieve users of all types.

### *userStatus*

(Mandatory, string) With a *GET /user/list* request: Retrieve users who have this status. Options are:

- *active* — Active users.
- *inactive* — Inactive users. These users have had their status set to inactive via the *POST /user*

method (with the *UserInfo* object attribute "active" set to false). These users' stored S3 objects still exist, but their S3 access credentials have been deactivated.

- *deleted* — Deleted users. These users have been deleted from the S3 service via the *DELETE /user* method. Their S3 access credentials have been deleted, and their S3 buckets and objects have been deleted and are unrecoverable.
- *deleting* — These users are in the process of being deleted from the S3 service via the *DELETE /user* method. The deletion process for these users has not yet completed.
- *all* — Retrieve active users and inactive users. This does **not** retrieve users who have status "deleted" or "deleting". To retrieve deleted or deleting users, specify "deleted" or "deleting" for the *userStatus* request parameter -- not "all".

**Note** Since the CMC does not support retrieving users with status "deleted" or "deleting", the only way to retrieve a list of such users is through the Admin API.

#### *prefix*

(Optional, string) With a *GET /user/list* request: If specified, a user ID prefix to use for filtering. For example, if you specify "prefix=arc" then only users whose user ID starts with "arc" would be retrieved.

Defaults to empty string (meaning that no prefix-based filtering is performed).

#### *limit*

(Optional, integer) With a *GET /user/list* request: For purposes of pagination, the maximum number of users to return in one response. In the response the users are sorted alphanumerically and if more than "limit" users meet the filtering criteria, then the actual number of users returned will be "limit plus 1" (for example, 101 users if the limit is 100). The last, extra returned user — the "plus 1" — is an indicator that there are more users than could be returned in the current response (given the specified "limit" value). That last user's ID can then be used as the "offset" value in a subsequent request that retrieves additional users.

**Note** If the offset user happens to be the last user in the entire set of matching users, the subsequent query using the offset will return no users.

Defaults to 100.

#### *offset*

(Optional, string) With a *GET /user/list* request: The user ID with which to start the response list of users for the current request, sorted alphanumerically. The "offset" parameter can be used for purposes of pagination within a large result set that is being retrieved via multiple sequential requests. See the description of "limit" above for more information.

If "offset" is not specified, the first user in the response list will be the alphanumerically first user from the entire result set.

#### *ratingPlanId*

(Mandatory, string) With a *POST /user/ratingPlanId* request: Unique identifier of the rating plan to assign to the user, for billing purposes.

#### *region*

(Optional, string) If your service deployment has multiple service regions, rating plan assignment is on a per-region basis. With the `POST /user/ratingPlanId` method, use the "region" parameter to indicate the service region in which to apply the specified rating plan. For example, if `userId=Cody&groupId=Engineering&ratingPlanId=Gold&region=East`, then the Gold rating plan will be applied to user Cody's service activity in the East region.

#### *password*

(Mandatory, string) With a `POST /user/password` or `GET /user/password/verify` request: The user's supplied CMC password.

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 [common.csv](#), see "**user\_password\_min\_length**" (page 527) and the subsequent settings.

## 12.13.20. user Objects

This section describes the JSON-formatted data objects that constitute the request or response payloads for the User related Admin API methods.

**Note** For examples of these objects see the API method request and response examples.

- "**SecurityInfo Object**" (page 921)
- "**UserInfo Object**" (page 922)

### 12.13.20.1. SecurityInfo Object

The *SecurityInfo* object consists of the following attributes:

#### *accessKey*

(String) User's access key (public key) for the HyperStore S3 service. Example:

```
"accessKey": "009c156c79e64e0e4928"
```

#### *active*

(Boolean) Whether the credential is active, *true* or *false*. An inactive credential cannot be used to access the HyperStore S3 service. Example:

```
"active": true
```

*createDate*

(String) Creation timestamp for the credential in UTC milliseconds. Example:

```
"createDate": 1502279336024
```

*expireDate*

(String) Credential expiration date. In the current version of HyperStore this attribute's value is always null. Example:

```
"expireDate": null
```

*secretKey*

(String) User's secret key (private key) for the HyperStore S3 service. Example:

```
"secretKey": "wVHk2nA0M03RWSMlrFHFAtuhow6S1DKN0gWjPhDG"
```

### 12.13.20.2. UserInfo Object

The *UserInfo* object consists of the following attributes:

*active*

(Optional, string) Whether the user is active — "true" or "false".

- "true" indicates an active user.
- "false" indicates that the user is not an active user. Non-active users are users who are in one of these statuses:
  - Inactive — These users have had their status set to inactive via the *POST /user* method (with *UserInfo* object attribute "active" set to false). These users' stored S3 objects still exist, but their S3 access credentials have been deactivated and they cannot log into the CMC.
  - Deleted — These users have been deleted from the S3 service via the *DELETE /user* method. Their S3 access credentials have been deleted, and their S3 buckets and objects have been deleted and are unrecoverable. (These users cannot be retrieved through the *GET /user* method — they can only be retrieved through the *GET /user/list* method.)
  - Deleting — These users are in the process of being deleted from the S3 service via the *DELETE /user* method. The deletion process for these users has not yet completed. (These users cannot be retrieved through the *GET /user* method — they can only be retrieved through the *GET /user/list* method.)

If the "active" attribute is unspecified for a *PUT /user* operation, it defaults to "true". If the "active" attribute is unspecified for a *POST /user* operation, the user will retain her existing status.

Example:

```
"active": "true"
```

**Note** The only way to retrieve a list of users who have been deleted or are in the process of being deleted is to specify "deleted" or "deleting" for the *userStatus* request parameter with a *GET /user/list* request. In the response body, the "active" attribute for such users will say "false".

*address1*

(Optional, string) User's street address line 1. Example:

```
"address1": "123 Main St."
```

*address2*

(Optional, string) User's street address line 2. Example:

```
"address2": ""
```

*canonicalUserId*

(Optional, string) Canonical user ID, globally unique within the HyperStore system. This is automatically generated by the system when a new user is created.

- For users created prior to HyperStore 3.0 (before the canonical ID feature existed), the canonicalUserId is the user's <groupId>|<userId> combination.
- For users created in HyperStore 3.0 and later, the canonicalUserId is a system-generated, globally unique printable string. The canonicalUserId is unique per user across the system and over time, even in the case where a user with a specific <groupId>|<userId> combination is deleted from the system and then a new user is subsequently added with the same <groupId>|<userId> combination. The new user will be assigned a different canonicalUserId than the deleted user. This allows past and present users to be uniquely identified for purposes such as usage reporting.
- Client must not supply a canonicalUserId in a *PUT /user* request and does not need to supply one in a *POST /user* request.

Example:

```
"canonicalUserId": "bd0796cd9746ef9cc4ef656ddaacf4"
```

*city*

(Optional, string) User's city. Example:

```
"city": "Portsmouth"
```

*country*

(Optional, string) User's country. Example:

```
"country": "US"
```

*emailAddr*

(Optional, string) User's email address. Example:

```
"emailAddr": "me@mail.com"
```

*fullName*

(Optional, string) User's full name. By default the maximum length is 64 characters. This maximum is configurable by the setting *common.csv: "cloudian\_userid\_length"* (page 514).

Example:

```
"fullName": "John Thompson"
```

*groupId*

(Mandatory, string) Group ID of the group to which the user belongs.

**Note** For all SystemAdmin type users the groupId is "0".

Example:

```
"groupId": "QA"
```

Second example (for a system administrator):

```
"groupId": "0"
```

#### *ldapEnabled*

(Optional, boolean) Whether the CMC authenticates the user by checking an LDAP system, *true* or *false*. Defaults to *false*. If the user is enabled for LDAP, when authenticating the user the CMC uses the LDAP connection information configured for the user's group. For more information see **"LDAP Integration"** (page 131). If the user is not LDAP enabled, the CMC authenticates the user by requiring a password that the CMC maintains.

Example:

```
"ldapEnabled": false
```

#### *phone*

(Optional, string) User's phone number. Example:

```
"phone": "890-123-4567"
```

#### *state*

(Optional, string) User's state. Example:

```
"state": "NH"
```

#### *userId*

(Mandatory, string) User ID.

- Only letters, numbers, dashes, and underscores are allowed.
- By default the maximum length is 64 characters. This maximum is configurable by the setting *common.csv*: **"cloudian\_userid\_length"** (page 514).
- The following IDs are reserved for system use and are not available to individual users: "anonymous", "public", "null", "none", "admin", "0".

Example:

```
"userId": "John"
```

**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
- \* Must start with a letter
- \* Cannot end with an underscore

#### *userType*

(Mandatory, string) User type. One of {"User","GroupAdmin","SystemAdmin"}. Example:

```
"userType": "User"
```

**Note** For all SystemAdmin type users the corresponding *groupId* is "0".

*website*

(Optional, string) User's website URL. Example:

```
"website": "www.me.com"
```

*zip*

(Optional, string) User's postal zip code. Example:

```
"zip": "12345"
```

**Note** In the "address1", "address2", "zip", "email", "website", and "phone" fields, the Admin Service prohibits the use of any of these characters:

```
` | ; & > <
```

**Note** The *PUT /user* method does not create a CMC login password for the new user. After creating a new user with the *PUT /user* method, use the *POST /user/password* method to create a CMC password for the user.

## 12.14. whitelist

The Admin API methods built around the **whitelist** resource are for managing a billing "whitelist" of source IP addresses or subnets that you want to allow to have free S3 traffic with the HyperStore storage service. For background information on the whitelist feature, including how to enable the feature, see **"Creating a Whitelist" for Free Traffic** (page 148). The whitelist feature is disabled by default.

**Note** If you are using load balancers in front of the HyperStore S3 Service, the whitelist 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 **"s3\_proxy\_protocol\_enabled"** (page 530) in [common.csv](#). 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 whitelist feature. You must use PROXY Protocol if you have load balancers in front of the S3 Service and want to use the whitelist feature .

### 12.14.1. GET /whitelist

GET /whitelist    Get whitelist content

The request line syntax for this method is as follows.

```
GET /whitelist?whitelistId=Default-WL
```

For parameter description click on the parameter name or see "**whitelist Query Parameters**" (page 928).

There is no request payload.

**Note** In the current version of HyperStore, only one whitelist is supported and its ID is "Default-WL".

#### 12.14.1.0.1. Example Using cURL

The [example](#) below retrieves the current contents of the whitelist with ID "Default-WL".

```
curl -X GET -k -u sysadmin:public \
https://localhost:19443/whitelist?whitelistId=Default-WL | python -mjson.tool
```

The response payload is a JSON-formatted *Whitelist* object, which in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see "**Whitelist Object**" (page 929).

```
{
  "id": "Default-WL",
  "list": [
    "10.20.2.10",
    "10.20.2.11",
    "10.20.2.12"
  ],
  "name": "Default Whitelist",
  "ratingPlanId": "Whitelist-RP"
}
```

**Note** By default the "Default-WL" whitelist that comes with your HyperStore system is empty. The whitelist in the example above has had some IP addresses added to it.

#### 12.14.1.0.2. Response Format

The response payload is a JSON-formatted *Whitelist* object (see example above). There is no response payload. For response status code this method will return one of the "**Common Response Status Codes**" (page 746) or one of these method-specific status codes:

Status Code	Description
204	Whitelist does not exist
400	Missing required parameter : whitelistId

### 12.14.2. POST /whitelist

**POST /whitelist**    Change whitelist content (by request body object)

The request line syntax for this method is as follows.

POST /whitelist

The required request payload is a JSON-formatted *Whitelist* object. See example below.

**Note** For billing purposes, changes that you make to the composition of the whitelist (by adding or deleting IP addresses or subnets) will take effect starting with the next hourly roll-up of HyperStore usage data.

### 12.14.2.0.1. Example Using cURL

The [example](#) below uploads whitelist content as specified in the request body. In this example the JSON-formatted *Whitelist* object is specified in a text file named *default\_whitelist.txt* which is then referenced as the data input to the cURL command.

```
curl -X POST -H "Content-Type: application/json" -k -u sysadmin:public \
-d @default_whitelist.txt https://localhost:19443/whitelist
```

The *default\_whitelist.txt* file content in this example is as follows. For description of a particular attribute, click on the attribute name; or for the full list of attribute descriptions see **"Whitelist Object"** (page 929).

```
{
  "id": "Default-WL",
  "list": ["10.20.2.10", "10.20.2.11", "10.20.2.12"],
  "name": "Default Whitelist",
  "ratingPlanId": "Whitelist-RP"
}
```

### 12.14.2.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Whitelist does not exist
400	Missing required attributes : {id, name, ratingPlanId}
400	Invalid JSON object
400	Invalid IP Address or IPv4 Subnet CIDR: <value>

## 12.14.3. POST /whitelist/list

### POST /whitelist/list    Change whitelist content (by query parameters)

The request line syntax for this method is as follows.

```
POST /whitelist/list?whitelistId=xxx&list=xxx
```

For description of a particular parameter, click on the parameter name; or for the full list of parameter descriptions see **"whitelist Query Parameters"** (page 928).

There is no request payload.

**Note** For billing purposes, changes that you make to the composition of the whitelist (by adding or deleting IP addresses or subnets) will take effect starting with the next hourly roll-up of HyperStore usage data.

#### 12.14.3.0.1. Example Using cURL

The [example](#) below replaces the existing whitelist contents with a new list of IP addresses.

```
curl -X POST -k -u sysadmin:public \
'https://localhost:19443/whitelist/list?whitelistId=Default-WL&list=10.20.2.10,10.20.2.11,10.20.2.12'
```

#### 12.14.3.0.2. Response Format

There is no response payload. For response status code this method will return one of the **"Common Response Status Codes"** (page 746) or one of these method-specific status codes:

Status Code	Description
400	Whitelist does not exist
400	Missing required attributes : {whitelistId, list}
400	Invalid IP Address or IPv4 Subnet CIDR: {value}

### 12.14.4. whitelist Query Parameters

#### *whitelistId*

(Mandatory, string) Unique identifier of the whitelist. In the current HyperStore release, only one whitelist is supported and its ID is "Default-WL".

#### *list*

(Mandatory, string) With a *POST /whitelist/list* request: A comma-separated list of IP addresses or subnets. This list will **overwrite** the existing whitelist contents, so be sure to specify your full desired list of addresses or subnets (not just new additions). IP addresses can be IPv4 or IPv6 format. For subnets, only IPv4 format is supported in the current HyperStore release. IP addresses are validated for IPv4 or IPv6 syntax, and subnets are validated for CIDR syntax.

### 12.14.5. whitelist Objects

This section describes the JSON-formatted data objects that constitute the request or response payloads for the Whitelist related Admin API methods.

**Note** For examples of these objects see the API method request and response examples.

- **"Whitelist Object"** (page 929)

### 12.14.5.1. Whitelist Object

The *Whitelist* object consists of the following attributes:

*id*

(Mandatory, string) Unique ID of the whitelist.

In the current HyperStore release only one whitelist is supported and its non-editable ID is "Default-WL".

Example:

```
"id": "Default-WL"
```

*list*

(Mandatory, list<string>) JSON array of source IP addresses and/or subnets.

To indicate an empty list, use an empty JSON array.

Example:

```
"list": ["10.20.2.10", "10.20.2.11", "10.20.2.12"]
```

**Note** IP addresses can be IPv4 or IPv6 format. For subnets, only IPv4 format is supported in the current HyperStore release. IP addresses are validated for IPv4 or IPv6 syntax, and subnets are validated for CIDR syntax.

*name*

(Mandatory, string) Display name of the whitelist. The default whitelist object has display name "Default Whitelist". This is editable.

Example:

```
"name": "Default Whitelist"
```

*ratingPlanId*

(Mandatory, string) Unique ID of the rating plan assigned to the whitelist. The default whitelist object is assigned rating plan "Whitelist-RP". This system-provided default whitelist rating plan makes all inbound and outbound traffic free of charge. (By contrast, data storage continues to be priced according to the user's regular assigned rating plan.) You can edit the "ratingPlanId" to associate a different rating plan with the default whitelist. You can also edit the "Whitelist-RP" rating plan, using the usual rating plan APIs.

Example:

```
"ratingPlanId": "Whitelist-RP"
```

This page left intentionally blank

# Chapter 13. S3 API

## 13.1. Introduction

### 13.1.1. HyperStore Support for the AWS S3 API

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

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

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

This documentation also describes ways in which the HyperStore system extends the AWS S3 API, to support additional functionality. Most of these extensions are in the form of additional request headers that add enhanced functionality to standard AWS S3 operations on buckets. These extensions are described within the sections that document HyperStore compliance with standard AWS S3 operations. The extensions are always identified by a sub-heading that says **HyperStore Extension to the S3 API**. (For a summary of the extensions see **"HyperStore Extensions to the S3 API"** (page 937).)

#### 13.1.1.1. Cautions

##### 13.1.1.1.1. Organization of Data Within a Bucket

Some atypical ways of organizing data within a bucket can result in sub-optimal performance for certain S3 operations on that bucket. These include having no folders -- with all objects being stored in the root level of the bucket -- and having massive numbers of folders with very few objects in each folder. For detail see **"Object Metadata Structure in Cassandra"** (page 167).

##### 13.1.1.1.2. Mass Deletes

Do not attempt to delete more than 100,000 objects from a single bucket in less than an hour (using the S3 API method [DELETE Multiple Objects](#)). Doing so will result in `TombstoneOverwhelmingException` errors in the Cassandra logs and an inability to successfully execute an [S3 GET Bucket \(List Objects\) Version 1](#) or [GET Bucket \(List Objects\) Version 2](#) operation on the bucket. If the system is in this error condition, you can trigger a tombstone purge as described in **"Dealing with Excessive Tombstone Build-Up"** (page 476).

#### 13.1.1.2. Using TLS/SSL

By default the HyperStore S3 Service does not accept HTTPS (HTTP over TLS/SSL) connections. It listens only for regular HTTP connections, on port 80. However you can set up HTTPS for the S3 Service as described in **"HTTPS Support (TLS/SSL)"** (page 114) . If you do so then the S3 Service will listen for HTTPS connections on port 443, as well as listening for regular HTTP connections on port 80.

### 13.1.2. S3 Client Application Options

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

- **"Using the CMC as Your S3 Client"** (page 932)
- **"Using Third Party S3 Applications"** (page 933)
- **"Developing Custom S3 Applications for HyperStore"** (page 933)

**IMPORTANT !** Some atypical ways of organizing data within a bucket can result in sub-optimal performance for certain S3 operations on that bucket. For detail see **"Object Metadata Structure in Cassandra"** (page 167).

