Copyright © 2018 Nexenta Systems™, ALL RIGHTS RESERVED

Notice: No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or stored in a database or retrieval system for any purpose, without the express written permission of Nexenta Systems (hereinafter referred to as “Nexenta”).

Nexenta reserves the right to make changes to this document at any time without notice and assumes no responsibility for its use. Nexenta products and services only can be ordered under the terms and conditions of Nexenta Systems’ applicable agreements. All of the features described in this document may not be available currently. Refer to the latest product announcement or contact your local Nexenta Systems sales office for information on feature and product availability. This document includes the latest information available at the time of publication.

Nexenta, NexentaStor, NexentaEdge, NexentaFusion, and NexentaConnect are registered trademarks of Nexenta Systems in the United States and other countries. All other trademarks, service marks, and company names in this document are properties of their respective owners.

This document applies to the following product versions:

<table>
<thead>
<tr>
<th>Product</th>
<th>Versions supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>NexentaEdge™</td>
<td>2.1.3</td>
</tr>
<tr>
<td>Ubuntu Linux</td>
<td>Ubuntu 16.04.4 LTS</td>
</tr>
<tr>
<td>CentOS</td>
<td>7.4 with ELRepo LT kernel only</td>
</tr>
<tr>
<td>Red Hat Enterprise Linux</td>
<td></td>
</tr>
<tr>
<td>Chef</td>
<td>12.x</td>
</tr>
<tr>
<td>OpenStack Swift</td>
<td>Newton, Ocata, Pike</td>
</tr>
<tr>
<td>OpenStack Cinder</td>
<td></td>
</tr>
<tr>
<td>Docker</td>
<td>1.13</td>
</tr>
<tr>
<td>S3cmd tool</td>
<td>1.6.1</td>
</tr>
</tbody>
</table>
## Contents

1. **Introduction to NexentaEdge** .............................. 1
   - About NexentaEdge ........................................... 1
   - Components of a NexentaEdge Cluster ................. 3
   - About This User Guide ................................ 6

2. **Managing Servers and Storage Devices** ................. 7
   - Managing Servers ........................................... 7
   - Managing Devices ......................................... 14

3. **Administering the NexentaEdge System** ................. 20
   - Displaying Cluster Status ................................ 20
   - Displaying Statistics for the Replicast Network .... 24
   - Configuring Zoning for Data Nodes .................... 26
   - Initializing the Cluster .................................. 27
   - Configuring Autosupport .................................. 27
   - Enabling SNMP Traps ....................................... 30

4. **Managing Logical Clusters** ................................ 31
   - Overview of Logical Cluster Administration ........ 31
   - Managing Clusters ......................................... 32
   - Managing Tenants ......................................... 35
   - Managing Buckets ............................................. 40

5. **Configuring an iSCSI Storage Service Group** ............ 44
   - Overview ..................................................... 44
   - Before You Begin .......................................... 44
   - Setting Up an iSCSI Storage Service Group .......... 45
   - Setting Resource Limits for the iSCSI Storage Service Group ........................................... 48
   - Setting iSCSI Target Parameters ....................... 50
   - Configuring iSCSI Portals ................................ 50
   - Configuring CHAP Authentication for Initiators .. 51
   - Managing LUNs .............................................. 51
Managing Snapshots .......................................................... 54
Configuring an iSCSI Storage Service Group for High Availability ........................................ 56
Managing NBD Devices ....................................................... 60

6 Configuring an OpenStack Swift Storage Service Group ......................................................... 62
Overview ........................................................................... 62
Before You Begin ............................................................... 62
Setting Up an OpenStack Swift Storage Service Group ......................................................... 63
Configuring Authentication for the OpenStack Swift Storage Service Group .................. 65
Changing the Ports for OpenStack Swift API Calls ................................................................. 68
Configuring the OpenStack Swift Gateway to Provide MD5 Checksum Values for Objects .. 69
Enabling Versioning Support on the OpenStack Swift Gateway ......................................... 71
OpenStack Swift Large Objects Support .............................................................................. 72
Setting Resource Limits for the OpenStack Swift Storage Service Group ......................... 72

7 Configuring an Amazon S3 Storage Service Group ................................................................. 75
Overview ........................................................................... 75
Before You Begin ............................................................... 75
Setting Up an Amazon S3 Storage Service Group ................................................................. 76
Configuring Authentication for the Amazon S3 Storage Service Group ............................ 79
Changing the Ports for Amazon S3 API Calls ...................................................................... 83
Changing the Region Setting Used for Amazon S3 API Calls .............................................. 84
Configuring the Amazon S3 Gateway to Provide Vendor-Defined Checksum Values for Objects ................................................................. 85
Enabling Versioning Support on the Amazon S3 Gateway ................................................. 86
Amazon S3 Multipart Upload Support .................................................................................. 87
Setting Resource Limits for the Amazon S3 Storage Service Group ................................... 88

8 Configuring an NFS Storage Service Group .......................................................... 91
Overview ........................................................................... 91
Before You Begin ............................................................... 91
Setting Up an NFS Storage Service Group ........................................................................... 92
Setting Resource Limits for the NFS Storage Service Group ............................................. 96
Configuring Access Control Lists for the NFS Storage Service Group ............................. 97
9  NexentaEdge CLI Commands ....................................................... 99
    Overview ............................................................................. 99
    NEDEPLOY Commands ......................................................... 100
    NEADM Commands ............................................................... 100

10  Supported OpenStack Swift API Functions In NexentaEdge ................. 102
    Account API Functions ......................................................... 103
    Container API Functions ....................................................... 106
    Object API Functions .......................................................... 110

11  Supported Amazon S3 API Functions In NexentaEdge ............................ 116
    Service API Functions .......................................................... 117
    Bucket API Functions .......................................................... 118
    Object API Functions ........................................................... 134
This documentation presents information specific to Nexenta products. The information is for reference purposes and is subject to change.

Intended Audience

This documentation is intended for Object Storage Administrators and Network Administrators and assumes that you have a working knowledge of UNIX. The document also assumes that you have experience with data storage concepts, such as object storage, ZFS, iSCSI, NFS, CIFS, and so on.