**Note** When the CMC or other S3 client applications delete S3 objects, the HyperStore system deletes the object metadata immediately but does not delete the actual objects immediately. Instead the objects are [batched for deletion by a cron job](#). When S3 clients overwrite S3 objects, the HyperStore system writes the new version of the object immediately, and updates the object metadata immediately, but does not delete the outdated version of the object immediately. Instead the outdated object versions are batched for deletion by the same cron job. (Note that in a bucket that has [versioning](#) enabled, the old object versions would be retained rather than deleted.)

#### 13.1.2.1. Using the CMC as Your S3 Client

The CMC's **Buckets & Objects** section serves as a graphical S3 client for interacting with the HyperStore object store. With the CMC, users can do the following:

- **"Add a Bucket"** (page 218)
- Set bucket properties
  - **"Set Custom S3 Permissions for a Bucket"** (page 222)
  - **"Set "Canned" S3 Permissions for a Bucket"** (page 224)
  - **"View a Bucket's Storage Policy Information"** (page 226)
  - **"Configure a Bucket Lifecycle Policy for Object Auto-Tiering or Expiration"** (page 227)
  - **"Configure a Bucket as a Static Website"** (page 235)
  - **"Configure Cross-Region Replication for a Bucket"** (page 237)
  - **"Set Versioning for a Bucket"** (page 239)
  - **"Set Logging for a Bucket"** (page 240)
- **"Delete a Bucket"** (page 244)
- **"Create or Delete a "Folder""** (page 245)
- **"Upload an Object"** (page 246)
- Set file properties
  - **"Set Custom S3 Permissions on an Object"** (page 249)
  - **"Set "Canned" S3 Permissions on an Object"** (page 252)

- **"Set Public URL Permissions on an Object"** (page 254)
- **"List or Search for Objects"** (page 257)
- **"Download an Object"** (page 258)
- **"Delete an Object"** (page 261)
- **"Restore an Auto-Tiered Object"** (page 258)

**Note** The CMC system administrator role does not and cannot have its own S3 storage user account. However you can [create a regular user account](#) for yourself, and use that to access the data store. You can also [manage other regular users' data](#) on their behalf, if that capability is enabled in your system by configuration.

### 13.1.2.2. Using Third Party S3 Applications

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

To check to see what is your HyperStore S3 Service endpoint -- the URI to which you will submit S3 requests with your third party application -- go to the CMC's [Cluster Information](#) page.

### 13.1.2.3. Developing Custom S3 Applications for HyperStore

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

Good online resources for S3 application developers include:

- [Amazon Simple Storage Service Developer Guide](#)
- [Amazon S3 resources](#)

#### 13.1.2.3.1. What's Distinct About Developing for the HyperStore S3 Service

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

- HyperStore S3 client applications must use the HyperStore S3 service endpoint rather than the Amazon S3 service endpoint. To check to see what is your HyperStore S3 Service endpoint -- the URI to which you will submit S3 requests with your custom application -- go to the CMC's [Cluster Information](#) page.
- As detailed in the "Supported S3 Operations" section of this documentation, the HyperStore S3 service supports the great majority of but not the entire Amazon S3 API.
- Also as detailed in the "Supported S3 Operations" section of this documentation, the HyperStore S3 service supports a small number of extensions to the Amazon S3 API. (For an overview of the extensions see **"HyperStore Extensions to the S3 API"** (page 937)).

### 13.1.3. Authenticating Requests (AWS Signature Version 4)

HyperStore supports AWS Signature Version 4 for authenticating inbound API requests. The HyperStore implementation of this feature is compliant with Amazon's specification of the feature. For example, you can express

authentication information in the HTTP Authorization header or in query string parameters; and you can compute a checksum of the entire payload prior to transmission, or for large uploads, you can use chunked upload.

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

HyperStore continues to support AWS Signature Version 2 as well.

**Note** For HyperStore, the region name validation aspect of Signature Version 4 is disabled by default. You can enable it with the **"cloudian.s3.authorizationV4.singleregioncheck"** (page 559) and/or **"cloudian.s3.authorizationV4.multiregioncheck"** (page 560) settings in *mts.properties.erb*. Even if you do enable region name validation, the HyperStore S3 Service employs a fall-back device where if the region name specified in the request's authorization header does not match against the local region name, the system checks whether the specified region name matches against the S3 service domain. If both checks fail then the request is rejected. This is to accommodate legacy HyperStore systems where the S3 service endpoint may not necessarily include the region name.

### 13.1.4. Access Control List (ACL) Support

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

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

#### 13.1.4.1. AWS S3 Predefined Groups

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

#### 13.1.4.2. Permission Types

- READ
- WRITE
- READ\_ACP
- WRITE\_ACP
- FULL\_CONTROL

#### 13.1.4.3. Canned ACL

- private
- public-read
- public-read-write
- authenticated-read
- bucket-owner-read

- bucket-owner-full-control
- log-delivery-write

#### *HyperStore Extension to the S3 API*

The HyperStore system supports these additional canned ACLs:

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

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

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

### 13.1.5. S3 Common Request and Response Headers

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

- Authorization
- Content-Length
- Content-Type
- Content-MD5
- Date
- Expect
- Host
- x-amz-content-sha256
- x-amz-date

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

- Content-Length
- Content-Type
- Connection

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

### 13.1.6. S3 Error Responses

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

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

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

### 13.1.7. HyperStore Extensions to the S3 API

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

Extension	Purpose	Detail
<i>x-gmt-policyid</i> as optional request header for "PUT Bucket" and response header for "GET Bucket Object (List Objects) version 1", "GET Bucket Object (List Objects) version 2", and "HEAD Bucket"	Specify the HyperStore storage policy to use for a new bucket	<ul style="list-style-type: none"> <li>• <b>"CreateBucket"</b> (page 941)</li> <li>• <b>"ListObjects"</b> (page 960)</li> <li>• <b>"ListObjectsV2"</b> (page 962)</li> <li>• <b>"HeadBucket"</b> (page 956)</li> <li>• <b>"Storage Policies Feature Overview"</b> (page 76)</li> </ul>
<i>x-gmt-tieringinfo</i> and <i>x-gmt-compare</i> and <i>x-gmt-post-tier-copy</i> as optional request headers for "PUT Bucket lifecycle" and response headers for "GET Bucket lifecycle"	Set up auto-tiering for a bucket	<ul style="list-style-type: none"> <li>• <b>"PutBucketLifecycle"</b> (page 969)</li> <li>• <b>"GetBucketLifecycle"</b> (page 949)</li> <li>• <b>"Auto-Tiering Feature Overview"</b> (page 176)</li> </ul>
<i>x-gmt-error-code</i> and <i>x-gmt-message</i> as supported response headers for "GET Object" and "Head Object"	Provide additional information about HTTP 4xx errors	<ul style="list-style-type: none"> <li>• <b>"GetObject"</b> (page 952)</li> <li>• <b>"HeadObject"</b> (page 957)</li> </ul>

## 13.2. Supported S3 Operations

### 13.2.1. AbortMultipartUpload

This operation aborts a multipart upload.

Along with the [common headers](#), HyperStore supports the operation-specific parameters listed below.

For operation details and examples see the AWS documentation: [AbortMultipartUpload](#)

*Former operation name: Abort Multipart Upload*

#### 13.2.1.1. Query Parameters

- uploadId

### 13.2.2. CompleteMultipartUpload

Completes a multipart upload by assembling previously uploaded parts.

Along with the [common headers](#), HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: [CompleteMultipartUpload](#)

*Former operation name: Complete Multipart Upload*

### 13.2.2.1. Request Elements

- CompleteMultipartUpload
- Part
- PartNumber
- ETag

### 13.2.2.2. Response Headers

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

### 13.2.2.3. Response Elements

- CompleteMultipartUploadResult
- Location
- Bucket
- Key
- ETag

## 13.2.3. CopyObject

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

Along with the [common headers](#), HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: [CopyObject](#)

*Former operation name: PUT Object - Copy*

### 13.2.3.1. Request Headers

- x-amz-copy-source
- x-amz-metadata-directive
- x-amz-copy-source-if-match
- x-amz-copy-source-if-none-match
- x-amz-copy-source-if-unmodified-since
- x-amz-copy-source-if-modified-since
- x-amz-storage-class

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

- x-amz-tagging-directive
- x-amz-tagging
- x-amz-website-redirect-location
- x-amz-object-lock-mode

**Note** For more information on HyperStore's support for the S3 "Object Lock" feature, see **"WORM (Object Lock)"** (page 121).

- x-amz-object-lock-retain-until-date
- x-amz-object-lock-legal-hold
- x-amz-server-side-encryption

**Note** For information about HyperStore's support of the *x-amz-server-side-encryption* and *x-amz-server-side-encryption-customer-\** request headers, and set-up steps that you must perform in order to use HyperStore's server-side encryption features, see **"Server-Side Encryption"** (page 105).

- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key
- x-amz-server-side-encryption-customer-key-MD5
- x-amz-copy-source-server-side-encryption-customer-algorithm
- x-amz-copy-source-server-side-encryption-customer-key
- x-amz-copy-source-server-side-encryption-customer-key-MD5
- x-amz-acl
- x-amz-grant-read
- x-amz-grant-write
- x-amz-grant-read-acp
- x-amz-grant-write-acp
- x-amz-grant-full-control

#### 13.2.3.2. Response Headers

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

#### 13.2.3.3. Response Elements

- CopyObjectResult
- ETag
- LastModified

### 13.2.4. CreateBucket

Creates a new bucket.

Along with the [common headers](#), HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: [CreateBucket](#)

*Former operation name: PUT Bucket*

**IMPORTANT !** Some atypical ways of organizing data within a bucket can result in sub-optimal performance for certain S3 operations on that bucket. For detail see **"Object Metadata Structure in Cassandra"** (page 167).

**Note** By default each user is allowed a maximum of 100 buckets. You can change this setting in the CMC's [Configuration Settings](#) page.

#### 13.2.4.1. Request Headers

- x-amz-acl
- x-amz-object-lock-enabled

**Note** For more information on HyperStore's support for the S3 "Object Lock" feature, see **"WORM (Object Lock)"** (page 121).

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

*HyperStore Extension to the S3 API*

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

Name	Description	Required
x-gmt-policyid	<p>This header specifies the unique ID of the storage policy to assign to the newly created bucket. The storage policy determines how data in the bucket will be distributed and protected through either replication or erasure coding. System administrators can create multiple storage policies through the CMC and the system automatically assigns each a unique policy ID that becomes part of the policy definition. (To obtain a list of storage policies for your system and their policy IDs, you can use the Admin API's <a href="#">GET /bppolicy/listpolicy</a> method).</p> <p>With the "x-gmt-policyid" request header for "PUT Bucket", you specify the ID of the desired storage policy when you create a new bucket. Note how-</p>	No

Name	Description	Required
	<p>ever that some policies may not be available to all user groups — a policy's availability is specified by system administrators at the time of policy creation, and this information becomes part of the policy definition. When you specify an "x-gmt-policyid" value with a "PUT Bucket" request, the policy ID must be for a policy that is available to the group to which the bucket owner belongs.</p> <p>Also the policy ID must be for a storage policy from the service region that is specified in the "PUT Bucket" request's LocationConstraint element.</p> <p>If the "PUT Bucket" request does not include the "x-gmt-policyid" request header, then the system will automatically assign the system default storage policy to the bucket during bucket creation.</p> <div> <p><b>Note</b> 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.</p> </div> <div> <p><b>Note</b> A 403 error response is returned if you specify a policy ID that does not exist, has been disabled, is not available to the region in which the bucket is being created, or is not available to the group to which the bucket owner belongs. A 403 is also returned if you do not specify an "x-gmt-policyid" header and the system does not yet have an established default storage policy.</p> </div> <p>Example header:</p> <pre>x-gmt-policyid: 1bc90238f9f11cb32f5e4e901675d50b</pre> <p>For more information on storage policies, see <b>"Storage Policies Feature Overview"</b> (page 76).</p>	

#### 13.2.4.2. Request Elements

- CreateBucketConfiguration
- LocationConstraint

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

#### 13.2.5. CreateMultipartUpload

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

Along with the [common headers](#), HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: [CreateMultipartUpload](#)

*Former operation name: Initiate Multipart Upload*

### 13.2.5.1. Request Headers

- Cache-Control
- Content-Disposition
- Content-Encoding
- Content-Type
- Expires
- x-amz-meta-
- x-amz-storage-class

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

- x-amz-website-redirect-location
- x-amz-object-lock-mode

**Note** For more information on HyperStore's support for the S3 "Object Lock" feature, see **"WORM (Object Lock)"** (page 121).

- x-amz-object-lock-retain-until-date
- x-amz-object-lock-legal-hold
- x-amz-acl
- x-amz-grant-read
- x-amz-grant-write
- x-amz-grant-read-acp
- x-amz-grant-write-acp
- x-amz-grant-full-control
- x-amz-server-side-encryption

**Note** For information about HyperStore's support of the *x-amz-server-side-encryption* and *x-amz-server-side-encryption-customer-\** request headers, and set-up steps that you must perform in order to use HyperStore's server-side encryption features, see **"Server-Side Encryption"** (page 105).

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

### 13.2.5.2. Response Headers

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

### 13.2.5.3. Response Elements

- InitiateMultipartUploadResult
- Bucket
- Key
- UploadId

## 13.2.6. DeleteBucket

Deletes the bucket.

For this operation HyperStore supports the [S3 common headers](#).

For operation details and examples see the AWS documentation: [DeleteBucket](#)

*Former operation name: DELETE Bucket*

## 13.2.7. DeleteBucketCors

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

For this operation HyperStore supports the [S3 common headers](#).

For operation details and examples see the AWS documentation: [DeleteBucketCors](#)

*Former operation name: DELETE Bucket cors*

## 13.2.8. DeleteBucketEncryption

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

For this operation HyperStore supports the [S3 common headers](#).

For operation details and examples see the AWS documentation: [DeleteBucketEncryption](#)

*Former operation name: DELETE Bucket encryption*

## 13.2.9. DeleteBucketLifecycle

Deletes the lifecycle configuration from the specified bucket.

For this operation HyperStore supports the [S3 common headers](#).

For operation details and examples see the AWS documentation: [DeleteBucketLifecycle](#)

*Former operation name: DELETE Bucket lifecycle*

### 13.2.10. DeleteBucketPolicy

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

For this operation HyperStore supports the [S3 common headers](#).

For operation details and examples see the AWS documentation: [DeleteBucketPolicy](#)

*Former operation name: DELETE Bucket policy*

### 13.2.11. DeleteBucketReplication

Deletes the replication configuration from the bucket.

For this operation HyperStore supports the [S3 common headers](#).

For operation details and examples see the AWS documentation: [DeleteBucketReplication](#)

*Former operation name: DELETE Bucket replication*

### 13.2.12. DeleteBucketTagging

Deletes the tags from the bucket.

For this operation HyperStore supports the [S3 common headers](#).

For operation details and examples see the AWS documentation: [DeleteBucketTagging](#)

*Former operation name: DELETE Bucket tagging*

### 13.2.13. DeleteBucketWebsite

This operation removes the website configuration for a bucket.

For this operation HyperStore supports the [S3 common headers](#).

For operation details and examples see the AWS documentation: [DeleteBucketWebsite](#)

*Former operation name: DELETE Bucket website*

### 13.2.14. DeleteObject

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

Along with the [common headers](#), HyperStore supports the operation-specific headers listed below.

For operation details and examples see the AWS documentation: [DeleteObject](#)

*Former operation name: DELETE Object*

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

data deletes runs hourly on each node. The frequency with which the batch processing job runs is configurable by the **"cloudian.delete.queue.poll.interval"** (page 566) property in [mts.properties.erb](#).

**IMPORTANT !** Do not attempt to delete more than 100,000 objects from a single bucket in less than an hour. Doing so will result in `TombstoneOverwhelmingException` errors in the Cassandra logs and an inability to successfully execute a **"ListObjects"** (page 960) operation on the bucket. If the system is in this error condition, you can trigger a tombstone purge as described in **"Dealing with Excessive Tombstone Build-Up"** (page 476).

#### 13.2.14.1. Request Headers

- x-amz-bypass-governance-retention

**Note** For more information on HyperStore's support for the S3 "Object Lock" feature, see **"WORM (Object Lock)"** (page 121).

#### 13.2.14.2. Response Headers

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

#### 13.2.15. DeleteObjects

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

Along with the [common headers](#), HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: [DeleteObjects](#)

*Former operation name: Delete Multiple Objects*

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

**Note** Successful completion of a *DeleteObjects* request results in the system marking the objects as having been deleted. However the actual deletion of object data from disk will not occur until the next automatic running of the object deletion batch processing job. By default this batch processing of object data deletes runs hourly on each node. The frequency with which the batch processing job runs is configurable by the **"cloudian.delete.queue.poll.interval"** (page 566) property in [mts.properties.erb](#).

**IMPORTANT !** Do not attempt to delete more than 100,000 objects from a single bucket in less than an hour. Doing so will result in `TombstoneOverwhelmingException` errors in the Cassandra logs and an inability to successfully execute an S3 **"ListObjects"** (page 960) or **"ListObjectsV2"** (page 962)

operation on the bucket. If the system is in this error condition, you can trigger a tombstone purge as described in **"Dealing with Excessive Tombstone Build-Up"** (page 476).

#### 13.2.15.1. Request Headers

- Content-MD5
- Content-Length
- x-amz-bypass-governance-retention

**Note** For more information on HyperStore's support for the S3 "Object Lock" feature, see **"WORM (Object Lock)"** (page 121).

#### 13.2.15.2. Request Elements

- Delete
- Quiet
- Object
- Key
- VersionId

#### 13.2.15.3. Response Elements

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

### 13.2.16. DeleteObjectTagging

Removes the entire tag set from the specified object.

For this operation HyperStore supports the [S3 common headers](#).

For operation details and examples see the AWS documentation: [DeleteObjectTagging](#)

*Former operation name: DELETE Object tagging*

### 13.2.17. GetBucketAcl

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

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [GetBucketAcl](#)

*Former operation name: GET Bucket acl*

#### 13.2.17.1. Response Elements

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

### 13.2.18. GetBucketCors

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

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [GetBucketCors](#)

*Former operation name: GET Bucket cors*

#### 13.2.18.1. Response Elements

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

### 13.2.19. GetBucketEncryption

Returns the default encryption configuration for the bucket.