Documentation History

The following table lists the released revisions of this documentation.

Table 1: Documentation Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000-nedge-1.0-000054-A</td>
<td>July, 2015</td>
<td>1.0 release</td>
</tr>
<tr>
<td>3000-nedge-1.1-000054-A</td>
<td>February, 2016</td>
<td>1.1 release</td>
</tr>
<tr>
<td>3000-nedge-1.1-FP1-000054-B</td>
<td>May, 2016</td>
<td>1.1 FP1 release</td>
</tr>
<tr>
<td>3000-nedge-1.1-FP2-000054-A</td>
<td>May, 2016</td>
<td>1.1 FP2 release</td>
</tr>
<tr>
<td>3000-nedge-1.1-FP3-000054-A</td>
<td>June, 2016</td>
<td>1.1 FP3 release</td>
</tr>
<tr>
<td>3000-nedge-2.0-000054-A</td>
<td>April, 2017</td>
<td>2.0 GA release</td>
</tr>
<tr>
<td>3000-nedge-2.1-000054-A</td>
<td>December, 2017</td>
<td>2.1.2 GA release</td>
</tr>
<tr>
<td>3000-nedge-2.1.3-000054-A</td>
<td>April, 2018</td>
<td>2.1.3 release</td>
</tr>
</tbody>
</table>

Comments

Your comments and suggestions to improve this documentation are greatly appreciated. Send any feedback to doc.comments@nexenta.com and include the documentation title, number, and revision. Refer to specific pages, sections, and paragraphs whenever possible.
Introduction to NexentaEdge

This chapter includes the following topics:

- About NexentaEdge
- Components of a NexentaEdge Cluster
- About This User Guide

About NexentaEdge

NexentaEdge is Nexenta Systems’ software-defined object storage system. It is designed to allow enterprises to store large amounts of object data securely, read and write that data efficiently, and add or remove storage capacity easily and inexpensively. It is designed to be flexible, with support for the following protocols:

- Object protocols: OpenStack Swift, Amazon S3
- Block protocols: iSCSI, OpenStack Cinder, Network Block Device (NBD)
- File protocol: NFS

NexentaEdge is an edge-directed object storage system because decisions about how to store and retrieve data are based on information that comes from the storage gateways and data nodes themselves (the edge of the system), as opposed to software-defined rules or services running in a central location.

About Object Storage Systems

An object storage system is a means for storing collections of data such as video files, images, and documents as objects, rather than in a file system hierarchy or as block storage. At a basic level, an object consists of data, metadata, and a unique identifier. An object storage system operates essentially as a large pool of storage for the objects, which can be stored and retrieved using PUT and GET operations. These are typically implemented as RESTful APIs.

Object storage systems are typically deployed as clusters of servers, with data and metadata distributed across the cluster. When a user stores an object, the object storage software decides how the object should be written to the servers in the cluster. The software ensures that it can retrieve the object version when a user requests it. This includes ensuring that the object version is replicated sufficient times to protect against loss of storage devices or servers. Users do not have to be concerned with explicit creation of replicas, nor with any of their locations.

Object storage systems often are designed to run using commodity hardware, so storage can be added easily and inexpensively by installing industry standard servers and disks. From an architectural standpoint, there is no upper limit to the size of an object storage system.
NexentaEdge Features

NexentaEdge offers a number of enhancements over other object storage systems. Key features of the NexentaEdge system are the Cloud Copy on Write (CCOW) storage system, Replicast communications protocol, and FlexHash dynamically updated hash allocation table.

Within the NexentaEdge cluster, these features combine to provide automatic distributed deduplication, load balancing, and compression for stored objects.

Cloud Copy on Write (CCOW)

Nexenta’s Cloud Copy on Write (CCOW) allows for secure storage and efficient retrieval of data in the cluster while minimizing the network resources required for these functions. The following are the primary benefits of Nexenta’s CCOW:

• Efficient distribution of data across the cluster
  When objects are stored in the NexentaEdge cluster, they are broken into chunks and delivered to a group of servers. Instead of copying the chunks to each server individually, CCOW multicasts them to the server group, sending the data once to multiple recipients. Transmitting data to the servers in this way saves substantial network resources over transmitting it multiple times. CCOW uses strong cryptographic hashing to ensure accurate transmission of the data.

  As you add more servers to your cluster, NexentaEdge automatically balances the workload among them.

• Optimal server selection when retrieving data
  When a user requests an object, CCOW selects servers from the multicast group that can deliver each chunk the fastest. Selecting the server from a small group greatly increases the chances of finding the shortest possible queue for the request, while still keeping the processing load for requests under control.

• Preservation of multiple versions of objects
  NexentaEdge keeps a record of the chunks that comprise each object. It uses this record to assemble an object when it is requested by a user. Each time an object is modified, a new record is created that references the new chunks and the chunks that stayed the same.

  NexentaEdge preserves the records and chunks for the current and all previous versions of the object. This allows the current and previous versions to be available to clients for rollback, clone, or GET operations. All of an object’s previous versions are retained in the cluster until they are deleted, either according to a set policy or manually by an administrator. CCOW creates a unique version identifier for each version, even during network partitions. It does not need to obtain a version number from a “master” server.

• Automatic distributed inline deduplication
  As objects are added and changed, only chunks containing new or changed data are added to the cluster, while the chunks containing data that did not change are left as is. This approach avoids storing redundant data, effectively deduplicating the data in the cluster. The deduplication takes place globally, across all of the devices in cluster, which allows for efficient use of storage space and avoids ingesting the same data twice over the network.
- **Distributed inline compression**

  When objects are broken into chunks, the chunks are compressed before they are distributed across the cluster, further conserving storage space and network resources.

**FlexHash**

Object storage systems use a hash allocation table to select the set of servers that store an object and the specific server that returns the object when requested. Some systems use consistent hashing, where objects and/or chunks are mapped to a set of storage servers for PUT operations and an ordered list for GET operations.

NexentaEdge uses a technology called FlexHash as an alternative to consistent hashing. With FlexHash, information about server load and capacity is taken into account when constructing the hash allocation table. For PUT operations, FlexHash maps chunks to multicast groups of servers that can store the chunk the fastest, based on the server load and capacity. For GET operations, FlexHash maps chunks to servers that can deliver the most recent version of the chunk the fastest.

**Replicast**

Replicast is NexentaEdge’s unique technology for storing and transporting objects in the cluster. It serves as the communications protocol for the servers in the NexentaEdge cluster.

Replicast uses information in the FlexHash table to determine the set of targets for PUT operations or the actual delivery server for GET operations. The result is that PUTs are directed to the least busy servers, and GETs are directed to the server that can provide the fastest response for the requested object.

**Components of a NexentaEdge Cluster**

From a physical perspective, a NexentaEdge cluster is a collection of server devices connected via a high-performance 10 Gigabit switch. From a logical perspective, a NexentaEdge cluster consists of data nodes and gateway nodes that communicate over a Replicast network. The cluster provides storage services over an external network using the protocols that NexentaEdge supports, including OpenStack Swift, OpenStack Cinder, Amazon S3, iSCSI, and NFS. Note that the term “NexentaEdge cluster” in this manual refers to a logical NexentaEdge cluster.

A NexentaEdge deployment consists of a single physical cluster and one or more logical clusters. Each logical cluster may have multiple namespaces configured for different tenants.

*Figure 1-1* shows the components of a NexentaEdge cluster.
A NexentaEdge cluster consists of the following components. A given device may have multiple roles assigned to it; for example, a server may be configured as a data node, gateway node and management controller.

- **Data nodes**

  The data nodes collectively provide the storage for the NexentaEdge cluster. Objects are broken into chunks and distributed across the data nodes using the Replicast protocol. The set of data nodes where the chunks are stored or retrieved is determined based on server load and capacity information.

  Data nodes must be configured with interfaces to an IPv6 Replicast network for data distribution and storage and to an IPv4 network (either an external network or dedicated management network) for initial configuration with the NEDEPLOY tool and subsequent administration with the NEADM tool.

  After initial configuration, data nodes require only a connection to the Replicast network, since administration of the data nodes is done by the deployment workstation via the management controller node, which has connectivity to both the management network and the Replicast network.
• Gateway nodes

Gateway nodes provide the connection between external clients and the data stored in the NexentaEdge cluster. Gateway nodes accept and respond to client requests, translating them into actions performed in the NexentaEdge cluster. Gateway nodes are provisioned with interfaces to the external network, Replicast network, and management network (if different from the external network).

When you configure a NexentaEdge cluster, you indicate which storage service(s) you want to provide for a given tenant, then specify which should be the gateway nodes for that cluster/tenant/service combination. Gateway nodes run the services within lightweight containers.

• Replicast network

The Replicast network is an isolated IPv6 VLAN used for communication and data transfer among the data nodes and gateway nodes in the NexentaEdge cluster. The Replicast protocol provides the means for efficient storage and retrieval of data in the cluster.

• Deployment workstation

The deployment workstation is the system from which you deploy and configure the NexentaEdge software to the other nodes. NexentaEdge uses the Chef environment for installation. You deploy NexentaEdge using a Chef Solo instance packaged with the NexentaEdge software.

To deploy NexentaEdge to the nodes in the cluster using the NEDEPLOY tool, the deployment workstation must have IPv4 network connectivity to the nodes, either through a management network or an external network.

• Management controller node

One of the data nodes in the cluster is designated the management controller node. The management controller translates external cluster-wide behavior into internal component-specific configuration, and may provide the connection between the deployment workstation and the data nodes in the Replicast network. Only one of the data nodes in the cluster can be a management controller. The management controller node needs to have network connectivity to both the deployment workstation and to the other nodes in the cluster.

• External network

External clients store and retrieve data in the NexentaEdge cluster by communicating with gateway nodes on the external network. The external network may use IPv4 or IPv6 and is likely to carry traffic unrelated to NexentaEdge.

• Management network

To aid in deploying and administering the NexentaEdge cluster, you may elect to place the deployment workstation and data nodes in a dedicated IPv4 management network. The NEDEPLOY and NEADM tools, running on the deployment workstation, send configuration information to and receive status information from the data nodes over this network.

• External clients

External clients are end-user machines that access data stored in the NexentaEdge cluster via gateway nodes. External clients access data in the cluster using APIs of the storage services NexentaEdge supports: OpenStack Swift and Amazon S3 via HTTP/REST, NFS or iSCSI block storage.
From a client perspective, the NexentaEdge cluster acts as an OpenStack Swift or Amazon S3 object storage system. To accommodate applications that expect block storage, the NexentaEdge cluster can act as an iSCSI or OpenStack Cinder target, or NFS share.

About This User Guide

This guide contains information about how to administer a NexentaEdge cluster and configure it to work with storage services. After completing the procedures in the *NexentaEdge Installation Guide* you should have a working cluster; this guide describes additional configuration tasks you may need to perform, including the following:

- Node management tasks, such as adding and removing servers and storage devices
- Cluster management tasks, including creating and managing clusters and tenants
- Configuration tasks that are specific to individual storage services: iSCSI, OpenStack Swift, NFS, and Amazon S3

This guide also includes a listing of the specific OpenStack Swift and Amazon S3 API function calls supported by NexentaEdge.
Managing Servers and Storage Devices

This chapter includes the following topics:

- Managing Servers
- Managing Devices

Managing Servers

This section describes how to add or remove server nodes from the NexentaEdge cluster. Server nodes contain the devices that provide storage in the NexentaEdge cluster.

- Adding New Nodes
- Removing Servers
- Managing Memory Usage in Hyper-Converged Setups
- Setting a Different Management Controller Node for the Cluster

Adding New Nodes

You can add new nodes to the existing NexentaEdge cluster as needed. NexentaEdge automatically distributes the workload among the newly added and existing nodes.