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [GetBucketEncryption](#)

*Former operation name: GET Bucket encryption*

### 13.2.19.1. Response Elements

- ApplyServerSideEncryptionByDefault
- Rule
- ServerSideEncryptionConfiguration
- SSEAlgorithm

## 13.2.20. GetBucketLifecycle

Returns the lifecycle configuration information set on the bucket.

Along with the [common headers](#), HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: [GetBucketLifecycle](#)

*Former operation name: GET Bucket lifecycle*

### 13.2.20.1. Response Headers

*HyperStore Extension to the S3 API*

The HyperStore system supports the following Response Headers as extensions to the "GET Bucket lifecycle" operation:

Name	Description	Required
x-gmt-tieringinfo	See <b>"PutBucketLifecycle"</b> (page 969).	No
x-gmt-compare	See <b>"PutBucketLifecycle"</b> (page 969).	No
x-gmt-post-tier-copy	See <b>"PutBucketLifecycle"</b> (page 969).	No

### 13.2.20.2. Response Elements

- GetBucketLifecycleOutput
- Rule

## 13.2.21. GetBucketLocation

Returns the Region the bucket resides in.

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [GetBucketLocation](#)

*Former operation name: GET Bucket location*

### 13.2.21.1. Response Elements

- LocationConstraint

### 13.2.21.1.1. NOTE: GetBucketLocation Response for Buckets in the Default Service Region

The GetBucketLocation operation behaves as follows:

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

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

## 13.2.22. GetBucketLogging

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

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [GetBucketLogging](#)

*Former operation name: GET Bucket logging*

### 13.2.22.1. Response Elements

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

## 13.2.23. GetBucketNotificationConfiguration

Returns the notification configuration of a bucket.

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [GetBucketNotificationConfiguration](#)

*Former operation name: GET Bucket notification*

### 13.2.23.1. Response Elements

- NotificationConfiguration
- QueueConfiguration
- TopicConfiguration

### 13.2.24. GetBucketPolicy

Returns the policy of a specified bucket.

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [GetBucketPolicy](#)

*Former operation name: GET Bucket policy*

#### 13.2.24.1. Response Elements

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

### 13.2.25. GetBucketReplication

Returns the replication configuration of a bucket.

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [GetBucketReplication](#)

*Former operation name: GET Bucket replication*

#### 13.2.25.1. Response Elements

- ReplicationConfiguration
- Role
- Rule

### 13.2.26. GetBucketTagging

Returns the tag set associated with the bucket.

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [GetBucketTagging](#)

*Former operation name: GET Bucket tagging*

**Note** 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](#).

#### 13.2.26.1. Response Elements

- Tagging
- TagSet
- Tag
- Key
- Value

### 13.2.27. GetBucketVersioning

Returns the versioning state of a bucket.

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [GetBucketVersioning](#)

*Former operation name: GET Bucket versioning*

#### 13.2.27.1. Response Elements

- Status
- VersioningConfiguration

### 13.2.28. GetBucketWebsite

Returns the website configuration for a bucket.

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [GetBucketWebsite](#)

*Former operation name: GET Bucket website*

#### 13.2.28.1. Response Elements

- ErrorDocument
- IndexDocument
- WebsiteConfiguration

### 13.2.29. GetObject

Retrieves objects from the S3 storage system.

Along with the [common headers](#), HyperStore supports the operation-specific parameters, headers, and elements listed below.

For operation details and examples see the AWS documentation: [GetObject](#)

*Former operation name: GET Object*

#### 13.2.29.1. Query Parameters

- partNumber
- versionId
- response-content-type
- response-content-language
- response-expires
- response-cache-control

- response-content-disposition
- response-content-encoding

### 13.2.29.2. Request Headers

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

**Note** For information about HyperStore's support of the *x-amz-server-side-encryption-customer-\** request headers, and set-up steps that you must perform in order to use HyperStore's server-side encryption features, see **"Server-Side Encryption"** (page 105).

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

### 13.2.29.3. Response Headers

- x-amz-delete-marker
- x-amz-expiration
- x-amz-meta-\*
- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key-MD5
- x-amz-restore
- x-amz-tagging-count
- x-amz-version-id
- x-amz-website-redirect-location
- x-amz-object-lock-mode
- x-amz-object-lock-retain-until-date
- x-amz-object-lock-legal-hold

#### *HyperStore Extension to the S3 API*

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

Name	Description
x-gmt-error-code	In the event of an HTTP 4xx response, these two response headers provide additional information about the nature of the error. The <i>x-gmt-error-code</i> header values will be from among the list in <b>"S3 Error Responses"</b> (page 936).
x-gmt-message	

### 13.2.30. GetObjectAcl

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

Along with the [common headers](#), HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: [GetObjectAcl](#)

*Former operation name: GET Object acl*

#### 13.2.30.1. Response Elements

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

### 13.2.31. GetObjectLegalHold

Gets an object's current Legal Hold status.

Along with the [common headers](#), HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: [GetObjectLegalHold](#)

*Former operation name: GET Object legal hold*

#### 13.2.31.1. Query Parameters

- versionId

#### 13.2.31.2. Response Elements

- LegalHold
- Status

### 13.2.32. GetObjectLockConfiguration

Gets the Object Lock configuration for a bucket.

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [GetObjectLockConfiguration](#)

*Former operation name: GET Bucket object lock configuration*

#### 13.2.32.1. Response Elements

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

### 13.2.33. GetObjectRetention

Retrieves an object's retention settings.

Along with the [common headers](#), HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: [GetObjectRetention](#)

*Former operation name: GET Object retention*

#### 13.2.33.1. Query Parameters

- versionId

#### 13.2.33.2. Response Elements

- Retention
- Mode
- RetainUntilDate

### 13.2.34. GetObjectTagging

Returns the tag-set of an object.

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [GetObjectTagging](#)

*Former operation name: GET Object tagging*

### 13.2.34.1. Response Elements

- Tagging
- TagSet
- Tag
- Key
- Value

### 13.2.35. GetObjectTorrent

Return torrent files from a bucket.

Along with the [common headers](#), HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: [GetObjectTorrent](#)

*Former operation name: GET Object torrent*

#### 13.2.35.1. Implementation Notes

- HyperStore does not provide a BitTorrent "tracker". You must either provide your own tracker or use one of the many publicly available trackers. **You must edit the "cloudian.s3.torrent.tracker" (page 577) property in [mts.properties](#)** to specify the URL of the tracker that you are using.
- HyperStore implements BitTorrent HTTP seeding for in accordance with the BEP19 specification ([http://www.bittorrent.org/beps/bep\\_0019.html](http://www.bittorrent.org/beps/bep_0019.html)). Therefore torrent files returned by HyperStore in response to *GET Object torrent* requests will include a "url-list" key and the value of that key will be the URL of the object in HyperStore.
- HyperStore objects that have been auto-tiered to a destination S3 system cannot be retrieved via BitTorrent, unless the objects are first restored to local HyperStore storage (via the S3 **"RestoreObject" (page 988)** method). Restored objects can be retrieved from HyperStore via BitTorrent.
- Like with Amazon S3, with HyperStore only publicly readable objects are eligible for BitTorrent retrieval. And like with Amazon S3, the following types of objects are **not** retrievable via BitTorrent:
  - Objects larger than 5GB
  - Non-current versions of versioned objects
  - Objects encrypted via SSE-C (SSE with Customer-managed key; by contrast, BitTorrent retrieval is supported for objects encrypted with regular SSE)

### 13.2.36. HeadBucket

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

Along with the [common headers](#), HyperStore supports the operation-specific headers listed below.

For operation details and examples see the AWS documentation: [HeadBucket](#)

*Former operation name: HEAD Bucket*

### 13.2.36.1. Response Headers

- x-amz-bucket-region

#### *HyperStore Extension to the S3 API*

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

Parameter	Description
x-gmt-policyid	This header specifies the unique ID of the storage policy assigned to the bucket. For more information see <b>"CreateBucket"</b> (page 941).

### 13.2.37. HeadObject

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

Along with the [common headers](#), HyperStore supports the operation-specific parameters and headers listed below.

For operation details and examples see the AWS documentation: [HeadObject](#)

*Former operation name: HEAD Object*

#### 13.2.37.1. Query Parameters

- partNumber
- versionId

#### 13.2.37.2. Request Headers

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

#### 13.2.37.3. Response Headers

- x-amz-expiration
- x-amz-meta-\*
- x-amz-restore
- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm

- x-amz-server-side-encryption-customer-key-MD5
- x-amz-tagging-count
- x-amz-version-id
- x-amz-object-lock-mode
- x-amz-object-lock-retain-until-date
- x-amz-object-lock-legal-hold

#### HyperStore Extension to the S3 API

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

Name	Description
x-gmt-error-code	In the event of an HTTP 4xx response, these two response headers provide additional information about the nature of the error. The <i>x-gmt-error-code</i> header values will be from among the list in <b>"S3 Error Responses"</b> (page 936).
x-gmt-message	

### 13.2.38. ListBuckets

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

Along with the [common headers](#), HyperStore supports the operation-specific parameter listed below.

For operation details and examples see the AWS documentation: [ListBuckets](#)

*Former operation name: GET Service*

#### HyperStore Extension to the S3 API

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

**Note** Support for this extension is disabled by default. To enable support for this extension, in **"mts.-properties.erb"** (page 553) set *cloudian.s3.enablesharedbucket* to true, then do a Puppet push and then restart the S3 Service.

Name	Description	Required
shared	<p>If the <i>shared</i> parameter is included in the request, the ListBuckets operation returns a list of buckets that other users have shared with the requesting user. This will be buckets that have been shared specifically with the requesting user, plus buckets that have been shared with the group to which the requesting user belongs, plus buckets that have been shared with everyone.</p> <p>Example:</p> <pre>GET /?shared HTTP/1.1. Host: s3-region1.enterprise4.mobi-cloud.com. Accept-Encoding: identity. Date: Fri, 05 Apr 2019 15:34:01 GMT.</pre>	No

Name	Description	Required
	<pre> Content-Length: 0. Authorization: AWS akey2:jTcwdlTa+5sZftVHGtEEyweojdk=. User-Agent: Boto/2.42.0 Python/2.7.5 Linux/3.10.0-693.el7.x86_64.  HTTP/1.1 200 OK. Date: Fri, 05 Apr 2019 15:34:01 GMT. x-amz-request-id: 1721b414-267b-1341-93e6-d4ae52ce5402. Content-Type: application/xml; charset=UTF-8. Content-Length: 432. Server: CloudianS3.  &lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/"&gt; &lt;Owner&gt;&lt;ID&gt;8ce1c49e532edc91b0a43e0c7e7d5975&lt;/ID&gt; &lt;DisplayName&gt;robot1&lt;/DisplayName&gt;&lt;/Owner&gt; &lt;Buckets&gt;&lt;Bucket&gt;&lt;Name&gt;sharedbucket1&lt;/Name&gt; &lt;CreationDate&gt;2019-04-05T15:30:03.897Z&lt;/CreationDate&gt;&lt;/Bucket&gt; &lt;Bucket&gt;&lt;Name&gt;sharedbucket2&lt;/Name&gt; &lt;CreationDate&gt;2019-04-05T15:27:26.300Z&lt;/CreationDate&gt; &lt;/Bucket&gt;&lt;/Buckets&gt;&lt;/ListAllMyBucketsResult&gt; </pre> <div> <p><b>Note</b> When the 'shared' parameter is used, the <i>ListBuckets</i> call returns only buckets that have been shared with the requesting user -- <b>not buckets owned by the requesting user</b>. So to retrieve all buckets that a user has access to, an S3 client application must submit two <i>ListBuckets</i> calls -- one without the 'shared' parameter (to retrieve the user's own buckets) and one with the 'shared' parameter (to retrieve buckets that have been shared with the user).</p> <p><b>Note</b> When the 'shared' parameter is used, in the <i>ListBuckets</i> response body the "Owner" is the requesting user, not the actual owner(s) of the shared bucket(s).</p> </div>	

### 13.2.39. ListMultipartUploads

This operation lists in-progress multipart uploads.

Along with the [common headers](#), HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: [ListMultipartUploads](#)

*Former operation name: List Multipart Uploads*

### 13.2.39.1. Query Parameters

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

### 13.2.39.2. Response Elements

- ListMultipartUploadsResult
- Bucket
- KeyMarker
- UploadIdMarker
- NextKeyMarker
- NextUploadIdMarker
- Encoding-Type
- MaxUploads
- IsTruncated
- Upload
- Key
- UploadId
- Initiator
- ID
- DisplayName
- Owner
- StorageClass
- Initiated
- ListMultipartUploadsResult.Prefix
- Delimiter
- CommonPrefixes
- CommonPrefixes.Prefix

## 13.2.40. ListObjects

Returns some or all of the objects in a bucket.

Along with the [common headers](#), HyperStore supports the operation-specific parameters, headers, and elements listed below.

For operation details and examples see the AWS documentation: [ListObjects](#)

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

**Note** HyperStore also supports the newer version of this API operation, [ListObjectsV2](#).

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

### 13.2.40.1. Query Parameters

- delimiter

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

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

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

### 13.2.40.2. Response Headers

#### *HyperStore Extension to the S3 API*

The HyperStore system supports the following Response Header as an extension to the "GET Bucket (List Objects) Version 1" operation:

Name	Description	Required
x-gmt-policyid	This header specifies the unique ID of the storage policy assigned to the bucket. For more information see <b>"CreateBucket"</b> (page 941).	No

### 13.2.40.3. Response Elements

- Contents
- CommonPrefixes
- Delimiter
- DisplayName
- Encoding-Type
- ETag
- ID
- IsTruncated
- Key
- LastModified

- ListBucketResult
- Marker
- MaxKeys
- Name
- NextMarker
- Owner
- Prefix
- Size
- StorageClass (values STANDARD and GLACIER only)

### 13.2.41. ListObjectsV2

Returns some or all of the objects in a bucket.

Along with the [common headers](#), HyperStore supports the operation-specific parameters, headers, and elements listed below.

For operation details and examples see the AWS documentation: [ListObjectsV2](#)

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

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

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

#### 13.2.41.1. Query Parameters

- continuation-token
- delimiter

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

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

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

### 13.2.41.2. Response Headers

*HyperStore Extension to the S3 API*

The HyperStore system supports the following Response Header as an extension to the "GET Bucket (List Objects) Version 2" operation:

Name	Description	Required
x-gmt-policyid	This header specifies the unique ID of the storage policy assigned to the bucket. For more information see <b>"CreateBucket"</b> (page 941).	No

### 13.2.41.3. Response Elements

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

### 13.2.42. ListObjectVersions

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

Along with the [common headers](#), HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: [ListObjectVersions](#)

*Former operation name: GET Bucket Object versions*

### 13.2.42.1. Query Parameters

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

### 13.2.42.2. Response Elements

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

### 13.2.43. ListParts

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

Along with the [common headers](#), HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: [ListParts](#)

*Former operation name: List Parts*

### 13.2.43.1. Query Parameters

- encoding-type
- uploadId
- max-parts
- part-number-marker

### 13.2.43.2. Response Elements

- x-amz-abort-date
- x-amz-abort-rule-id
- ListPartsResult
- Bucket
- Encoding-Type
- Key
- UploadId
- Initiator
- ID
- DisplayName
- Owner
- StorageClass
- PartNumberMarker
- NextPartNumberMarker
- MaxParts
- IsTruncated
- Part
- PartNumber
- LastModified
- ETag
- Size

## 13.2.44. OPTIONS Object

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

Along with the [common headers](#), HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: [OPTIONS Object](#)

*Former operation name: OPTIONS Object (no change)*

### 13.2.44.1. Request Headers

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

### 13.2.44.2. Response Headers

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

## 13.2.45. POST Object

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

Along with the [common headers](#), HyperStore supports the operation-specific form fields listed below.

For operation details and examples see the AWS documentation: [POST Object](#)

*Former operation name: POST Object (no change)*

### 13.2.45.1. Form Fields

- AWSAccessKeyId
- acl
- Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires
- file
- key
- policy
- success\_action\_redirect, redirect
- success\_action\_status
- tagging
- x-amz-storage-class

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

- x-amz-meta-\*

**Note** The metadata values must be UTF-8 and must not contain control characters less than 0x20 except for \r, \n, and \t. Also, normal XML escaping is required where appropriate.

- x-amz-website-redirect-location
- x-amz-object-lock-mode

**Note** For more information on HyperStore's support for the S3 "Object Lock" feature, see **"WORM (Object Lock)"** (page 121).

- x-amz-object-lock-retain-until-date
- x-amz-object-lock-legal-hold
- x-amz-server-side-encryption

**Note** For information about HyperStore's support of the *x-amz-server-side-encryption* and *x-amz-server-side-encryption-customer-\** request headers, and set-up steps that you must perform in order to use HyperStore's server-side encryption features, see **"Server-Side Encryption"** (page 105).

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

### 13.2.45.2. Response Headers

- x-amz-expiration
- success\_action\_redirect, redirect
- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm
- x-amz-server-side-encryption-customer-key-MD5
- x-amz-version-id

### 13.2.45.3. Response Elements

- Bucket
- ETag
- Key
- Location

## 13.2.46. PutBucketAcl

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

Along with the [common headers](#), HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: [PutBucketAcl](#)

*Former operation name: PUT Bucket acl*

### 13.2.46.1. Request Headers

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

### 13.2.46.2. Request Elements

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

## 13.2.47. PutBucketCors

Sets the *cors* configuration for your bucket.

Along with the [common headers](#), HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: [PutBucketCors](#)

*Former operation name: PUT Bucket cors*

### 13.2.47.1. Request Headers

- Content-MD5

### 13.2.47.2. Request Elements

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

### 13.2.48. PutBucketEncryption

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

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [PutBucketEncryption](#)

*Former operation name: PUT Bucket encryption*

**Note** For information about HyperStore's support for server-side encryption -- including the interaction of object level, bucket level, and storage policy level encryption settings -- see "**Server-Side Encryption**" (page 105).

**Note** In the current HyperStore release, **only the bucket owner is allowed to perform operations relating to bucket encryption**. HyperStore does not currently support the use of bucket policies to extend bucket encryption permissions to users other than the bucket owner. Specifically, with regard to "**PutBucketPolicy**" (page 977), HyperStore does not currently support the "s3:PutEncryptionConfiguration" or "s3:GetEncryptionConfiguration" actions.