To add nodes to the cluster, you log into the NexentaEdge deployment workstation and use the NEDEPLOY tool. The NEDEPLOY tool features a pre-check procedure that verifies that the new node(s) meet the system requirements for NexentaEdge. The pre-check procedure ensures that the nodes are running a supported operating system, have sufficient network speed, the recommended amount of RAM, and unallocated disk space. If the pre-check procedure determines that any of the nodes do not meet the system requirements, NEDEPLOY exits without installing the NexentaEdge software. If this happens, the errors that caused the deployment to fail are displayed on screen, and you can view them in the ~/nedeploy/nedeploy_logs/nedge-deploy.log<timestamp> file. See the NexentaEdge Installation Guide for the node system requirements.

Before You Begin

Prior to adding new server nodes to a NexentaEdge cluster, do the following:

- Do the initial configuration for the new servers.
  
  Each new server must be provisioned with an operating system and management IP address. SSH access must also be enabled on the server. The /etc/sudoers on each new server must allow all root privileges to the user ID that will be deploying NexentaEdge.

- Configure networking for the new servers.
This includes specifying VLAN membership for the switch port(s) to which the new server is connected. Servers to be used as data nodes must be part of the Replicast VLAN and can optionally be part of a management VLAN or client VLAN. Servers to be used as gateway nodes must be part of the Replicast VLAN and client VLAN and can optionally be part of a management VLAN.

- Prepare storage devices for NexentaEdge deployment

During deployment, NexentaEdge allocates all raw (unpartitioned) devices visible on the node to NexentaEdge cluster storage. Note that if any data exists on these devices, it will be lost during the deployment process.

It is possible that the node to which you are deploying NexentaEdge already has disks that are formatted and/or mounted. In order for NexentaEdge to allocate these disks to cluster storage, the partition tables for the disks need to be erased. To do this, enter the following commands for each device to be used as NexentaEdge cluster storage:

```
# dd if=/dev/zero of=<device> bs=1M count=100
# hdparm -z <device>
```

Note that on CentOS systems, the `hdparm` utility may not be installed by default. If you need to install it, enter the following command:

```
# yum install hdparm -y
```

### Adding Servers Using the NEDEPLOY CLI

- To add servers using the NEDEPLOY CLI:

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEDEPLOY tool is installed).
2. Run the pre-check utility to ensure that the node meets the requirements for being added to the cluster.

   ```sh
   $ nedeploy precheck <ip-address> <username:password> -i <interface> [-t <profile>] [-x <disks-to-exclude>] [-X <disks-to-reserve>]
   ```

   Where:

   - `<ip-address>`: Is the management IP address of the node.
   - `username`: Is a user account on the node that has administrative privileges; for example, root.
     
     If you specify a different account than root, it must have password-less SSH access to the node, as well as all root privileges set in `/etc/sudoers` on the node.
   - `password`: Is the password for this user account.
   - `-i <interface>`: Is the Ethernet interface to be used for communication between nodes within the NexentaEdge cluster (that is, the Replicast network).
Example:

$ nedeploy precheck 10.3.32.12 root:password -i eth1 -t capacity

System response:

10.3.32.12 - Prechecking node
10.3.32.12 - Connecting
10.3.32.12 - Operating System: Ubuntu 16.04 LTS
10.3.32.12 - Maximum transmission unit (MTU): 1500
10.3.32.12 - Network Interface Speed: 10000
10.3.32.12 - Total Memory: 8175256kB
10.3.32.12 - Raw Disks:
10.3.32.12 - HDD /dev/sdb 40G
10.3.32.12 - HDD /dev/sdc 40G
10.3.32.12 - HDD /dev/sdd 40G
10.3.32.12 - HDD /dev/sde 40G
10.3.32.12 - Total Disk Space: 160GB
CPU instructions detected:  fpu vme de pse tsc mtrr pge mca cmov pat pse36 cflush dts mmx fxsr sse sse2 ss ht syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts nopl xtopology tsc_reliable nonstop_tsc aperfmperf pni pclmulqdq ssse3 cx16 sse4_1 sse4_2 popcnt aes hypervisor lahf_lm epb dtherm ida arat

CPU must support AVX + SSE_4.2+
10.3.32.12 - PRE-CHECK FAILED

If the pre-check utility indicates that the node does not meet the requirements for NexentaEdge, correct the issue if possible and re-run the utility.

3. After the server passes the pre-check, use the following command to add it to the cluster:

$ nedeplpy deploy solo {<ip-address>[:port] | <nodename>}<username:password>
- i <interface> [ -I <interface>] [-t <profile>]
[-x <disks-to-exclude>] [ -X <disks-to-reserve>] [ -z <zone>]

Where:

- `<ip-address>` Is the management IP address of the node. You can optionally specify the SSH port if it is not the default of 22.

- `<nodename>` Is the name of the node to add to the cluster. This will be the name recorded in the Chef database for the node.

- `username` Is a user account on the node that has administrative privileges; for example, root.

- `password` Is the password for this user account.

- `-i <interface>` Is the Ethernet interface to be used for communication between nodes within the NexentaEdge cluster (that is, the Replicast network).

- `-I <interface>` Optionally specifies the Ethernet interface to be used for the gateway. If a storage service (iSCSI, Swift, S3, NFS) is started on this node, it would use this interface. If you do not specify the gateway interface, the storage service uses the interface specified with the `-i` option.
Has to do with how metadata is distributed among the hard disk drives (HDDs) and solid state drives (SSDs) on the node and whether journaling operations are enabled. You can specify one of the following profiles:

**capacity** NexentaEdge uses all of the available HDDs and SSDs as one large storage pool.

**performance** NexentaEdge offloads the majority of metadata to SSD, assuming a hybrid SSD/HDD deployment. This is the default profile.

**gateway** NexentaEdge configures the node as a gateway with no disks allocated for cluster storage.

**all-flash** NexentaEdge optimizes the node for use in an all-flash cluster.

Specifies a comma-separated list of one or more devices that NexentaEdge will not use as cluster storage. NexentaEdge adds these disks to a list of devices that it will never allocate to cluster storage. In this list, you can specify disks that you are using for other applications besides NexentaEdge.

Specifies a comma-separated list of one or more devices that NexentaEdge will not allocate to cluster storage for this deployment. The difference between this list and the `<disks-to-exclude>` list is that the disks in the `<disks-to-exclude>` list are permanently excluded from use in the NexentaEdge cluster, while disks in the `<disks-to-reserve>` can be added to the cluster at a later time.

Note that the pre-check utility includes the `<disks-to-reserve>` when calculating the total memory required per node.

Specifies the zone to which the node belongs. All of the nodes in a given zone are considered to be part of the same failure domain; for example, a group of nodes that receive power from the same source can be placed in the same zone. NexentaEdge ensures that data in the cluster is replicated across multiple zones, so that failure of the nodes in a zone does not result in lost access to the data.

Zone settings must be applied prior to system initialization. See the NexentaEdge Installation Guide for more information about zone configuration.

Configures this node as a management controller.

Optionally specifies the filesystem type to use for LFS drivers. You can specify `ext4` or `none`. The default of `none` causes the RD (raw disk) driver to be automatically enabled.
4. Repeat the previous step for each node you want to add to the cluster.

5. Verify the system status:

   $ neadm system status
   
   System response:

<table>
<thead>
<tr>
<th>ZONE:HOST:CID</th>
<th>SID</th>
<th>UTIL</th>
<th>CAP</th>
<th>CPU</th>
<th>MEM</th>
<th>DEVs</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:node32:</td>
<td>B5DEF7B9B3C690...</td>
<td>0%</td>
<td>160G</td>
<td>2/0.2@2.34Ghz</td>
<td>4.16G/7.8G</td>
<td>4/4</td>
<td>ONLINE</td>
</tr>
<tr>
<td>0:node33:</td>
<td>[M] B3EF9C30B6F79F...</td>
<td>0%</td>
<td>160G</td>
<td>2/0.1@2.34Ghz</td>
<td>4.18G/7.8G</td>
<td>4/4</td>
<td>ONLINE</td>
</tr>
<tr>
<td>0:node34:</td>
<td>155B81C291FA4E...</td>
<td>0%</td>
<td>160G</td>
<td>2/0.1@2.34Ghz</td>
<td>4.18G/7.8G</td>
<td>4/4</td>
<td>ONLINE</td>
</tr>
<tr>
<td>0:node35:</td>
<td>9BB624B94C8E2C...</td>
<td>0%</td>
<td>160G</td>
<td>2/0.1@2.34Ghz</td>
<td>4.15G/7.8G</td>
<td>4/4</td>
<td>ONLINE</td>
</tr>
<tr>
<td>0:node36:</td>
<td>F88C6C3180C34E...</td>
<td>0%</td>
<td>160G</td>
<td>2/0.1@2.34Ghz</td>
<td>4.15G/7.8G</td>
<td>4/4</td>
<td>ONLINE</td>
</tr>
</tbody>
</table>

   Ensure that the servers you added to the cluster have a state of ONLINE.

   **Note:** Whenever you run the `neadm system status` command, NexentaEdge by default automatically performs a service checkpoint status check. Normally this has no impact on system performance, but on larger deployments, it may take some time to complete. On systems that are larger and more stable, you can disable the service checkpoint status check by changing the `ENABLE_AUTOMATED_CHECKPOINT` setting in the `.neadmrc` file on the NexentaEdge deployment workstation from 1 to 0.

### Removing Servers

To remove a server node from the NexentaEdge cluster, simply disconnect it or power it off. NexentaEdge automatically detects the change, removes the server from the cluster, and rebalances the workload among the remaining servers. Enter the `neadm system service-checkpoint set` command after making physical changes to the number of nodes in the cluster.

**Note:** that the server you remove may have a status of **FAULTED** when you run the `neadm system status` command. To reset the status for the cluster, enter the `neadm system service-checkpoint refresh` command.
Managing Memory Usage in Hyper-Converged Setups

NexentaEdge supports fully hyper-converged setups. A hyper-converged setup allows storage and compute environments to exist seamlessly within any node on the Replicast network.

In a hyper-converged setup, you should manage memory allocation on the node to account for the maximum amount of memory dedicated to any particular service hosted on the node, as well as the potential memory usage the data services may require.

The default configuration for a data node could potentially use 70% or more of the available system resources for storage services. You can control the memory usage on a data node with the following setting on the data node:

```bash
$ echo "export  CCOWD_CGROUP_MEMLIM <number-of-bytes>" >> /opt/nedge/.local
```

After you change this setting, enter the following command to restart all services on the data node:

```bash
$ service nedge restart
```

You should also manage resource utilization on multi-service nodes, where multiple NexentaEdge services are resident on the same node. This is done using the \texttt{X-Service-Mem-Limit} setting for a given storage service. See the sections on “Setting Resource Limits” in this guide for the storage service you are configuring.

By default, any storage service can request and use all resources on the node, so you should ensure that the settings for \texttt{CCOWD_CGROUP_MEMLIM} + \texttt{X-Service-Mem-Limit} for all services resident on any given node do not exceed the node’s available resources.

Setting a Different Management Controller Node for the Cluster

In a NexentaEdge cluster, the management controller node translates external cluster-wide behavior into internal component-specific configuration, and may provide the connection between the deployment workstation and the data nodes in the Replicast network.

At least one of the nodes in the cluster must be designated the management controller node. The management controller node needs to have network connectivity to both the deployment workstation and to the other nodes in the cluster.

In the event that the management controller node should fail, or if you want to use a different node in the cluster as the management controller node, do the following:

1. On the node that you want to designate as the new management controller node:
   1. Edit the file `/opt/nedge/etc/ccow/auditd.ini` and change the following setting:
      ```ini
      is_aggregator=0
      ```
      to:
      ```ini
      is_aggregator=1
      ```
   2. Enable the path for the NexentaEdge NEFADM tool:
      ```bash
      $ source /opt/nedge/env.sh
      ```
   3. Restart the NexentaEdge audit service:
      ```bash
      $ nefadm restart auditserv
      ```
On the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed):

1. Edit the file `/neadm/.neadmrc` and change the following setting:
   ```
   API_URL=http://<ip-address>:8080
   ```
   So that `<ip-address>` is the address of the new management controller node.

2. If you installed the NexentaEdge GUI, enter the following commands:
   ```
   $ neadm service config nedgeui X-Mgmt-Address <ip-address>
   $ neadm service restart nedgeui
   ```
   Where `<ip-address>` is the address of the new management controller node.