#### 13.2.48.1. Request Elements

- ApplyServerSideEncryptionByDefault
- KMSEMasterKeyID
- Rule
- ServerSideEncryptionConfiguration
- SSEAlgorithm

### 13.2.49. PutBucketLifecycle

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

Along with the [common headers](#), HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: [PutBucketLifecycle](#)

*Former operation name: PUT Bucket lifecycle*

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

**Note** Do not set a bucket lifecycle rule and a [cross-region replication](#) configuration on the same bucket.

## 13.2.49.1. Request Headers

- Content-MD5

*HyperStore Extension to the S3 API*

The HyperStore system supports the following Request Headers as extensions to the "PUT Bucket lifecycle" operation:

Name	Description	Required
x-gmt-tieringinfo	<p>This request header enables you to configure a bucket for schedule-based automatic transitioning of objects from local HyperStore storage to a remote storage system. For background information on the HyperStore auto-tiering feature, see <b>"Auto-Tiering Feature Overview"</b> (page 176).</p> <p>The <i>x-gmt-tieringinfo</i> header is formatted as follows:</p> <pre>x-gmt-tieringinfo: PROTOCOL EndPoint:Endpoint,Action:Action[,Mode:proxy] [,TieringBucket:TieringBucket]</pre> <ul style="list-style-type: none"> <li>• <i>PROTOCOL</i> (mandatory) — Specify one of these values: <ul style="list-style-type: none"> <li>◦ S3 -- Transition the objects to Amazon S3 storage.</li> <li>◦ S3GLACIER -- Transition the objects to Amazon Glacier.</li> <li>◦ GCS -- Transition the objects to Google Cloud Storage.</li> <li>◦ AZURE -- Transition the objects to Microsoft Azure.</li> <li>◦ SPECTRA -- Transition the objects to a Spectra Logic BlackPearl destination.</li> </ul> </li> </ul> <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0; background-color: #f9f9f9;"> <p><b>Note</b> If you are tiering to an S3-compliant system other than Amazon S3, Glacier, or Google Cloud Storage, use "S3" as the protocol. This would include, for instance, tiering to a remote HyperStore region or system.</p> </div> <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0; background-color: #f9f9f9;"> <p><b>Note</b> Auto-tiering restrictions based on destination type:</p> <ul style="list-style-type: none"> <li>* By default the largest object size that can be auto-tiered to Amazon, Google, or other S3-compliant destinations is 50GB. If you want to tier objects larger than this, consult with Cloudian Support. This 50GB limit does not apply to tiering to Azure or Spectra BlackPearl.</li> <li>* Tiering to Azure or Spectra BlackPearl is not supported for source buckets that have versioning enabled or that have had versioning enabled in the past.</li> <li>* 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.</li> </ul> </div> <ul style="list-style-type: none"> <li>• <i>EndPoint:Endpoint</i> (mandatory) — The service endpoint URL to use as your</li> </ul>	No

Name	Description	Required
	<p>auto-tiering destination. For example with Amazon S3, choose the region endpoint that's most suitable for your location (such as <code>s3-us-west-1.amazonaws.com</code> if your organization is in northern California). Or in the case of Spectra BlackPearl, specify the URL for your Spectra BlackPearl destination.</p> <p>If your ultimate tiering destination is Glacier, you must specify an Amazon S3 endpoint here, <b>not</b> a Glacier endpoint. The HyperStore system will first transition the objects to your specified Amazon S3 endpoint and then from there they will be immediately transitioned to the corresponding Glacier location.</p> <p>If you want to auto-tier to an external HyperStore system -- not a different region within the same HyperStore system but rather a different HyperStore system altogether -- see <b>"Note About Tiering to a Different HyperStore Region or System"</b> (page 182), in regard to the format requirements for the tiering endpoint.</p> <div data-bbox="453 788 1305 954"> <p><b>Note</b> You must use nested URL encoding. First URL encode the Endpoint value (the endpoint itself), and then URL encode the whole <code>x-gmt-tieringinfo</code> value.</p> </div> <div data-bbox="453 976 1305 1137"> <p><b>Note</b> Once you've configured an auto-tiering lifecycle on a source bucket you cannot subsequently change the tiering endpoint for that source bucket.</p> </div> <ul style="list-style-type: none"> <li>• Action:<i>Action</i> (mandatory) — This parameter specifies how the HyperStore system will handle S3 <i>GET Object</i> requests for objects that have been transitioned to the tiering destination. The choices are: <ul style="list-style-type: none"> <li>◦ stream — If the client submits a <i>GET Object</i> request to HyperStore, the HyperStore system retrieves the object from the destination and streams it through to the client. This method is supported only if the <i>Protocol</i> is S3, GCS, or AZURE.</li> <li>◦ nostream — If the client submits a <i>GET Object</i> request to HyperStore, the HyperStore system rejects the GET request. Instead, clients must submit a <i>POST Object restore</i> request in order to temporarily restore a copy of the object to local HyperStore storage.</li> </ul> <p>If the <i>Protocol</i> is S3, GCS, or AZURE you can use either "stream" or "nostream". If the <i>Protocol</i> is S3GLACIER or SPECTRA you must use "nostream" (the "stream" option is not supported for those destinations).</p> </li> <li>• Mode:proxy (optional) — If you specify this option, then: <ul style="list-style-type: none"> <li>◦ <b>All objects uploaded to the bucket from this point forward</b> (all objects uploaded after you successfully submit the <i>PUT Bucket lifecycle</i> request) will be <b>immediately</b> transitioned to the destination system.</li> <li>◦ Any objects already in the bucket at the time that you submit the <i>PUT Bucket lifecycle</i> request will be subject to the transition schedule</li> </ul> </li> </ul>	

Name	Description	Required
	<p>defined in the request body.</p> <div data-bbox="464 320 1235 734" style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <p><b>Note</b> If you want to use proxy mode and do not want schedule-based tiering of any objects that are already in the bucket, you <b>still must include a request body</b>. In this case in the lifecycle configuration in the request body you can set the "Status" attribute to "Disabled" (for an example see <b>"Request Body if Using Proxy Mode"</b> (page 975)). The result will be that any objects already in the bucket will not be auto-tiered, and all objects subsequently uploaded to the bucket (after successfully submitting the <i>PUT Bucket lifecycle</i> request) will be immediately moved to the destination system.</p> </div> <p>Proxy mode is supported only if the <i>Protocol</i> is S3, GCS, or AZURE (proxy mode is not supported for S3GLACIER or SPECTRA tiering). For more information on proxy mode -- also known as "bridge mode" -- see <b>"Auto-Tiering Feature Overview"</b> (page 176).</p> <ul style="list-style-type: none"> <li>• TieringBucket: <i>TieringBucket</i> (optional) — The name of the bucket to transition objects into, in the tiering destination system. This can be either: <ul style="list-style-type: none"> <li>◦ The <b>name of a bucket that already exists in the destination system</b>, and for which you are the bucket owner. In this case HyperStore will use this existing bucket as the tiering destination.</li> <li>◦ The <b>name of a bucket that you want HyperStore to create in the destination system</b>, to use as the tiering destination. Be sure to choose a bucket name that is very likely to be unique in the destination system. If your supplied bucket name is not unique in the destination system, HyperStore will be unable to create the bucket and the <i>PUT Bucket lifecycle</i> request will fail.</li> </ul> </li> </ul> <p>If you <b>omit</b> the tiering bucket parameter, then in the destination system HyperStore will create a tiering bucket named as follows:</p> <p style="text-align: center;"><i>&lt;origin-bucket-name-truncated-to-34-characters&gt;-&lt;28-character-random-string&gt;</i></p> <p><b>Example <i>x-gmt-tieringinfo</i> request headers:</b></p> <div data-bbox="301 1594 1235 2000" style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <pre># Example 1 (before URL encoding) Tiering to Amazon S3, into target bucket # named 'bucket12'. Streaming for local GETs will be supported.  x-gmt-tieringinfo: S3 EndPoint:http://s3.amazonaws.com,Action:stream, TieringBucket:bucket12  # Example 1 after nested URL encoding (endpoint value first, then whole # header value)  x-gmt-tieringinfo: S3%7CEndPoint%3Ahttp%253A%252F%252Fs3.amazonaws.com %2CAction%3Astream%2CTieringBucket%3Abucket12</pre> </div>	

Name	Description	Required
	<pre># Example 2 (before URL encoding) Tiering to Azure. HyperStore will derive target # bucket name from source bucket name. Streamed local GETs will not be supported, # clients must use Restore.  x-gmt-tieringinfo: AZURE EndPoint:https://blob.core.windows.net,Action:nostream  # Example 2 after nested URL encoding (endpoint value first, then whole # header value)  x-gmt-tieringinfo: AZURE%7CEndPoint%3Ahttps%253A%252F%252Fblob.core.windows.net %2CAction%3Anostream</pre> <p><b>IMPORTANT !</b> If you use the <i>x-gmt-tieringinfo</i> request header, then <b>for the request body element "StorageClass" you must specify "GLACIER"</b>. Set the Storage Class to GLACIER even if your final tiering destination is Amazon S3 or some other S3-compliant destination.</p>	
x-gmt-compare	<p>If you include this header in your "PUT Bucket lifecycle" request and set the header value to "LAT", then in lifecycle rules that you configure with the "Days" comparator the rule will be implemented as number of days since the object's <b>Last Access Time</b>.</p> <p>If you do not use this extension header, or if you include the header but assign it no value or any value other than "LAT", then "Days" based lifecycle rules will be implemented as number of days since the object's Creation Time (the default Amazon S3 behavior).</p> <p>You can use this header to create:</p> <ul style="list-style-type: none"> <li>• Last Access Time based auto-tiering rules (use this header and also the <i>x-gmt-tierinfo</i> header).</li> <li>• Last Access Time based expiration rules (use this header but not the <i>x-gmt-tierinfo</i> header).</li> </ul> <p><b>Note</b> An object's Last Access Time is updated if the object is accessed either for retrieval (GET or HEAD) or modification (PUT/POST/Copy). If an object is created and then never accessed, its Last Access Time will be its Creation Time.</p> <p><b>Note</b> If you use the <i>x-gmt-compare</i> header and set it to "LAT", it does not apply to any in <i>NoncurrentVersionTransition</i> or <i>NoncurrentVersionExpiration</i> rules within the lifecycle policy (for non-current versions of versioned objects).</p>	No

Name	Description	Required
	These types of rules are always based on the time elapsed since an object version became non-current (was replaced by a new version of the object).	
x-gmt-post-tier-copy	<p>If you use the <i>x-gmt-tieringinfo</i> request header to configure auto-tiering for a bucket, you can optionally also use the <i>x-gmt-post-tier-copy</i> request header to <b>specify a number of days for which a local copy of auto-tiered objects should be retained</b>. For example if you set <i>x-gmt-post-tier-copy: 7</i> then after each object is auto-tiered to the tiering destination, a copy of the object will be kept in the HyperStore source bucket for 7 days. After that the local copy will be deleted and only object metadata will be retained locally.</p> <p>There is no upper limit on this value. So if you want the local copy retention period to be practically limitless, you could for example set this header to 36500 to indicate a local copy retention period of 100 years.</p> <p>If you <b>omit</b> the <i>x-gmt-post-tier-copy</i> request header, then by default local objects are deleted after they are successfully auto-tiered to the tiering destination system, and only object metadata is retained locally.</p>	No

### 13.2.49.2. Request Elements

- AbortIncompleteMultipartUpload
- Date
- Days
- DaysAfterInitiation
- Expiration
- ExpiredObjectDeleteMarker
- Filter
- ID
- LifecycleConfiguration
- NoncurrentDays
- NoncurrentVersionExpiration
- NoncurrentVersionTransition
- Prefix

**Note** If you are using "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.

- Rule
- Status
- StorageClass

**Note** If you are using the HyperStore extension request header *x-gmt-tieringinfo*, then for the request element "StorageClass" you must specify "GLACIER". Set the Storage Class to GLACIER even if your final tiering destination is Amazon S3, Google, Azure, or some other system.

Note that in the case of tiering to Azure, if you do a *GET Bucket (List Objects)* call on a bucket, for any objects that have been tiered to Azure the response will indicate that the storage class of those tiered objects is STANDARD (despite your having specified GLACIER as the storage class in the *PUT Bucket Lifecycle* call). This behavior is expected. With objects tiered to any other destination, a *GET Bucket (List Objects)* call will indicate that tiered objects have the storage class that you specified in the *PUT Bucket Lifecycle* call.