3. If the physical number of nodes in the cluster has changed, enter the following command:
   ```
   $ neadm system clear
   ```

4. Set the checkpoint for the cluster; enter the following command:
   ```
   $ neadm system service-checkpoint set
   ```

Managing Devices

A *device* refers to a storage device installed on a server node in the NexentaEdge cluster. Devices can be hard disk drives, SSDs, key value storage drives, and so on. NexentaEdge uses all of the storage devices installed on a server node (other than the operating system disk) for data storage in the cluster.

Using NEADM, you can perform the following actions with devices:

- Viewing the List of Devices
- Viewing Device Status
- Adding a Device
- Removing a Device
- Replacing a Device

**Viewing the List of Devices**

You can view the list of devices available on a specified server node.

To view the list of devices on a server node:

1. Log in to the NexentaEdge deployment workstation.
2. List the IDs of the server nodes:
   ```
   $ neadm system status
   ```
   System response:

<table>
<thead>
<tr>
<th>ZONE:HOST:CID</th>
<th>SID</th>
<th>UTIL</th>
<th>CAP</th>
<th>CPU</th>
<th>MEM</th>
<th>DEVs</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:node32:</td>
<td>B5DEF7B9B3C690...</td>
<td>0%</td>
<td>160G</td>
<td>2/0.2@2.34Ghz</td>
<td>4.16G/7.8G</td>
<td>4/4</td>
<td>ONLINE</td>
</tr>
</tbody>
</table>
0:node33:    [M] B3EF9C30B6F79F... 0%   160G 2/0.1@2.34Ghz 4.18G/7.8G 4/4 ONLINE
0:node34:    155B81C291FA4E... 0%   160G 2/0.1@2.34Ghz 4.18G/7.8G 4/4 ONLINE
0:node35:    9BB624B94C8E2C... 0%   160G 2/0.1@2.34Ghz 4.18G/7.8G 4/4 ONLINE
0:node36:    F88C6C3180C34E... 0%   160G 2/0.1@2.34Ghz 4.18G/7.8G 4/4 ONLINE

3. Note the ID of the server for which you want to display device information.

4. Using the server ID, list the devices on the node:

   $ neadm device list <server_id>

   Example:

   $ neadm device list 155B81C291FA4E2F8E5B5ED3C1017956

   System response:

<table>
<thead>
<tr>
<th>ZONE:HOST:CID</th>
<th>SID/DEVID</th>
<th>UTIL</th>
<th>CAP</th>
<th>RLAT</th>
<th>WLAT</th>
<th>REPQ</th>
<th>VERQ</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:node34:</td>
<td>155B81C291FA4E2F8E5B5ED3C1017956</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ONLINE</td>
</tr>
</tbody>
</table>

Viewing Device Status

You can view the status of the devices in the NexentaEdge cluster, including utilization percentage and online/offline state.

- To view the status of all devices on all server nodes:

  1. Log in to the NexentaEdge deployment workstation.
  2. Type:

     $ neadm system status -v

   System response:

<table>
<thead>
<tr>
<th>ZONE:HOST:CID</th>
<th>SID/DEVID</th>
<th>UTIL</th>
<th>CAP</th>
<th>RLAT</th>
<th>WLAT</th>
<th>REPQ</th>
<th>VERQ</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:data01-ab:</td>
<td>B5DEF7B9B3C69055A9079125BC24FD29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ONLINE</td>
</tr>
<tr>
<td>0:data02-ab:</td>
<td>155B81C291FA4E2F8E5B5ED3C1017956</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ONLINE</td>
</tr>
<tr>
<td>0:data03-ab:</td>
<td>9BB624B94C8E2C45BCC7D0255B69F2C2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ONLINE</td>
</tr>
<tr>
<td>0:data04-ab:</td>
<td>F88C6C3180C34E687ED75625CBF6F126</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ONLINE</td>
</tr>
</tbody>
</table>

Copyright © 2018 Nexenta Systems, ALL RIGHTS RESERVED
www.nexenta.com
Adding a Device

To add a new storage device, you first physically install the device in the server node, then add the device to the cluster using the NEADM tool. Note that when you add a device to the cluster, any pre-existing data on the device is lost.

To add a device:

1. Install the new storage device in the server node.
2. Write down the serial number of the new device; it will be used by the kernel to identify the device.
3. Log in to the server node as an administrator.
4. Change the directory to `/dev/disk/by-id`.
5. List the contents of the `/dev/disk/by-id` directory:
   
   ```
   ls
   ```
   
   System response:
   
   ```
   ata-VMware_Virtual_IDE_CDROM_Drive_1000000000000000000001
   scsi-36000c2968f9444b14298d471217c33
   scsi-36000c2993181eedcc0ffbe07ae3466
   scsi-36000c29bd5bc5759ea4e738a384fb3
   scsi-36000c29bd5bc5759ea4e738a384fb3-part1
   scsi-36000c29bd5bc5759ea4e738a384fb3-part2
   scsi-36000c29bd5bc5759ea4e738a384fb3-part5
   scsi-36000c29c1400925c595e7fb7012900
   scsi-36000c29d07120fdc48526f82bfe7ea
   ```
6. Copy the name of the new storage device.
7. Log into the NexentaEdge deployment workstation.
8. Add the new device to the NexentaEdge cluster:
   
   ```
   $ neadm device add <server_id> <device_id>
   ```
   
   Example:
   
   ```
   $ neadm device add D586BF84009C4230BFEE31CAC961197D scsi-36000c2993181eedcc0ffbe07ae3466
   ```
   
   Use the `neadm system status` command if you need to find the ID of the server node.
9. Verify that the new device was successfully added to the NexentaEdge cluster:
   
   ```
   $ neadm device list <server_id>
   ```
Example:

```
$ neadm device list CF59EA43F4B5F936108CC07C8882A2FA
```

System response:

```
ZONE:HOST:CID     SID/DEVID                     UTIL CAP  RLAT WLAT REPQ   STATE
0:ee-nedge-node4: CF59EA43F4B5F936108CC07C8882A2FA
ata_1533110BB9E AEA068AC57458B15F2913F366F830 0% 676G 22 35 0      ONLINE
ata_1533110BB78 1117E576DE4413FFE0F412B4999AE 0% 676G 15 29 0      ONLINE
```