- Transition

### 13.2.49.2.1. Request Body if Using "Proxy Mode"

If you are applying a "proxy mode" bucket lifecycle (see "HyperStore Extension to the S3 API" in the "Request Headers" section above), you still must include a request body. The lifecycle configuration specified in the request body will apply to objects already in the bucket (if any) while the proxy mode (for immediate transitioning) will apply to all objects uploaded after the *PUT Bucket lifecycle* request is successfully submitted. If there are already objects in the bucket and you don't want any transitioning applied to them, make the request body a lifecycle configuration with the "Status" attribute set to "Disabled" such as in the following example:

```
<LifecycleConfiguration>
  <Rule>
    <ID>DummyLifecycleConfig</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Disabled</Status>
    <Transition>
      <StorageClass>GLACIER</StorageClass>
      <Days>0</Days>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

## 13.2.50. PutBucketLogging

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

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [PutBucketLogging](#)

*Former operation name: PUT Bucket logging*

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

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

### 13.2.50.1. Request Elements

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

### 13.2.51. PutBucketNotificationConfiguration

Enables notifications of specified events for a bucket.

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [PutBucketNotificationConfiguration](#)

*Former operation name: PUT Bucket notification*

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

**Note** HyperStore's bucket notification feature and its SQS Service (for notification message queueing and delivery) are **disabled by default**. For information on how to enable this feature set see **"HyperStore Support for the AWS SQS API"** (page 1041).

### 13.2.51.1. Request Elements

- NotificationConfiguration
- QueueConfiguration

- Id
- Event

For event types, HyperStore supports only the following:

- s3:ObjectCreated:\*
- s3:ObjectCreated:Put
- s3:ObjectCreated:Post
- s3:ObjectCreated:Copy
- s3:ObjectCreated:CompleteMultipartUpload
- s3:ObjectRemoved:\*
- s3:ObjectRemoved:Delete
- s3:ObjectRemoved:DeleteMarkerCreated
- Queue
- Filter
- S3Key
- FilterRule
- Name
- Value

## 13.2.52. PutBucketPolicy

Applies an S3 bucket policy to an S3 bucket.

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [PutBucketPolicy](#)

*Former operation name: PUT Bucket policy*

### 13.2.52.1. Request Elements

The request body is a JSON-formatted bucket policy containing one or more policy statements. Within a policy's *Statement* block(s), HyperStore support for policy statement elements and their values is as follows:

- *Sid* -- Same as Amazon: Custom string identifying the statement, for example "Statement1" or "Only allow access from partner source IPs"
- *Effect* -- Same as Amazon: "Allow" or "Deny"
- *Principal* -- Must be one of the following:
  - "\*" -- Statement applies to all users.
  - {"CanonicalUser": "<canonicalUserId>"} -- Statement applies to the specified user.
  - {"CanonicalUser": ["<canonicalUserId>", "<canonicalUserId>"]} -- Statement applies to the specified users.

**Note** You can obtain a HyperStore user's canonical ID by retrieving the user through the CMC's **"Manage Users"** (page 262) page or by using the Admin API method [GET /user](#).

**Note** HyperStore **does not support having an IAM user as the Principal** in a bucket policy. Instead any specified users must be HyperStore root account holders.

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

#### 13.2.52.1.1. Supported "Action" Values

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

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

#### Object Actions

- s3:AbortMultipartUpload
- s3:BypassGovernanceRetention
- s3:DeleteObject
- s3:DeleteObjectTagging
- s3:DeleteObjectVersion
- s3:DeleteObjectVersionTagging
- s3:GetObject
- s3:GetObjectAcl
- s3:GetObjectLegalHold
- s3:GetObjectRetention
- s3:GetObjectTagging
- x3:GetObjectTorrent
- s3:GetObjectVersion
- s3:GetObjectVersionAcl
- s3:GetObjectVersionTagging
- s3:ListMultipartUploadParts
- s3:PutObject
- s3:PutObjectAcl
- s3:PutObjectLegalHold
- s3:PutObjectRetention
- s3:PutObjectTagging

- s3:PutObjectVersionAcl
- s3:PutObjectVersionTagging
- s3:RestoreObject

**Bucket Actions**

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

**Bucket Subresource Actions**

- s3>DeleteBucketPolicy
- s3>DeleteBucketWebsite
- s3:GetBucketAcl
- s3:GetBucketCORS
- s3:GetBucketLocation
- s3:GetBucketLogging
- s3:GetBucketNotification
- s3:GetBucketObjectLockConfiguration
- s3:GetBucketPolicy
- s3:GetBucketRequestPayment
- s3:GetBucketTagging
- s3:GetBucketVersioning
- s3:GetBucketWebsite
- s3:GetLifecycleConfiguration
- s3:GetReplicationConfiguration
- s3:PutBucketAcl
- s3:PutBucketCORS
- s3:PutBucketLogging
- s3:PutBucketNotification
- s3:PutBucketObjectLockConfiguration
- s3:PutBucketPolicy
- s3:PutBucketRequestPayment
- s3:PutBucketTagging
- s3:PutBucketVersioning
- s3:PutBucketWebsite
- s3:PutLifecycleConfiguration
- s3:PutReplicationConfiguration

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

### 13.2.52.1.2. Supported "Condition" Values

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

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

#### Condition Operators

- IpAddress

**Note** If you are using load balancers in front of the HyperStore S3 Service, then IP address based bucket policies 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 "[s3\\_proxy\\_protocol\\_enabled](#)" (page 530) in "[common.csv](#)" (page 512). For guidance on configuring the load balancers consult with Clouidian Sales Engineering or Support.

Note that using the "X-Forwarded-For" HTTP header is **not** sufficient to support IP address based bucket policies. You must use PROXY Protocol if you have load balancers in front of the S3 Service and want to use IP address based bucket policies .

- NotIpAddress
- NumericEquals
- NumericNotEquals
- NumericLessThan
- NumericLessThanEquals
- NumericGreaterThan
- NumericGreaterThanEquals
- StringEquals
- StringNotEquals
- StringEqualsIgnoreCase
- StringNotEqualsIgnoreCase
- StringLike
- ForAllValues:StringLike

- ForAnyValue:StringLike
- StringNotLike

### Condition Keys

- aws:Referer
- aws:SourceIp

**Note** If you create a bucket policy that restricts access based on source IP address, these restrictions will not apply to IP addresses within your HyperStore cluster. IP addresses from within your cluster are automatically "whitelisted".

- s3:delimiter
- s3:ExistingObjectTag/<tag-key>
- s3:LocationConstraint
- s3:max-keys
- s3:prefix
- s3:RequestObjectTag/<tag-keys>
- s3:RequestObjectTagKeys
- s3:VersionId
- s3:x-amz-acl
- s3:x-amz-copy-source
- s3:x-amz-grant-full-control
- s3:x-amz-grant-read
- s3:x-amz-grant-read-acp
- s3:x-amz-grant-write
- s3:x-amz-grant-write-acp
- s3:x-amz-metadata-directive
- s3:x-amz-server-side-encryption
- s3:object-lock-mode
- s3:object-lock-retain-until-date
- s3:object-lock-legal-hold
- s3:object-lock-remaining-retention-days

For examples of the kinds of things you can do with bucket policies, see the AWS documentation on [Bucket Policy Examples](#).

### 13.2.53. PutBucketReplication

Creates a replication configuration or replaces an existing one.

Along with the [common headers](#), HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: [PutBucketReplication](#)

*Former operation name: PUT Bucket replication*

**IMPORTANT !** Unlike Amazon S3, HyperStore does not require that you set up an IAM Role (or anything analogous) in order to use bucket replication. Also, HyperStore does not require that the destination bucket be in a different region than the source bucket. With HyperStore you can replicate to a destination bucket that's in the same region as the source bucket, if you want to.

Apart from the above exceptions, HyperStore bucket replication has the same requirements as Amazon S3 bucket replication, including that **versioning must be enabled** (using the **"PutBucketVersioning"** (page 983) operation) on both the source bucket and the destination bucket in order to use bucket replication.

**Note** Do not set a cross-region replication configuration and a [bucket lifecycle rule](#) on the same bucket.

### 13.2.53.1. Request Headers

- Content-MD5

### 13.2.53.2. Request Elements

- ReplicationConfiguration
- Role

**Note** As with the Amazon S3 API specification, for HyperStore the "Role" element must be included in the PUT Bucket replication request. However, HyperStore ignores the "Role" element's value (so, you can use any random string as its value). HyperStore does not use an IAM role or anything analogous when implementing cross-region replication.

- Rule
- ID
- Status
- Prefix
- Destination
- Bucket

**Note** Use the same "Bucket" value formatting as in the Amazon S3 API spec, i.e.

- `arn:aws:s3:::<bucketname>`.
- StorageClass

**Note** If you include this optional element in the request, HyperStore ignores its value.

### 13.2.54. PutBucketTagging

Sets the tags for a bucket.

Along with the [common headers](#), HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: [PutBucketTagging](#)

*Former operation name: PUT Bucket tagging*

**Note** 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](#).

#### 13.2.54.1. Request Headers

- Content-MD5

#### 13.2.54.2. Request Elements

- Tagging
- TagSet
- Tag
- Key
- Value

### 13.2.55. PutBucketVersioning

Sets the versioning state of an existing bucket.

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [PutBucketVersioning](#)

*Former operation name: PUT Bucket versioning*

#### 13.2.55.1. Request Elements

- Status
- VersioningConfiguration

### 13.2.56. PutBucketWebsite

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

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [PutBucketWebsite](#)

*Former operation name: PUT Bucket website*

### 13.2.56.1. Request Elements

- WebsiteConfiguration
- RedirectAllRequestsTo
- HostName
- Protocol
- WebsiteConfiguration
- IndexDocument
- ErrorDocument

### 13.2.57. PutObject

Adds an object to a bucket.

Along with the [common headers](#), HyperStore supports the operation-specific headers listed below.

For operation details and examples see the AWS documentation: [PutObject](#)

*Former operation name: PUT Object*

#### 13.2.57.1. Request Headers

- Cache-Control
- Content-Disposition
- Content-Encoding
- Content-Length
- Content-MD5
- Content-Type
- Expect
- Expires
- x-amz-meta-\*

**Note** The metadata values must be UTF-8 and must not contain control characters less than 0x20 except for \r, \n, and \t. Also, normal XML escaping is required where appropriate.

- x-amz-storage-class

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

- x-amz-tagging
- x-amz-website-redirect-location
- x-amz-object-lock-mode

**Note** For more information on HyperStore's support for the S3 "Object Lock" feature, see **"WORM (Object Lock)"** (page 121).

- x-amz-object-lock-retain-until-date
- x-amz-object-lock-legal-hold
- x-amz-acl
- x-amz-grant-read
- x-amz-grant-write
- x-amz-grant-read-acp
- x-amz-grant-write-acp
- x-amz-grant-full-control
- x-amz-server-side-encryption

**Note** For information about HyperStore's support of the *x-amz-server-side-encryption* and *x-amz-server-side-encryption-customer-\** request headers, and set-up steps that you must perform in order to use HyperStore's server-side encryption features, see **"Server-Side Encryption"** (page 105).

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

#### *HyperStore Restrictions on Object Names*

The following control characters cannot be used anywhere in an object name and will result in a 400 Bad Request response: 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A ("\""), 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 "/"

#### 13.2.57.2. Response Headers

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

## 13.2.58. PutObjectAcl

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

Along with the [common headers](#), HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: [PutObjectAcl](#)

*Former operation name: PUT Object acl*

### 13.2.58.1. Request Headers

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

### 13.2.58.2. Request Elements

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

### 13.2.58.3. Response Headers

- x-amz-version-id

## 13.2.59. PutObjectLegalHold

Applies a Legal Hold configuration to the specified object.

Along with the [common headers](#), HyperStore supports the operation-specific parameters and elements listed below.

For operation details and examples see the AWS documentation: [PutObjectLegalHold](#)

*Former operation name: PUT Object legal hold*

**Note** For more information on HyperStore's support for the S3 "Object Lock" feature, see **"WORM (Object Lock)"** (page 121).

### 13.2.59.1. Query Parameters

- versionId

### 13.2.59.2. Request Elements

- LegalHold
- Status

## 13.2.60. PutObjectLockConfiguration

Places an Object Lock configuration on the specified bucket.

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [PutObjectLockConfiguration](#)

*Former operation name: PUT Bucket object lock configuration*

**Note** For more information on HyperStore's support for the S3 "Object Lock" feature, see **"WORM (Object Lock)"** (page 121).

### 13.2.60.1. Request Elements

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

## 13.2.61. PutObjectRetention

Places an Object Retention configuration on an object.

Along with the [common headers](#), HyperStore supports the operation-specific parameters, headers, and elements listed below.

For operation details and examples see the AWS documentation: [PutObjectRetention](#)

*Former operation name: PUT Object retention*

**Note** For more information on HyperStore's support for the S3 "Object Lock" feature, see **"WORM (Object Lock)"** (page 121).

#### 13.2.61.1. Query Parameters

- versionId

#### 13.2.61.2. Request Headers

- x-amz-bypass-governance-retention

#### 13.2.61.3. Request Elements

- Retention
- Mode
- RetainUntilDate

### 13.2.62. PutObjectTagging

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

Along with the [common headers](#), HyperStore supports the operation-specific elements listed below.

For operation details and examples see the AWS documentation: [PutObjectTagging](#)

*Former operation name: PUT Object tagging*

#### 13.2.62.1. Request Elements

- Tagging
- TagSet
- Tag
- Key
- Value

### 13.2.63. RestoreObject

Restores a tiered object back into HyperStore.

Along with the [common headers](#), HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: [RestoreObject](#)

*Former operation name: POST Object restore*

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

Glacier. For information about the HyperStore auto-tiering feature, see **"Auto-Tiering Feature Overview"** (page 176).

### 13.2.63.1. Request Headers

- Content-MD5

### 13.2.63.2. Request Elements

- RestoreRequest
- Days
- GlacierJobParameters

**Note** For the sake of S3 API compatibility, HyperStore's S3 Service allows the request elements *GlacierJobParameters* and *Tier* to be included in a "POST Object restore" request -- but in the current HyperStore release these elements will have no effect on how the restore request is implemented.

- Tier

## 13.2.64. UploadPart

Uploads a part in a multipart upload.

Along with the [common headers](#), HyperStore supports the operation-specific headers listed below.

For operation details and examples see the AWS documentation: [UploadPart](#)

*Former operation name: Upload Part*

### 13.2.64.1. Request Headers

- Content-Length
- Content-MD5
- Expect
- x-amz-server-side-encryption
- x-amz-server-side-encryption-customer-algorithm

**Note** For information about HyperStore's support of the *x-amz-server-side-encryption-customer-*\* request headers, and set-up steps that you must perform in order to use HyperStore's server-side encryption features, see **"Server-Side Encryption"** (page 105).

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

### 13.2.64.2. Response Headers

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

### 13.2.65. UploadPartCopy

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

Along with the [common headers](#), HyperStore supports the operation-specific headers and elements listed below.

For operation details and examples see the AWS documentation: [UploadPartCopy](#)

*Former operation name: Upload Part - Copy*

#### 13.2.65.1. Request Headers

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

#### 13.2.65.2. Response Headers

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

#### 13.2.65.3. Response Elements

- CopyPartResult
- ETag
- LastModified

# Chapter 14. IAM API

## 14.1. Introduction

### 14.1.1. HyperStore Support for the AWS IAM API

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Restrictions and Limitations in HyperStore's IAM Support"** (page 991)
- **"HyperStore IAM Extensions to Support RBAC for Admin Functions"** (page 991)
- **"Deleting or Suspending HyperStore Users Who Have Created IAM Users"** (page 992)
- **"Disabling the HyperStore IAM Service"** (page 992)
- **"The IAM Service in Multi-Region Systems"** (page 992)

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

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

#### 14.1.1.1. Restrictions and Limitations in HyperStore's IAM Support

- HyperStore supports most but not all of the Amazon IAM API "Actions". For the list of supported actions see Section 14.2 "Supported IAM Actions".
- HyperStore supports most but not all of the Amazon IAM API policy elements, actions, resources, and condition keys. For more information see **"Supported IAM Policy Elements"** (page 1023)
- HyperStore does not support editing managed policies, after they've been created. Likewise HyperStore does not support managed policy "versions". HyperStore does support editing "inline" policies.
- IAM users cannot login to the CMC, and cannot use the CMC as their S3 client application. To access the HyperStore S3 Service, IAM users must use a third party S3 client application.

#### 14.1.1.2. HyperStore IAM Extensions to Support RBAC for Admin Functions

The HyperStore implementation of the IAM API includes extensions that:

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

- Allow HyperStore system admins, group admins, or regular users to grant their IAM users permission to execute those same read-only HyperStore administrative functions.

For more information, including information about the client tool that HyperStore provides to help you use this feature, see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

### 14.1.1.3. Deleting or Suspending HyperStore Users Who Have Created IAM Users

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

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

### 14.1.1.4. Disabling the HyperStore IAM Service

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

1. On your Puppet master node open this configuration file in a text editor:

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

2. Change the setting `iam_service_enabled,true` to `iam_service_enabled,false` and save your change.
3. Push your change to the cluster and restart the S3 Service. If you need instructions see **"Pushing Configuration File Edits to the Cluster and Restarting Services"** (page 506).

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

### 14.1.1.5. The IAM Service in Multi-Region Systems

In a multi-region HyperStore system, the IAM Service runs only on nodes in the [default service region](#). In your [DNS set-up](#), the IAM service endpoint should resolve to a load balancer that distributes traffic to HyperStore nodes in the default service region.

## 14.1.2. IAM Client Application Options

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"CMC Support for IAM Functions"** (page 993)
- **"Accessing the IAM Service with a Third Party or Custom Client Application"** (page 993)
- **"Creating S3 Access Credentials for the Default System Admin User"** (page 993)

Users can access and use the HyperStore IAM Service either through the CMC, a third party client application that supports IAM calls, or a custom IAM client that you develop. Whether using the CMC or a third party or

custom client application, application users must have S3 access credentials (access key ID and secret key) in order to use the HyperStore IAM Service.

#### 14.1.2.1. CMC Support for IAM Functions

Through the CMC, HyperStore account root users can:

- [Add, Manage, and Delete IAM Users](#)
- [Add, Manage, and Delete IAM Groups](#)
- [Add, Manage, and Delete IAM Policies](#)

#### 14.1.2.2. Accessing the IAM Service with a Third Party or Custom Client Application

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

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

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

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

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

Note that:

- These are the **default** service endpoints for the HyperStore IAM Service. You can customize the endpoints as described in "**Changing S3, Admin, CMC, or IAM Service Endpoints**" (page 600)..
- The HyperStore IAM Service by default uses a self-signed certificate for its HTTPS listener, so if you are using HTTPS to access the service your client application must be configured to allow self-signed certificates.
- You must configure your DNS environment to resolve the IAM Service endpoint as described in DNS Set-Up "DNS Set-Up" in the *HyperStore Installation Guide*.

#### 14.1.2.3. Creating S3 Access Credentials for the Default System Admin User

If you want the default HyperStore system admin user -- the user whose user ID is "admin" in the CMC -- to be able to use the IAM Service, do the following:

1. Log into the CMC as the "admin" user. (You will see that an **IAM** tab now displays in the CMC interface, but clicking that tab will return an authorization error until after you've completed Steps 2 and 3 below.)
2. On the right side of the CMC's top navigation bar, hold your cursor over your login name ("admin") and then in the drop-down menu select **Security Credentials**.
3. In the security credentials page's **S3 Access Credentials** section, click **Create New Key**.

This creates S3 access credentials (access key ID and secret key) for the "admin" user. S3 access credentials are required in order to access HyperStore's IAM Service. The CMC will use these credentials automatically when the "admin" user uses the CMC to access IAM functions (on the **IAM** tab). Or if you are using a third party application to access the HyperStore IAM Service, you will need to provide the credentials to that application.

**Note** If you created any additional system admin users prior to the HyperStore 7.1 release, and if you want those system admin users to be able to use the IAM Service, those system admin users will need

to complete the steps described above to create S3 access credentials for themselves.

Regular users and group admins created in the CMC are given S3 credentials automatically as part of the user creation process, so such users already have the credentials that they need to access the IAM Service. Also, additional system admins that you create in HyperStore 7.1 or later are automatically given S3 credentials.

### 14.1.3. IAM Common Request Parameters

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

- Action
- Version

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

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

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

### 14.1.4. IAM Common Errors

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

- AccessDenied
- IncompleteSignature
- InternalFailure
- InvalidAction
- InvalidClientTokenId
- InvalidParameterCombination
- InvalidParameterValue
- InvalidQueryParameter
- MalformedQueryString

- MissingAction
- MissingAuthenticationToken
- MissingParameter
- OptInRequired
- RequestExpired
- ServiceUnavailable
- ThrottlingException
- ValidationError

## 14.2. Supported IAM Actions

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

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

### 14.2.1. AddUserToGroup

Adds the specified user to the specified group.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [AddUserToGroup](#)

#### 14.2.1.1. Request Parameters

- GroupName
- UserName

#### 14.2.1.2. Errors

- LimitExceeded
- NoSuchEntity
- ServiceFailure

### 14.2.2. AttachGroupPolicy

Attaches the specified managed policy to the specified IAM group.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [AttachGroupPolicy](#)

#### 14.2.2.1. Request Parameters

- GroupName
- PolicyArn

#### 14.2.2.2. Errors

- InvalidInput
- LimitExceeded
- NoSuchEntity
- PolicyNotAttachable
- ServiceFailure

### 14.2.3. AttachRolePolicy

Attaches the specified managed policy to the specified IAM role.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [AttachRolePolicy](#)

#### 14.2.3.1. Request Parameters

- PolicyArn
- RoleName

#### 14.2.3.2. Errors

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

### 14.2.4. AttachUserPolicy

Attaches the specified managed policy to the specified user.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [AttachUserPolicy](#)

#### 14.2.4.1. Request Parameters

- PolicyArn
- UserName

#### 14.2.4.2. Errors

- InvalidInput
- LimitExceeded
- NoSuchEntity
- PolicyNotAttachable
- ServiceFailure

### 14.2.5. CreateAccessKey

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

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

For action details and examples see the AWS documentation: [CreateAccessKey](#)

#### 14.2.5.1. Request Parameters

- UserName

#### 14.2.5.2. Response Elements

- AccessKey

#### 14.2.5.3. Errors

- LimitExceeded
- NoSuchEntity
- ServiceFailure

**Note** By default the HyperStore system allows only two key pairs per IAM user. This restriction is configurable by the **"credentials.iamuser.max"** (page 571) setting in *mts.properties.erb*. Note that an IAM user's inactive credentials (if any) count toward this limit, as well as active credentials.

### 14.2.6. CreateGroup

Creates a new group.

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

For action details and examples see the AWS documentation: [CreateGroup](#)

#### 14.2.6.1. Request Parameters

- GroupName
- Path

### 14.2.6.2. Response Elements

- Group

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

For example: `e97eb4557aea18781f53eb2b8f7e282e|iamgroup2`

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

Similarly, the "Arn" attribute value will be in this format:

`arn:aws:iam::<Canonical-UID-of-HyperStore-User>:group/<IAM-groupname>`

### 14.2.6.3. Errors

- EntityAlreadyExists
- LimitExceeded
- NoSuchEntity
- ServiceFailure

## 14.2.7. CreatePolicy

Creates a new managed policy under your HyperStore account.

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

For action details and examples see the AWS documentation: [CreatePolicy](#)

### 14.2.7.1. Request Parameters

- Description
- Path
- PolicyDocument

**Note** For information about HyperStore's IAM policy document support see "**Supported IAM Policy Elements**" (page 1023).

- PolicyName

### 14.2.7.2. Response Elements

- Policy

### 14.2.7.3. Errors

- EntityAlreadyExists
- InvalidInput
- LimitExceeded
- MalformedPolicyDocument
- ServiceFailure

## 14.2.8. CreateRole

Creates a new role under your account.

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

For action details and examples see the AWS documentation: [CreateRole](#)

See also the AWS documentation: [IAM Roles](#)

### 14.2.8.1. Request Parameters

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

Example role 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/saml" } }
  }
}
```

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/saml*.

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

*aws:TokenIssueTime*

*sts:ExternalId*

*saml:aud*

*saml:doc*

*saml:iss*

```
saml:namequalifier  
saml:sub  
saml:sub_type
```

#### 14.2.8.2. Response Elements

- Role

#### 14.2.8.3. Errors

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

### 14.2.9. CreateSAMLProvider

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

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

For action details and examples see the AWS documentation: [CreateSAMLProvider](#)

#### 14.2.9.1. Request Parameters

- Name
- SAMLMetadataDocument

#### 14.2.9.2. Response Elements

- SAMLProviderArn

#### 14.2.9.3. Errors

- EntityAlreadyExists
- LimitExceeded
- ServiceFailure

### 14.2.10. CreateUser

Creates a new IAM user under your account.

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

For action details and examples see the AWS documentation: [CreateUser](#)

### 14.2.10.1. Request Parameters

- Path
- UserName

### 14.2.10.2. Response Elements

- User

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

For example: `e97eb4557aea18781f53eb2b8f7e282e|iamuser2`

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

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

### 14.2.10.3. Errors

- EntityAlreadyExists
- LimitExceeded
- NoSuchEntity
- ServiceFailure

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

## 14.2.11. DeleteAccessKey

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

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [DeleteAccessKey](#)

### 14.2.11.1. Request Parameters

- AccessKeyId
- UserName

#### 14.2.11.2. Errors

- LimitExceeded
- NoSuchEntity
- ServiceFailure

### 14.2.12. DeleteGroup

Deletes the specified IAM group.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [DeleteGroup](#)

#### 14.2.12.1. Request Parameters

- GroupName

#### 14.2.12.2. Errors

- DeleteConflict
- LimitExceeded
- NoSuchEntity
- ServiceFailure

### 14.2.13. DeleteGroupPolicy

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

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [DeleteGroupPolicy](#)

#### 14.2.13.1. Request Parameters

- GroupName
- PolicyName

#### 14.2.13.2. Errors

- LimitExceeded
- NoSuchEntity
- ServiceFailure

### 14.2.14. DeletePolicy

Deletes the specified managed policy.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [DeletePolicy](#)

#### 14.2.14.1. Request Parameters

- PolicyArn

#### 14.2.14.2. Errors

- DeleteConflict
- InvalidInput
- LimitExceeded
- NoSuchEntity
- ServiceFailure

### 14.2.15. DeleteRole

Deletes the specified role.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [DeleteRole](#)

#### 14.2.15.1. Request Parameters

- RoleName

#### 14.2.15.2. Errors

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

### 14.2.16. DeleteRolePolicy

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

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [DeleteRolePolicy](#)

#### 14.2.16.1. Request Parameters

- PolicyName
- RoleName

### 14.2.16.2. Errors

- LimitExceeded
- NoSuchEntity
- ServiceFailure
- UnmodifiableEntity

## 14.2.17. DeleteSAMLProvider

Deletes a SAML provider resource in IAM.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [DeleteSAMLProvider](#)

### 14.2.17.1. Request Parameters

- SAMLProviderArn

### 14.2.17.2. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

## 14.2.18. DeleteUser

Deletes the specified IAM user.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [DeleteUser](#)

### 14.2.18.1. Request Parameters

- UserName

### 14.2.18.2. Errors

- DeleteConflict
- LimitExceeded
- NoSuchEntity
- ServiceFailure

## 14.2.19. DeleteUserPolicy

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

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [DeleteUserPolicy](#)

#### 14.2.19.1. Request Parameters

- PolicyName
- UserName

#### 14.2.19.2. Errors

- LimitExceeded
- NoSuchEntity
- ServiceFailure

### 14.2.20. DetachGroupPolicy

Removes the specified managed policy from the specified IAM group.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [DetachGroupPolicy](#)

#### 14.2.20.1. Request Parameters

- GroupName
- PolicyArn

#### 14.2.20.2. Errors

- InvalidInput
- LimitExceeded
- NoSuchEntity
- ServiceFailure

### 14.2.21. DetachRolePolicy

Removes the specified managed policy from the specified role.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [DetachRolePolicy](#)

#### 14.2.21.1. Request Parameters

- PolicyArn
- RoleName

#### 14.2.21.2. Errors

- InvalidInput
- LimitExceeded
- NoSuchEntity
- ServiceFailure
- UnmodifiableEntity

#### 14.2.22. DetachUserPolicy

Removes the specified managed policy from the specified user.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [DetachUserPolicy](#)

##### 14.2.22.1. Request Parameters

- PolicyArn
- UserName

##### 14.2.22.2. Errors

- InvalidInput
- LimitExceeded
- NoSuchEntity
- ServiceFailure

#### 14.2.23. GetGroup

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

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

For action details and examples see the AWS documentation: [GetGroup](#)

##### 14.2.23.1. Request Parameters

- GroupName

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

##### 14.2.23.2. Response Elements

- Group

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

For example: `e97eb4557aea18781f53eb2b8f7e282e|iamgroup2`

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

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

- IsTruncated

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

- Users.member.N

### 14.2.23.3. Errors

- NoSuchEntity
- ServiceFailure

## 14.2.24. GetGroupPolicy

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

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

For action details and examples see the AWS documentation: [GetGroupPolicy](#)

### 14.2.24.1. Request Parameters

- GroupName
- PolicyName

### 14.2.24.2. Response Elements

- GroupName
- PolicyDocument
- PolicyName

### 14.2.24.3. Errors

- NoSuchEntity
- ServiceFailure

### 14.2.25. GetPolicy

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

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [GetPolicy](#)

#### 14.2.25.1. Request Parameters

- PolicyArn

#### 14.2.25.2. Response Elements

- Policy

#### 14.2.25.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

### 14.2.26. GetPolicyVersion

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

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

For action details and examples see the AWS documentation: [GetPolicyVersion](#)

#### 14.2.26.1. Request Parameters

- PolicyArn

**Note** The "VersionId" request parameter, if submitted, is ignored and always defaults to "v1". HyperStore does not currently support IAM managed policy versioning. However, the "GetPolicyVersion" action is supported because this is the only action that returns the actual policy document (within the "PolicyVersion" object).

#### 14.2.26.2. Response Elements

- PolicyVersion

#### 14.2.26.3. Errors

- InvalidInput
- NoSuchEntity

- ServiceFailure

### 14.2.27. GetRole

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

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

For action details and examples see the AWS documentation: [GetRole](#)

#### 14.2.27.1. Request Parameters

- RoleName

#### 14.2.27.2. Response Elements

- Role

#### 14.2.27.3. Errors

- NoSuchEntity
- ServiceFailure

### 14.2.28. GetRolePolicy

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

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

For action details and examples see the AWS documentation: [GetRolePolicy](#)

#### 14.2.28.1. Request Parameters

- PolicyName
- RoleName

#### 14.2.28.2. Response Elements

- PolicyDocument
- PolicyName
- RoleName

#### 14.2.28.3. Errors

- NoSuchEntity
- ServiceFailure

### 14.2.29. GetSAMLProvider

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

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [GetSAMLProvider](#)

#### 14.2.29.1. Request Parameters

- SAMLProviderArn

#### 14.2.29.2. Response Elements

- CreateDate
- SAMLMetadataDocument
- ValidUntil

#### 14.2.29.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

### 14.2.30. GetUser

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

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

For action details and examples see the AWS documentation: [GetUser](#)

#### 14.2.30.1. Request Parameters

- UserName

#### 14.2.30.2. Response Elements

- User

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

For example: `e97eb4557aea18781f53eb2b8f7e282e|iamuser2`

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

created.

Similarly, the "Arn" attribute value will be in this format:

*arn:aws:iam::<Canonical-UID-of-HyperStore-User>:user/<IAM-username>*

### 14.2.30.3. Errors

- NoSuchEntity
- ServiceFailure

## 14.2.31. GetUserPolicy

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

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

For action details and examples see the AWS documentation: [GetUserPolicy](#)

### 14.2.31.1. Request Parameters

- PolicyName
- UserName

### 14.2.31.2. Response Elements

- PolicyDocument
- PolicyName
- UserName

### 14.2.31.3. Errors

- NoSuchEntity
- ServiceFailure

## 14.2.32. ListAccessKeys

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

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

For action details and examples see the AWS documentation: [ListAccessKeys](#)

### 14.2.32.1. Request Parameters

- UserName

#### 14.2.32.2. Response Elements

- AccessKeyMetadata.member.N
- IsTruncated

#### 14.2.32.3. Errors

- NoSuchEntity
- ServiceFailure

### 14.2.33. ListAttachedGroupPolicies

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

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

For action details and examples see the AWS documentation: [ListAttachedGroupPolicies](#)

#### 14.2.33.1. Request Parameters

- GroupName
- PathPrefix

#### 14.2.33.2. Response Elements

- AttachedPolicies.member.N
- IsTruncated

#### 14.2.33.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

### 14.2.34. ListAttachedRolePolicies

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

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

For action details and examples see the AWS documentation: [ListAttachedRolePolicies](#)

#### 14.2.34.1. Request Parameters

- PathPrefix
- UserName

### 14.2.34.2. Response Elements

- AttachedPolicies.member.N
- IsTruncated

### 14.2.34.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

## 14.2.35. ListAttachedUserPolicies

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

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

For action details and examples see the AWS documentation: [ListAttachedUserPolicies](#)

### 14.2.35.1. Request Parameters

- PathPrefix
- UserName

### 14.2.35.2. Response Elements

- AttachedPolicies.member.N
- IsTruncated

### 14.2.35.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

## 14.2.36. ListEntitiesForPolicy

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

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

For action details and examples see the AWS documentation: [ListEntitiesForPolicy](#)

### 14.2.36.1. Request Parameters

- EntityFilter
- PathPrefix
- PolicyArn

### 14.2.36.2. Response Elements

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

### 14.2.36.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

## 14.2.37. ListGroupPolicies

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

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

For action details and examples see the AWS documentation: [ListGroupPolicies](#)

### 14.2.37.1. Request Parameters

- GroupName

### 14.2.37.2. Response Elements

- IsTruncated
- PolicyNames.member.N

### 14.2.37.3. Errors

- NoSuchEntity
- ServiceFailure

## 14.2.38. ListGroups

Lists the IAM groups that have the specified path prefix.

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

For action details and examples see the AWS documentation: [ListGroups](#)

### 14.2.38.1. Request Parameters

- PathPrefix

### 14.2.38.2. Response Elements

- Groups.member.N
- IsTruncated

### 14.2.38.3. Errors

- ServiceFailure

## 14.2.39. ListGroupsForUser

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

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

For action details and examples see the AWS documentation: [ListGroupsForUser](#)

Request Parameters

- UserName

Response Elements

- Groups.member.N
- IsTruncated

Errors

- NoSuchEntity
- ServiceFailure

## 14.2.40. ListPolicies

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

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

For action details and examples see the AWS documentation: [ListPolicies](#)

### 14.2.40.1. Request Parameters

- OnlyAttached
- PathPrefix

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

### 14.2.40.2. Response Elements

- IsTruncated
- Policies.member.N

### 14.2.40.3. Errors

- ServiceFailure

## 14.2.41. ListPolicyVersions

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

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

For action details and examples see the AWS documentation: [ListPolicyVersions](#)

### 14.2.41.1. Request Parameters

- PolicyArn

### 14.2.41.2. Response Elements

- IsTruncated
- Versions.member.N

**Note** The only version returned will be "v1". HyperStore does not currently support IAM managed policy versioning.

### 14.2.41.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

## 14.2.42. ListRolePolicies

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

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

For action details and examples see the AWS documentation: [ListRolePolicies](#)

### 14.2.42.1. Request Parameters

- RoleName

### 14.2.42.2. Response Elements

- IsTruncated
- PolicyNames.member.N

### 14.2.42.3. Errors

- NoSuchEntity
- ServiceFailure

## 14.2.43. ListRoles

Lists the IAM roles that have the specified path prefix.

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

For action details and examples see the AWS documentation: [ListRoles](#)

### 14.2.43.1. Request Parameters

- PathPrefix

### 14.2.43.2. Response Elements

- IsTruncated
- Roles.member.N

### 14.2.43.3. Errors

- ServiceFailure

## 14.2.44. ListSAMLProviders

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

HyperStore supports the elements and errors listed below.

For action details and examples see the AWS documentation: [ListSAMLProviders](#)

### 14.2.44.1. Response Elements

- SAMLProviderList.member.N

### 14.2.44.2. Errors

- ServiceFailure

## 14.2.45. ListUserPolicies

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

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

For action details and examples see the AWS documentation: [ListUserPolicies](#)

#### 14.2.45.1. Request Parameters

- UserName

#### 14.2.45.2. Response Elements

- IsTruncated
- PolicyNames.member.N

#### 14.2.45.3. Errors

- NoSuchEntity
- ServiceFailure

### 14.2.46. ListUsers

Lists the IAM users that have the specified path prefix.

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

For action details and examples see the AWS documentation: [ListUsers](#)

#### 14.2.46.1. Request Parameters

- PathPrefix

#### 14.2.46.2. Response Elements

- IsTruncated
- Users.member.N

#### 14.2.46.3. Errors

- ServiceFailure

### 14.2.47. PutGroupPolicy

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

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [PutGroupPolicy](#)

#### 14.2.47.1. Request Parameters

- GroupName
- PolicyDocument

**Note** For information about HyperStore's IAM policy document support see "**Supported IAM Policy Elements**" (page 1023).

- PolicyName

#### 14.2.47.2. Errors

- LimitExceeded
- MalformedPolicyDocument
- NoSuchEntity
- ServiceFailure

### 14.2.48. PutRolePolicy

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

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [PutRolePolicy](#)

#### 14.2.48.1. Request Parameters

- PolicyDocument
- PolicyName
- RoleName

#### 14.2.48.2. Errors

- LimitExceeded
- MalformedPolicyDocument
- NoSuchEntity
- ServiceFailure
- UnmodifiableEntity

### 14.2.49. PutUserPolicy

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

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [PutUserPolicy](#)

#### 14.2.49.1. Request Parameters

- PolicyDocument

**Note** For information about HyperStore's IAM policy document support see "**Supported IAM Policy Elements**" (page 1023).

- PolicyName
- UserName

#### 14.2.49.2. Errors

- LimitExceeded
- MalformedPolicyDocument
- NoSuchEntity
- ServiceFailure

### 14.2.50. RemoveUserFromGroup

Removes the specified user from the specified group.

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [RemoveUserFromGroup](#)

#### 14.2.50.1. Request Parameters

- GroupName
- UserName

#### 14.2.50.2. Errors

- LimitExceeded
- NoSuchEntity
- ServiceFailure

### 14.2.51. UpdateAccessKey

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

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [UpdateAccessKey](#)

#### 14.2.51.1. Request Parameters

- AccessKeyId
- Status
- UserName

### 14.2.51.2. Errors

- LimitExceeded
- NoSuchEntity
- ServiceFailure

**Note** By default the HyperStore system allows only two key pairs per IAM user. This restriction is configurable by the "**credentials.iamuser.max**" (page 571) setting in *mts.properties.erb*. Note that an IAM user's inactive credentials (if any) count toward this limit, as well as active credentials.

## 14.2.52. UpdateAssumeRolePolicy

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

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [UpdateAssumeRolePolicy](#)

### 14.2.52.1. Request Parameters

- PolicyDocument
- RoleName

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

*aws:TokenIssueTime*  
*sts:ExternalId*  
*saml:aud*  
*saml:doc*  
*saml:iss*  
*saml:namequalifier*  
*saml:sub*  
*saml:sub\_type*

### 14.2.52.2. Errors

- LimitExceeded
- MalformedPolicyDocument
- NoSuchEntity
- ServiceFailure
- UnmodifiableEntity

## 14.2.53. UpdateGroup

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

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [UpdateGroup](#)

#### 14.2.53.1. Request Parameters

- GroupName
- NewGroupName
- NewPath

#### 14.2.53.2. Errors

- EntityAlreadyExists
- LimitExceeded
- NoSuchEntity
- ServiceFailure

### 14.2.54. UpdateRole

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

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [UpdateRole](#)

#### 14.2.54.1. Request Parameters

- Description
- MaxSessionDuration
- RoleName

#### 14.2.54.2. Errors

- NoSuchEntity
- ServiceFailure
- UnmodifiableEntity

### 14.2.55. UpdateRoleDescription

Although HyperStore supports this Action, it is recommended to use [UpdateRole](#) instead.

AWS documentation: [UpdateRoleDescription](#)

### 14.2.56. UpdateSAMLProvider

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

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

For action details and examples see the AWS documentation: [UpdateSAMLProvider](#)

#### 14.2.56.1. Request Parameters

- SAMLMetadataDocument
- SAMLProviderArn

#### 14.2.56.2. Response Elements

- SAMLProviderArn

#### 14.2.56.3. Errors

- InvalidInput
- NoSuchEntity
- ServiceFailure

### 14.2.57. UpdateUser

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

HyperStore supports the parameters and errors listed below.

For action details and examples see the AWS documentation: [UpdateUser](#)

#### 14.2.57.1. Request Parameters

- NewPath
- NewUserName
- UserName

#### 14.2.57.2. Errors

- EntityAlreadyExists
- EntityTemporarilyUnmodifiable
- LimitExceeded
- NoSuchEntity
- ServiceFailure

## 14.3. Supported IAM Policy Elements

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Policy Document Content for Granting S3 or IAM Permissions"** (page 1024)
- **"Policy Document Content for Granting HyperStore Administrative Permissions"** (page 1025)

This section describes HyperStore's support for IAM policy document content. IAM policies grant permissions to IAM groups and users. You can create IAM policy documents through the CMC or by using a third party or

custom IAM client. If you use the CMC, you have the choice of using a visual policy editor or using a JSON editor.

In regard to IAM policy document content, HyperStore supports most of the standard AWS IAM policy elements that define S3 permissions or IAM permissions. As an extension to the IAM standard, HyperStore also supports policy elements that define permissions for HyperStore administrative actions.

### 14.3.1. Policy Document Content for Granting S3 or IAM Permissions

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

**Note** In the current HyperStore release:

- \* HyperStore support for IAM policy document components is not fully comprehensive. HyperStore supports **most but not all** of the policy elements, actions, resources, and/or condition keys cited in the AWS documentation for IAM policy formation.
- \* IAM Actions are case sensitive.
- \* The use of the wildcard character "\*" to match missing text in an Action or Resource is not well supported. Explicit Action names or Resource ARNs can be grouped together in lists as a work-around.