### Removing a Device

- **To remove a device from the cluster:**
  1. Physically extract the device from the server node.
  2. Write down the serial number of the device.
  3. Log into the NexentaEdge deployment workstation.
  4. Delete the device from the cluster:
     ```
     $ neadm device detach <server_id> <device_id>
     ```
     Example:
     ```
     $ neadm device detach D586BF84009C4230BFEE31CAC961197D scsi-36000c2993181e0fde07ae3466
     ```
  5. Clear statistics for the detached device from the system:
     ```
     $ neadm system clear
     ```
     System response:
     Node stats cleared successfully
  6. Set a new service checkpoint:
     ```
     $ neadm system service-checkpoint set
     ```
     System response:
     Service checkpoint created successfully
  7. Verify that the device is no longer part of the NexentaEdge cluster:
     ```
     $ neadm device list <server_id>
     ```
     Example:
     ```
     $ neadm device list CF59EA43F4B5F936108CC07C8882A2FA
     ```
     System response:
     ```
     ZONE:HOST:CID     SID/DEVID                     UTIL CAP  RLAT WLAT REPQ   STATE
     0:ee-nedge-node4: CF59EA43F4B5F936108CC07C8882A2FA
     ata_1533110BB9E AEA068AC57458B15F2913F366F830 0% 676G 22 35 0      ONLINE
     ata_1533110BB78 1117E576DE4413FFE0F412B4999AE 0% 676G 15 29 0      ONLINE
     ```
Replacing a Device

If a storage device fails, Nexenta recommends that you replace it as soon as possible. Multiple replicas of stored objects are retained in the NexentaEdge cluster, so a failed device does not lead to any loss of data. However, a failed device is not able to participate in the cluster, which may lead to decreased overall performance.

To replace a device with a new one, you physically extract the failed device from the server node, note the device ID of the old and new devices, and insert the new device into the server node. You then replace the old device with the new one in the cluster using the NEADM tool.

To replace a device:

1. Physically extract the old device from the server node, and if necessary install the new device.
2. Write down the serial numbers of the old and new devices.
3. Log into the NexentaEdge deployment workstation.
4. Replace the old device in the cluster with the new device:
   ```
   $ neadm device replace <server_id> <old_device_id> <new_device_id> [-f]
   ```
   The `-f` option formats the new device. If you encounter errors when attempting to replace the device, use the `-f` option. Note that any data on the device will be lost.
   Example:
   ```
   $ neadm device replace D586BF84009C4230BFEE31CAC961197D scsi-36000c2993181edd0fbeb07ae3466 scsi-36000c295d3435e9e3927f3260c9958
   ```
5. Clear statistics for the old device from the system:
   ```
   $ neadm system clear
   ```
   System response:
   `Node stats cleared successfully`
6. Set a new service checkpoint:
   ```
   $ neadm system service-checkpoint set
   ```
   System response:
   `Service checkpoint created successfully`
7. Verify that the new device has replaced the old device in the NexentaEdge cluster:
   ```
   $ neadm device list <server_id>
   ```
   Example:
   ```
   $ neadm device list CF59EA43F4B5F936108CC07C8882A2FA
   ```
System response:

<table>
<thead>
<tr>
<th>ZONE:HOST:CID</th>
<th>SID/DEVID</th>
<th>UTIL</th>
<th>CAP</th>
<th>RLAT</th>
<th>WLAT</th>
<th>REPQ</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:ee-nedge-node4:</td>
<td>CF59EA43F4B5F936108CC07C8882A2FA</td>
<td>0%</td>
<td>676G</td>
<td>22</td>
<td>35</td>
<td>0</td>
<td>ONLINE</td>
</tr>
<tr>
<td>ata_1533110BB9E</td>
<td>AEA068AC57458B15F2913F366F830</td>
<td>0%</td>
<td>676G</td>
<td>15</td>
<td>29</td>
<td>0</td>
<td>ONLINE</td>
</tr>
<tr>
<td>ata_1533110BB78</td>
<td>1117E576DE4413FFE0F412B4999AE</td>
<td>0%</td>
<td>676G</td>
<td>22</td>
<td>35</td>
<td>0</td>
<td>ONLINE</td>
</tr>
<tr>
<td>ata_1533110BB9E</td>
<td>AEA068AC57458B15F2913F366F830</td>
<td>0%</td>
<td>676G</td>
<td>15</td>
<td>29</td>
<td>0</td>
<td>ONLINE</td>
</tr>
<tr>
<td>ata_1533110BB78</td>
<td>1117E576DE4413FFE0F412B4999AE</td>
<td>0%</td>
<td>676G</td>
<td>15</td>
<td>29</td>
<td>0</td>
<td>ONLINE</td>
</tr>
</tbody>
</table>
Administering the NexentaEdge System

This chapter includes the following topics:

- Displaying Cluster Status
- Displaying Statistics for the Replicast Network
- Configuring Zoning for Data Nodes
- Initializing the Cluster
- Configuring Autosupport
- Enabling SNMP Traps

Displaying Cluster Status

The NEADM tool provides commands to display information about the NexentaEdge cluster, as well as status for the server nodes and the storage devices used in the cluster.