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

- *Policies and Permissions*  
[http://docs.aws.amazon.com/IAM/latest/UserGuide/access\\_policies.html](http://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html)
- *IAM JSON Policy Elements Reference*  
[https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\\_policies\\_elements.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements.html)
- *Actions, Resources, and Condition Keys for Amazon S3*  
[https://docs.aws.amazon.com/IAM/latest/UserGuide/list\\_amazons3.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/list_amazons3.html)
- *Actions, Resources, and Condition Keys for Identity And Access Management*  
[https://docs.aws.amazon.com/IAM/latest/UserGuide/list\\_identityandaccessmanagement.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/list_identityandaccessmanagement.html)

**Note** HyperStore supports only the IAM actions listed in Section 14.2 "Supported IAM Actions".

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

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

### 14.3.2. Policy Document Content for Granting HyperStore Administrative Permissions

Along with granting S3 service permissions, you can also use IAM policies to grant IAM groups and users permissions to perform read-only HyperStore administrative actions. Note that:

- As is the case with S3 permissions, an **IAM user by default has no admin permissions** -- an IAM user gains permissions only if she is assigned an IAM policy that specifies those permissions, and she gains only the permissions specified in the policy.
- When an IAM policy grants an IAM user permission to an administrative action, the IAM user's permission scope in respect to that action is the **same as her parent HyperStore user's permission scope** (as identified in the table below).
- The IAM Service **will not allow an IAM user to execute an administrative action that her parent HyperStore user is not allowed to execute**, even if an IAM policy grants the IAM user permission to the action.

**Note** For an overview of this feature see **"IAM Extensions for Role-Based Access to HyperStore Admin Functions"** (page 1027).

The table below lists the administrative Action permissions supported by the HyperStore IAM Service, and indicates how, if an IAM user is granted permission to an Action, the IAM Service restricts the IAM user's use of the Action according to the role of the parent HyperStore user. Note that -- when specified as an "Action" in a policy document -- all HyperStore administrative actions are prefixed by "admin: " (analogous to how S3 actions are prefixed by "s3: "). For examples of policy documents granting HyperStore administrative permissions see below the table.

IAM Action Permission	IAM User's Permission Scope Based On Parent User's Role		
	System Admin	Group Admin	Regular User
admin:GetCloudianBill	Get any user's bill	Get bill of any user in own group	Get parent user's bill
admin:GetCloudianGroup	Get any group's profile	Get own group's profile	Not allowed
admin:GetCloudianGroupList	Get list of groups	Not allowed	Not allowed
admin:GetCloudianMonitorEvents	Get event list for a node	Not allowed	Not allowed
admin:GetCloudianMonitorNodelist	Get list of monitored nodes	Not allowed	Not allowed
admin:GetCloudianMonitorHost	Get monitoring stats for a node	Not allowed	Not allowed
admin:GetCloudianMonitorRegion	Get monitoring stats for a region	Not allowed	Not allowed
admin:GetCloudianQoSLimits	Get QoS limits for any group or user	Get QoS limits for own group or users in own group	Get parent user's QoS limits

IAM Action Permission	IAM User's Permission Scope Based On Parent User's Role		
	System Admin	Group Admin	Regular User
admin:GetCloudianSystemLicense	Get system license info	Not allowed	Not allowed
admin:GetCloudianSystemVersion	Get current system version	Get current system version	Get current system version
admin:GetCloudianUsage	Get usage info for any group or user	Get usage info for own group or users in own group	Get parent user's usage info
admin:GetCloudianUser	Get any user's profile	Get profile of any user in own group	Get parent user's profile
admin:GetCloudianUserCredentials	Get any user's S3 credential	Get S3 credential of any user in own group	Get parent user's S3 credential
admin:GetCloudianUserCredentialsList	Get any user's S3 credentials list	Get S3 credentials list of any user in own group	Get parent user's S3 credentials list
admin:GetCloudianUserCredentialsListActive	Get any user's active S3 credentials list	Get active S3 credentials list of any user in own group	Get parent user's active S3 credentials list
admin:GetCloudianUserList	Get list of users in any group	Get list of users in own group	Not allowed

**Note** When a HyperStore regular user grants his IAM users administrative action permissions that are allowed to a regular user -- such as "GetCloudianUsage" or "GetCloudianQosLimits" -- this gives the IAM users permission to perform those actions in regard to the **parent user's account**. HyperStore does not track usage, billing, or QoS information specifically for IAM users. This information is only tracked for the parent HyperStore user accounts.

Below is an example of a simple IAM policy document for HyperStore administrative permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "admin:GetCloudianBill",
    "Resource": "*"
  }]
}
```

**Note** You must include the "Resource" element and set it to "\*". This is because Resource is a required element in IAM policy document syntax.

## 14.4. IAM Extensions for Role-Based Access to HyperStore Admin Functions

*Subjects covered in this section:*

- *Introduction (immediately below)*
- **"Comparing the Admin API to the IAM API with RBAC Extensions"** (page 1027)
- **"Administrative Actions Supported by the IAM API"** (page 1028)
- **"Giving Administrative Action Privileges to IAM Users"** (page 1030)
- **"Using admin\_client.py to Call the IAM Service Extensions for Administrative Actions"** (page 1031)

The HyperStore IAM Service supports extensions to the IAM API that allow for role-based access control (RBAC) for read-only HyperStore administrative functions. The extensions take the form of additions to the list of valid values that can be specified by the "Action" request parameter in a request to the IAM Service. The supported Actions vary by the role of the requester: the IAM Service allows a HyperStore system administrator to execute a wider range of Actions than can a group administrator or a regular user.

### 14.4.1. Comparing the Admin API to the IAM API with RBAC Extensions

In HyperStore 7.0.x and older, the Admin API was the only administrative API for the system. In HyperStore 7.1 and newer, certain administrative functions can also be called through HyperStore's implementation of the IAM API. The table below compares the HyperStore Admin API to the HyperStore IAM API with its extensions for admin actions.

Admin API	IAM API with RBAC Extensions for Admin Actions
<p>Implemented by the HyperStore Admin Service</p> <ul style="list-style-type: none"> <li>• Enabled by default</li> <li>• Runs on each node in each of your service regions (but with limited functionality in regions other than the default region)</li> <li>• Listens on ports 19443 (HTTPS) and 18081 (HTTP, optional)</li> <li>• Includes a bundled self-signed certificate for HTTPS</li> <li>• Request authentication is by HTTP Basic Authentication</li> <li>• Should only be exposed to internal traffic, not user traffic</li> <li>• Makes no distinctions based on role of the requester -- all access is system administrator level access</li> </ul>	<p>Implemented by the HyperStore IAM Service</p> <ul style="list-style-type: none"> <li>• Disabled by default</li> <li>• If enabled, runs on each node in your default region only</li> <li>• Listens on ports 16443 (HTTPS) and 16080 (HTTP)</li> <li>• Includes a bundled self-signed certificate for HTTPS</li> <li>• Request authentication is by Amazon-compliant Signature v2 or v4</li> <li>• Can be exposed to user</li> </ul>

Admin API	IAM API with RBAC Extensions for Admin Actions
	<p>traffic</p> <ul style="list-style-type: none"> <li>Makes distinctions based on the role of the requester -- system administrators have a greater permissions scope than group administrators, who have a greater permission scope than regular users (role-based access control)</li> </ul>
<p>Proprietary RESTful API</p> <ul style="list-style-type: none"> <li>GET, PUT, POST, and DELETE are all supported, and are different operations with different consequences</li> <li>Request parameters are in lower camel case -- for example "canonicalUserId" and "billingPeriod"</li> <li>Response bodies are JSON formatted</li> </ul>	<p>Compliant with Amazon's IAM API</p> <ul style="list-style-type: none"> <li>Only GET and POST are supported and it doesn't matter which you use (what matters is the "Action" parameter)</li> <li>Request parameters are in upper camel case (Pascal case) -- for example "CanonicalUserId" and "BillingPeriod"</li> <li>Response bodies are XML formatted</li> </ul>
<p>Wide range of administrative tasks</p> <p>The Admin API supports more than 80 different methods for retrieving information about or making changes to the system</p>	<p>Narrow range of administrative tasks</p> <p>The IAM API extensions currently support only 16 administrative actions and these are all read-only (none of the supported actions make changes to the system)</p>

#### 14.4.2. Administrative Actions Supported by the IAM API

The table below lists the administrative Actions supported by the HyperStore IAM Service, and how the IAM Service restricts the use and implementation of these Actions according to the role (user account type) of the requester.

Also as shown by the table, each administrative Action supported by the IAM Service corresponds to an existing method in the Admin API -- in the sense that the IAM Action supports the same request parameters as the corresponding Admin API method (except the IAM version uses upper camel case for parameter names rather than lower camel case) and returns the same response body elements as the corresponding Admin API method (except the IAM version uses XML formatting for the response body rather than JSON). Consequently, for details about the request parameters and response body associated with a particular administrative IAM Action you can check the documentation of the corresponding Admin API method.

IAM Action	Permission Scope Based On Requester's Role			Corresponding Admin API Method
	System Admin	Group Admin	Regular User	
GetCloudianBill (see Important note below table)	Get any user's bill	Get bill of any user in own group	Get own bill	<a href="#">GET /billing</a>
GetCloudianGroup	Get any group's profile	Get own group's profile	Not allowed	<a href="#">GET /group</a>
GetCloudianGroupList	Get list of groups	Not allowed	Not allowed	<a href="#">GET /group/list</a>
GetCloudianMonitorEvents	Get event list for a node	Not allowed	Not allowed	<a href="#">GET /monitor/events</a>
GetCloudianMonitorNodelist	Get list of monitored nodes	Not allowed	Not allowed	<a href="#">GET /monitor/nodelist</a>
GetCloudianMonitorHost	Get monitoring stats for a node	Not allowed	Not allowed	<a href="#">GET /monitor/host</a>
GetCloudianMonitorRegion	Get monitoring stats for a region	Not allowed	Not allowed	<a href="#">GET /monitor</a>
GetCloudianQoSLimits	Get QoS limits for any group or user	Get QoS limits for own group or users in own group	Get own QoS limits	<a href="#">GET /qos/limits</a>
GetCloudianSystemLicense	Get system license info	Not allowed	Not allowed	<a href="#">GET /system/license</a>
GetCloudianSystemVersion	Get current system version	Get current system version	Get current system version	<a href="#">GET /system/version</a>
GetCloudianUsage	Get usage info for any group or user	Get usage info for own group or users in own group	Get own usage info	<a href="#">GET /usage</a>
GetCloudianUser	Get any user's profile	Get profile of any user in own group	Get own profile	<a href="#">GET /user</a>
GetCloudianUserCredentials	Get any user's S3 credential	Get S3 credential of any user in	Get own S3 credential	<a href="#">GET /user/credentials</a>

IAM Action	Permission Scope Based On Requester's Role			Corresponding Admin API Method
	System Admin	Group Admin	Regular User	
		own group		
GetCloudianUserCredentialsList	Get any user's S3 credentials list	Get S3 credentials list of any user in own group	Get own S3 credentials list	<a href="#">GET /user/credentials/list</a>
GetCloudianUserCredentialsListActive	Get any user's active S3 credentials list	Get active S3 credentials list of any user in own group	Get own active S3 credentials list	<a href="#">GET /user-credentials/list/active</a>
GetCloudianUserList	Get list of users in any group	Get list of users in own group	Not allowed	<a href="#">GET /user/list</a>

**IMPORTANT !** Before the "GetCloudianBill" Action can be used to retrieve billing data for a specified user and billing period, you must either execute the Admin API method [POST /billing](#) to generate billing data for that user and billing period, or else use the CMC's [Account Activity](#) page to generate billing data for that user and billing period. There is currently no RBAC version of the *POST /billing* call that generates user billing data.

### 14.4.3. Giving Administrative Action Privileges to IAM Users

Just as HyperStore users can use IAM policies to grant S3 action permissions to their IAM users, so too can HyperStore users use IAM policies to grant HyperStore admin permissions to their IAM users. A typical use case would be if a HyperStore system administrator wanted to create an IAM user who is allowed to perform some of the system admin Actions but not all of them.

At a high level this feature works as follows:

- As is the case with S3 permissions, an **IAM user by default has no admin permissions** -- an IAM user gains permissions only if she is assigned an IAM policy that specifies those permissions, and she gains only the permissions specified in the policy.
- When an IAM policy grants an IAM user permission to an administrative action, the IAM user's permission scope in respect to that action is the **same as her parent HyperStore user's permission scope** (as identified in the table above). For example:
  - If an IAM user is granted permission to the "GetCloudianGroup" action and her parent HyperStore user is a system administrator, the IAM user can get any HyperStore group's profile.
  - If an IAM user is granted permission to the "GetCloudianGroup" action and her parent HyperStore user is a group administrator, the IAM user can (only) get that HyperStore group's profile.
  - If an IAM user is granted permission to the "GetCloudianGroup" action and her parent HyperStore user is a regular user, the IAM Service will reject the IAM user's attempt to get any group

profile. The IAM Service **will not allow an IAM user to execute an administrative action that her parent HyperStore user is not allowed to execute.**

**Note** When a HyperStore regular user grants his IAM users administrative action permissions that are allowed to a regular user -- such as "GetCloudianUsage" or "GetCloudianQoSLimits" -- this gives the IAM users permission to perform those actions in regard to the **parent user's account**. For example an IAM user granted permission to the "GetCloudianUsage" action would be able to get usage information for the parent user account; and if granted permission to "GetCloudianQoSLimits" would be able to get the QoS limits associated with the parent user account. HyperStore does not track usage, billing, or QoS information specifically for IAM users. This information is only tracked for the parent HyperStore user accounts.

For more information on using IAM policies to grant admin permissions to IAM users, see:

- **"Supported IAM Policy Elements"** (page 1023) (for the HyperStore IAM Service's support of policy document elements)
- **"Manage IAM Policy"** (page 303) (for the CMC's support for creating IAM policies)

**Note** The CMC provides two tools for creating IAM policies -- a Visual Editor and a JSON Editor. The Visual Editor does not support creating a policy that contains HyperStore admin permissions, but the JSON Editor does.

#### 14.4.4. Using *admin\_client.py* to Call the IAM Service Extensions for Administrative Actions

In the current HyperStore release, the CMC's built-in IAM client does not support calling the IAM extensions for HyperStore administrative Actions. To perform administrative Actions through the HyperStore IAM Service you can either:

- Use a third party tool that you customize to be able to support the HyperStore admin Action strings (as listed in the table above) and their associated request parameters (detailed in the corresponding Admin API links provided for each action in the table).
- Use the Python tool *admin\_client.py* that comes bundled with HyperStore version 7.1 and later, as described below.

*If you are using the HyperStore Shell*

The [HyperStore Shell \(HSH\)](#) does not support using the *admin\_client.py* tool.

**Note** To use the *admin\_client.py* tool you will need to supply the tool with S3 access credentials. These can be your own credentials (see **"Creating S3 Access Credentials for the Default System Admin User"** (page 993) if applicable) or those of an IAM user to whom you have granted administrative action permissions in an IAM policy. .

HyperStore includes an interactive tool (written in Python) that makes it easy to call the administrative Actions that the HyperStore IAM Service supports. The tool is located in the following directory on each HyperStore node:

```
/opt/cloudian/tools
```

To launch the tool:

```
# ./admin_client.py
```

The first time that the tool is launched on a node, the tool automatically downloads and installs the required Python packages if they are not already present on the node (this requires outbound internet access from the node, in order for the tool to download the packages).

Once this completes, the tool's main menu displays:



```
RBAC API Client Main Menu

1. Setup Host and Credentials
2. Billing API Requests
3. Group API Requests
4. Monitoring API Requests
5. QOS API Requests
6. System API Requests
7. Usage API Requests
8. User API Requests
9. Quit

>>> █
```

Use option 1 to supply the tool with your S3 access credentials and to specify the host information (you can connect to any HyperStore host) and the region in which the host resides.

You can then perform admin Actions by choosing from the menus. For each request type the tool will prompt you to provide the needed parameter values (if any). The request response will display in the tool interface, in XML format.

It may be helpful to have the HyperStore Help open as you use the tool -- specifically the Admin API section of the Help. If needed you can check the documentation for the corresponding Admin API call as you provide the information required for a given request type. For example if you're using the tool to call the "GetCloudianBill" request and the tool prompts you for the "BillingPeriod", and you're not sure of the proper format for billing period -- you can check the Help for *GET /bill* to get this information. See **"Administrative Actions Supported by the IAM API"** (page 1028) to see which Admin API methods correspond to the administrative Actions that the IAM Service supports.

**Note** Although the CMC's built-in IAM client does not currently support calling the admin Actions, the CMC does support creating an IAM user, assigning that user to an IAM group, and creating an inline policy for that group which includes admin Action permissions. The IAM user will then have those admin Action permissions. However, the IAM user will not be able to execute those admin Actions through the CMC -- he would need to use the Python tool, or a third party IAM client that's been customized to support the IAM extensions.

## 14.5. SAML Support

HyperStore supports Security Assertion Markup Language (SAML 2.0) based access to S3 storage resources. It does so by supporting the standard AWS IAM Service calls and Security Token Service calls that are needed to set up and execute SAML based access. With SAML, federated users -- users who have been authenticated

by a trusted identity provider system (IdP) external to HyperStore -- can be granted temporary access to HyperStore S3 resources, subject to configurable permission restrictions.

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

- **"Downloading the HyperStore SAML Metadata Document for IdP Setup"** (page 1033)
- **"Using the IAM Service to Create and Manage SAML Provider Resources"** (page 1033)
- **"Using the IAM Service to Create and Manage Roles"** (page 1034)
- **"Using the STS Service to Assume a Role"** (page 1034)

### 14.5.1. Downloading the HyperStore SAML Metadata Document for IdP Setup

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

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

For example:

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

**If you have configured your load balancers so that external access to the CMC is through a different port number** than the CMC is listening on internally (which is 8443 by default), then before downloading the HyperStore SAML Provider Metadata document run the following commands on your Puppet master node:

```
hsctl config set cmc.ports.loadBalancer.https=<load balancer port for CMC>
hsctl config apply saml
```

For example:

```
hsctl config set cmc.ports.loadBalancer.https=443
hsctl config apply saml
```

This will result in the correct CMC external access port being specified within the Service Provider Metadata document.

### 14.5.2. Using the IAM Service to Create and Manage SAML Provider Resources

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

- [CreateSAMLProvider](#)
- [ListSAMLProviders](#)
- [GetSAMLProvider](#)
- [UpdateSAMLProvider](#)
- [DeleteSAMLProvider](#)

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

#### 14.5.2.1. Limitations

In the current HyperStore release, the CMC does not support the IAM calls for creating and managing SAML provider resources. You must use a third party or custom IAM client application to execute these calls to the HyperStore IAM Service.

### 14.5.3. Using the IAM Service to Create and Manage Roles

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

The relevant IAM calls are:

- [CreateRole](#)
- [ListRoles](#)
- [GetRole](#)
- [UpdateRole](#)
- [UpdateAssumRolePolicy](#)
- [DeleteRole](#)
- [AttachRolePolicy](#)
- [ListAttachedRolePolicies](#)
- [DetachRolePolicy](#)
- [PutRolePolicy](#)
- [ListRolePolicies](#)
- [GetRolePolicy](#)
- [DeleteRolePolicy](#)

#### 14.5.3.1. Limitations

- In the current HyperStore release, the CMC does not support the IAM calls for creating and managing roles. You must use a third party or custom IAM client application to execute these calls to the HyperStore IAM Service.
- Role session policies and role tags are not supported.
- Support for Conditions in trust policies is limited; see [CreateRole](#).

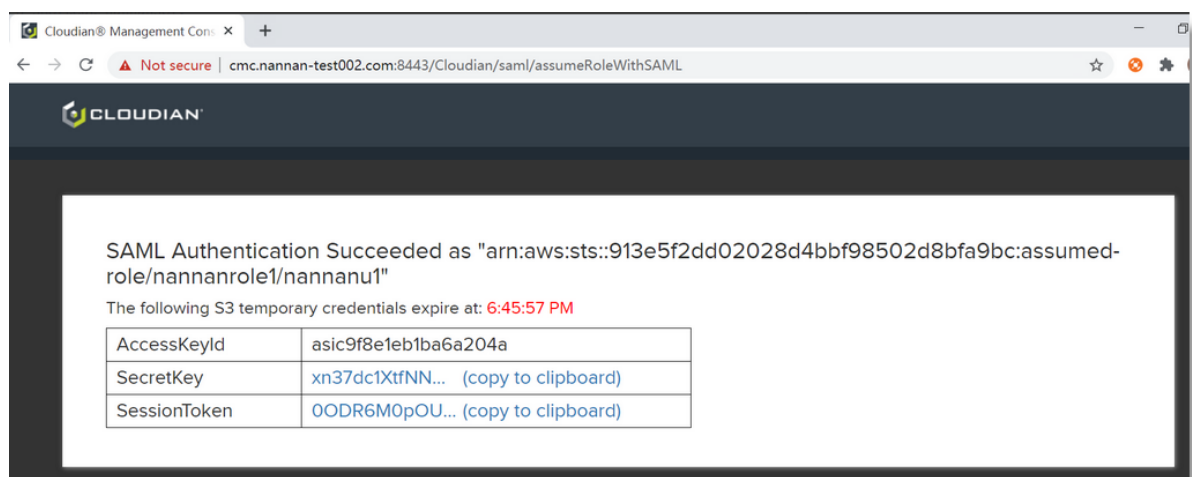
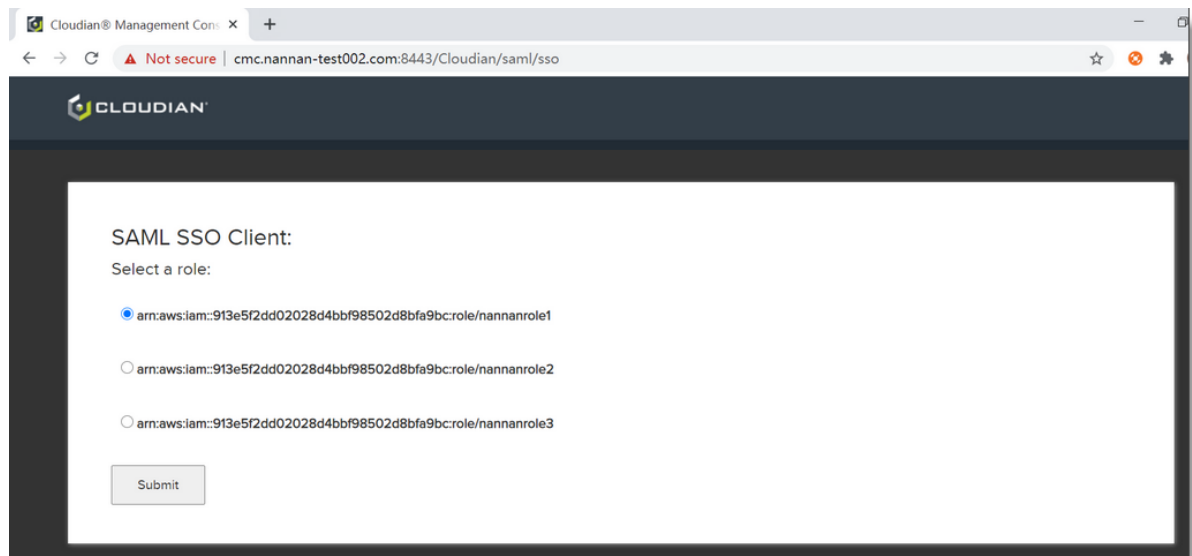
### 14.5.4. Using the STS Service to Assume a Role

Once an IdP has been configured with HyperStore's SAML metadata XML document, and you have created a SAML provider IAM resource for that IdP and specified that provider as part of an IAM role's trust policy, SAML assertions from that provider can be used to allow federated users to assume that role by calling the Hyper-

Store Security Token Service's [AssumeRoleWithSAML](#) API. This API initiates a role session during which properly authenticated and authorized users are provided with temporary S3 security credentials.

The HyperStore STS Service's *AssumeRoleWithSAML* API can be called by a third party or custom STS client application.

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



**Note** Users who obtain temporary security credentials, either through the CMC as described above or through a third party client application's calls directly to the HyperStore STS Service, will **not** be able to log in to the CMC to use it as their S3 client application. They must use a third party S3 client application to perform S3 operations on the HyperStore storage system.

#### 14.5.4.1. Limitations

In the current HyperStore release, users with temporary security credentials can only access the HyperStore S3 Service, and only by using a third party or custom S3 client application. Users with temporary security credentials cannot log into the CMC, and they cannot access the HyperStore IAM Service (even with a third party or customer IAM client application).

[[can't log into CMC]]

# Chapter 15. STS API

## 15.1. Introduction

### 15.1.1. HyperStore Support for the AWS STS API

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

- Only a few STS Actions are supported -- for details see Section 15.2 "Supported STS Actions".
- HyperStore does not allow users with temporary security credentials (obtained through the STS Service) to perform [IAM operations](#). The HyperStore IAM Service will reject requests that contain temporary credentials. Users with temporary credentials can only access the S3 Service (within the permission restrictions of the roles that such users assume).
- Users with temporary security credentials are not allowed to log into the CMC.

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

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

**Note** Be sure to [configure the STS endpoint domain in your DNS environment](#).

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

### 15.1.2. STS Common Request Parameters

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

- Action
- Version
- X-Amz-Algorithm
- X-Amz-Credential
- X-Amz-Date
- X-Amz-Security-Token (only supported for [GetCallerIdentity](#) requests)
- X-Amz-Signature
- X-Amz-SignedHeaders

### 15.1.3. STS Common Errors

From the ["Common Errors" section](#) of the AWS STS API specification, HyperStore supports the parameters listed below. If a common parameter from that specification section is not listed below, HyperStore does not

support it.

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

## 15.2. Supported STS Actions

### 15.2.1. AssumeRole

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

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

For action details and examples see the AWS documentation: [AssumeRole](#)

**Note** HyperStore does not allow users with temporary security credentials to perform [IAM operations](#).

#### 15.2.1.1. Request Parameters

- DurationSeconds
- ExternalId
- RoleArn
- RoleSessionName

#### 15.2.1.2. Response Elements

- AssumedRoleUser
- Credentials

#### 15.2.1.3. Errors

- InvalidParameterValue
- NoSuchEntity

## 15.2.2. AssumeRoleWithSAML

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

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

For action details and examples see the AWS documentation: [AssumeRoleWithSAML](#)

**Note** HyperStore does not allow users with temporary security credentials to perform [IAM operations](#).

### 15.2.2.1. Request Parameters

- DurationSeconds
- PrincipalArn
- RoleArn
- SAMLAssertion

### 15.2.2.2. Response Elements

- AssumedRoleUser
- Audience
- Credentials
- Issuer
- NameQualifier
- Subject
- SubjectType

### 15.2.2.3. Errors

- ExpiredToken
- InvalidClientTokenId
- InvalidParameterValue

## 15.2.3. GetCallerIdentity

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

HyperStore supports the elements listed below.

For action details and examples see the AWS documentation: [GetCallerIdentity](#)

### 15.2.3.1. Response Elements

- Account
- Arn

- UserId

#### 15.2.3.2. Errors

- ExpiredToken
- InvalidClientTokenId

# Chapter 16. SQS API

## 16.1. HyperStore Support for the AWS SQS API

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

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

- A third party SQS client application must be used to interface with the HyperStore SQS Service to perform operations such as creating and configuring queues and receiving and deleting queued messages. The CMC does not yet support SQS operations.
- A third party S3 client application must be used to execute the [PUT Bucket Notification](#) operation. The CMC does not yet support this S3 operation.
- For bucket notifications to an SQS queue to work, the bucket owner must also be the owner of the SQS queue.
- HTTPS access to the SQS Service is not supported. Only regular HTTP access is supported.
- The HyperStore SQS Service supports many of the Actions from the Amazon SQS API, but not all of them. For more detail see "**SQS Supported Actions**" (page 1042).

### 16.1.1. Enabling the Bucket Notification Feature and the SQS Service

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

1. In [common.csv](#) set *sqs\_enabled* to *true*.

**Note** Also in *common.csv* you can optionally edit the *sqs\_endpoint* setting, if you wish to have an SQS Service endpoint different than the default SQS Service endpoint. The default SQS Service endpoint is *s3-sqs.<your-domain>*.

2. In [mts.properties.erb](#):

- Edit the *cloudian.s3.unsupported* property to remove *notification* from the list of unsupported S3 request types. Be sure to delete the preceding comma as well.

Before your edit:

```
cloudian.s3.unsupported=accelerate,requestPayment,analytics,inventory,metrics,select,notification
```

After your edit:

```
cloudian.s3.unsupported=accelerate,requestPayment,analytics,inventory,metrics,select
```

- Below the `cloudian.s3.unsupported` property, **add this new property** to the file (it is not in the file by default):

```
cloudian.s3.bucketnotification=true
```

- Use the installer to [push the configuration changes to the cluster and restart the S3 Service and the SQS Service](#).

Once you have enabled the bucket notification feature and the SQS Service, then:

- A third party SQS client application can be used to submit requests to the HyperStore SQS Service, such as for creating and configuring a queue. For HyperStore support of SQS Actions see **"SQS Supported Actions"** (page 1042). The default SQS Service endpoint URL is `http://s3-sqs.<domain>:18090`
- A third party S3 client application can be used to submit a *PUT Bucket Notification* request to the HyperStore S3 Service, to configure notifications for an existing bucket. For HyperStore support of this S3 API method see [PUT Bucket Notification](#). As noted previously, the **bucket owner must also be the SQS queue owner**.

**Note** Information about requests processed by the SQS Service are logged to `cloudian-sqs-request-log`, which exists on each node. For more information see **"S3 Service Logs (including Auto-Tiering, CRR, and WORM)"** (page 619).

## 16.2. SQS Supported Actions

When enabled, the HyperStore SQS Service supports the following Actions from the Amazon Simple Queue Service API. If an SQS Action is not listed here, HyperStore does not support it. For descriptions of each Action and its associated parameters, elements, and errors, and for examples, see the AWS documentation links.

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

Supported Action	AWS Documentation
ChangeMessageVisibility	<a href="#">ChangeMessageVisibility</a>
CreateQueue	<a href="#">CreateQueue</a>
HyperStore does not currently support these queue attributes and will ignore them if included in the CreateQueue request:	
<ul style="list-style-type: none"> <li>Policy</li> <li>RedrivePolicy</li> <li>KmsMasterKeyId</li> <li>KmsDataKeyReusePeriodSeconds</li> <li>FifoQueue</li> <li>ContentBasedDeduplication</li> </ul>	
DeleteMessage	
DeleteQueue	
GetQueueAttributes	<a href="#">GetQueueAttributes</a>

Supported Action	AWS Documentation
GetQueueUrl	<a href="#">GetQueueUrl</a>
ListQueues	<a href="#">ListQueues</a>
PurgeQueue	<a href="#">PurgeQueue</a>
ReceiveMessage	<a href="#">ReceiveMessage</a>
SendMessage  The HyperStore S3 Service uses the SendMessage action to publish notification messages to a queue. The SendMessage action is not intended to be used by external SQS clients.  HyperStore does not currently support message attributes.	<a href="#">SendMessage</a>
SetQueueAttributes  See CreateQueue for a list of queue attributes that HyperStore does not currently support.	<a href="#">SetQueueAttributes</a>

This page left intentionally blank

# Chapter 17. Open Source License Agreements

Clouddian, Inc. acknowledges the redistribution of open source components under the licenses shown below.

Component or Library	License	License URL	Copyright
Airlift	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright 2011 Dain Sundstrom dain@iq80.com Copyright 2010 Cedric Beust cedric@beust.com
Amazon S3 SDK	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright 2010-2014 Amazon.com, Inc. or its affiliates.
Antlr	BSD	<a href="http://wwwantlr.org/license.html">http://wwwantlr.org/license.html</a>	Copyright (c) 2012 Terence Parr and Sam Harwell
Apache Commons	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright © 2018 The Apache Software Foundation.
Apache HTTPComponents	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright © 2005-2018 The Apache Software Foundation
Apache Tomcat	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright © 1999-2018, The Apache Software Foundation
Apache Velocity	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright © 2005-2018 The Apache Software Foundation
Avro	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright © 2012 The Apache Software Foundation."
Blueimp	MIT	<a href="https://opensource.org/licenses/MIT">https://opensource.org/licenses/MIT</a>	Copyright © 2010 Sebastian Tschan, <a href="https://blueimp.net">https://blueimp.net</a>
Bootstrap	MIT	<a href="https://opensource.org/licenses/MIT">https://opensource.org/licenses/MIT</a>	Copyright (c) 2011-2018 Twitter, Inc. Copyright (c) 2011-2018 The Bootstrap Authors
Cassandra	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright 2009-2014 The Apache Software Foundation
CentOS	GPL and various	<a href="http://mirror.centos.org/centos/6/os/i386/EULA">http://mirror.centos.org/centos/6/os/i386/EULA</a>	Copyright © 2017 The CentOS Project
D3	BSD	<a href="https://opensource.org/licenses/BSD-3-Clause">https://opensource.org/licenses/BSD-3-Clause</a>	Copyright 2010-2017 Mike Bostock
DataStax Java Driver	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright 2012-2018, DataStax
DataTables	MIT	<a href="https://opensource.org/licenses/MIT">https://opensource.org/licenses/MIT</a>	Copyright (C) 2008-2018, SpryMedia Ltd.
Disruptor	Apache	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	None

Component or Library	License	License URL	Copyright
	2.0	2.0	
DropWizard	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	None
Gson	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright 2008 Google Inc.
Guava	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	None
Hector	MIT	<a href="https://opensource.org/licenses/MIT">https://opensource.org/licenses/MIT</a>	Copyright (c) 2010 Ran Tavory
High-scale-lib	Public Domain	<a href="https://github.com/stephenc/high-scale-lib/blob/master/LICENSE">https://github.com/stephenc/high-scale-lib/blob/master/LICENSE</a>	None
Jackson	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	None
Java	Oracle binary code license	<a href="http://www.oracle.com/technetwork/java/javase/terms/license/index.html">http://www.oracle.com/technetwork/java/javase/terms/license/index.html</a>	Copyright © 1995, 2018, Oracle and/or its affiliates.
JCraft	BSD	<a href="http://www.jcraft.com/jsch">http://www.jcraft.com/jsch</a>	Copyright (c) 2002-2015 Atsuhiko Yamanaka, JCraft, Inc.
Jedis	Custom: No limitation if copyright included	<a href="https://github.com/xetorthio/jedis/blob/master/LICENSE.txt">https://github.com/xetorthio/jedis/blob/master/LICENSE.txt</a>	Copyright (c) 2010 Jonathan Leibusky
Jersey	CDDL v1.1	<a href="https://jersey.java.net/license.html">https://jersey.java.net/license.html</a>	Copyright ©2010-2017 Oracle Corporation.
Jetty	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright © 2016 The Eclipse Foundation.
JNA	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	None
Joda-Time	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright ©2002-2017 Joda.org.
Jquery	MIT	<a href="https://opensource.org/licenses/MIT">https://opensource.org/licenses/MIT</a>	Copyright JS Foundation and other contributors, <a href="https://js.foundation/">https://js.-foundation/</a>
jsviews	MIT	<a href="https://opensource.org/licenses/MIT">https://opensource.org/licenses/MIT</a>	Copyright (c) 2015 Boris Moore, <a href="https://github.com/BorisMoore/jsviews">https://github.com/BorisMoore/jsviews</a>
JYaml	BSD	<a href="http://jyaml.sourceforge.net/license.html">http://jyaml.sourceforge.net/license.html</a>	None
log4j	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright © 1999-2018 The Apache Software Foundation.

Component or Library	License	License URL	Copyright
LZ4	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	None
Netty	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	None
OpenCSV	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	None
Paranamer	BSD	<a href="https://github.com/paul-hammant/paranamer/blob/master/LICENSE.txt">https://github.com/paul-hammant/paranamer/blob/master/LICENSE.txt</a>	Copyright (c) 2006 Paul Hammant & ThoughtWorks Inc
Puppet	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright (C) 2005-2016 Puppet, Inc.
Redis	3-clause BSD	<a href="http://redis.io/topics/license">http://redis.io/topics/license</a>	Copyright (c) 2006-2015, Salvatore Sanfilippo
RocksDB	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright (c) 2011 The LevelDB Authors.
SLF4J	MIT	<a href="https://opensource.org/licenses/MIT">https://opensource.org/licenses/MIT</a>	Copyright (c) 2004-2017 QOS.ch
SnakeYaml	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	None
Snappy	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	None
SNMP4J	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright © 2003-2018, SNMP4J.org
Spring	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright (c) 2013 GoPivotal, Inc. Copyright (c) 2000-2011 INRIA, France Telecom Copyright (c) 1999-2009, OW2 Consortium < <a href="http://www.ow2.org/">http://www.ow2.org/</a> >
Thrift	Apache 2.0	<a href="http://www.apache.org/licenses/LICENSE-2.0">http://www.apache.org/licenses/LICENSE-2.0</a>	Copyright © 2005-2018 The Apache Software Foundation
UUID	MIT	<a href="https://opensource.org/licenses/MIT">https://opensource.org/licenses/MIT</a>	Copyright © 2003-2013 Johann Burkard