To display information about the NexentaEdge cluster:

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).
2. Type:
   ```bash
   $ neadm system summary
   ```

   System response:

<table>
<thead>
<tr>
<th>PARAM</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Raw Capacity</td>
<td>117G</td>
</tr>
<tr>
<td>Physical Free</td>
<td>116G</td>
</tr>
<tr>
<td>Physical Allocated</td>
<td>278M</td>
</tr>
<tr>
<td>Logical Used</td>
<td>1.67G</td>
</tr>
<tr>
<td>Estimated Used</td>
<td>715G</td>
</tr>
<tr>
<td>Estimated Reduction Ratio</td>
<td>18.45</td>
</tr>
<tr>
<td>Estimated Capacity Savings</td>
<td>4.73G</td>
</tr>
<tr>
<td>Estimated Available</td>
<td>715G</td>
</tr>
<tr>
<td>Utilization</td>
<td>0%</td>
</tr>
<tr>
<td>Total Objects</td>
<td>50</td>
</tr>
<tr>
<td>Servers Online/Degraded</td>
<td>3/0</td>
</tr>
<tr>
<td>VDEVs Online/Degraded</td>
<td>12/0</td>
</tr>
<tr>
<td>Cluster Replication Queue</td>
<td>0</td>
</tr>
</tbody>
</table>

Note that the statistics are not realtime numbers and may take 1 minute or longer to display, especially if the system is under heavy load.
The command displays the following output:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Raw Capacity</td>
<td>The total amount of storage available on all the nodes in the cluster.</td>
</tr>
<tr>
<td>Physical Free</td>
<td>The amount of physical storage available for storing objects in the cluster.</td>
</tr>
<tr>
<td>Physical Allocated</td>
<td>The amount of physical storage currently used for storing objects.</td>
</tr>
<tr>
<td>Logical Used</td>
<td>The uncompressed size of the objects stored in the cluster.</td>
</tr>
<tr>
<td>Estimated Used</td>
<td>The approximate amount of physical storage that would be used if the stored objects were not deduplicated and compressed, taking into account space used for replicating the data across the cluster.</td>
</tr>
<tr>
<td>Estimated Reduction Ratio</td>
<td>An estimate of the deduplication ratio for data in the cluster; that is, the amount of data input into the cluster divided by the amount of data actually stored in the cluster. The higher the number, the greater the amount of space saved over storing non-deduplicated data. For example, if the Reduction Ratio is 6, it means that deduplication has reduced the space needed to store data in the cluster to one-sixth of non-deduplicated size.</td>
</tr>
<tr>
<td>Estimated Capacity Savings</td>
<td>The approximate amount of space saved due to the stored objects being deduplicated and compressed, taking into account space used for replicating the data across the cluster. For example, if 100GB of user data is stored in the cluster without deduplication, it would take 300GB of cluster space, since the data is replicated three times across the cluster. If after deduplication, the data takes 50GB of cluster space (Reduction Ratio of 6), then the Capacity Savings would be 250GB.</td>
</tr>
<tr>
<td>Estimated Available</td>
<td>The approximate amount of logical space available to store objects, based on the amount of Physical Free space, the Reduction Ratio, and data replication. For example, if the Physical Free space is 100TB, and the Reduction Ratio is 6, and the deduplicated data is replicated 3 times across the cluster, then the Estimated Available space would be approximately 200TB (100TB x 6 / 3).</td>
</tr>
<tr>
<td>Utilization</td>
<td>The percentage of available storage space that is currently storing objects.</td>
</tr>
</tbody>
</table>
To view information about the server nodes in the cluster:

1. Log in to the NexentaEdge deployment workstation.
2. Type:
   
   `$ neadm system status`

   System response:

<table>
<thead>
<tr>
<th>ZONE:HOST:CID</th>
<th>SID</th>
<th>UTIL</th>
<th>CAP</th>
<th>CPU</th>
<th>MEM</th>
<th>DEVs</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:node32:</td>
<td>B5DEF7B9B3C690... 0%</td>
<td>160G</td>
<td>2/0.2@2.34Ghz</td>
<td>4.16G/7.8G</td>
<td>4/4 ONLINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0:node33:</td>
<td>[M] B3EF9C30B6F79F... 0%</td>
<td>160G</td>
<td>2/0.1@2.34Ghz</td>
<td>4.18G/7.8G</td>
<td>4/4 ONLINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0:node34:</td>
<td>155B81C291FA4E... 0%</td>
<td>160G</td>
<td>2/0.1@2.34Ghz</td>
<td>4.18G/7.8G</td>
<td>4/4 ONLINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0:node35:</td>
<td>9BB624B94C8E2C... 0%</td>
<td>160G</td>
<td>2/0.1@2.34Ghz</td>
<td>4.15G/7.8G</td>
<td>4/4 ONLINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0:node36:</td>
<td>F88C6C3180C34E... 0%</td>
<td>160G</td>
<td>2/0.1@2.34Ghz</td>
<td>4.15G/7.8G</td>
<td>4/4 ONLINE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The command displays the following output:

- **ZONE**: The zone (failure domain) specified for the node. Default is 0. See the NexentaEdge Installation Guide for information on zone configuration.
- **HOST**: The name of the node. This is the name you specified as the Chef name when you deployed the node. If the node is configured as the Management Controller, then [M] appears next to the node name.
- **CID**: The Docker container ID, if applicable.
- **SID**: The server ID of the node.
- **UTIL**: The percentage of storage space on the node that is currently in use.
- **CAP**: The total amount of storage available on the node.
- **CPU**: Number of CPUs on the node / average system load for the previous 15 minutes, and the speed of the CPUs.
- **MEM**: The used / total amount of RAM on the node.
- **DEVs**: The total number of devices on the node / the number of devices that are online.
- **STATE**: Whether the node is online or in a degraded state.

3. Use the `-v` option to display information about the devices on the server nodes. For example:
The command displays the following output:

<table>
<thead>
<tr>
<th>ZONE:HOST:CID</th>
<th>SID/DEVID</th>
<th>UTIL</th>
<th>CAP</th>
<th>LAT</th>
<th>WLAT</th>
<th>REPQ</th>
<th>VERQ</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:data01-ab:</td>
<td>B5DEF7B9B3C6055A9079125BC24FD29</td>
<td>0%</td>
<td>40G</td>
<td>64</td>
<td>0</td>
<td>16</td>
<td>ONLINE</td>
<td></td>
</tr>
<tr>
<td>sdsi-3600c293066e4f3d3d8f3387e70c7e25f987</td>
<td>7B792A402DAE892A9242CC47AB089A3 0%</td>
<td>40G</td>
<td>64</td>
<td>0</td>
<td>16</td>
<td>ONLINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sdsi-3600c2938d4e5d8f3387e70c7e25f987</td>
<td>95B3202C1D16ACF7B8595875876126CD1E 0%</td>
<td>40G</td>
<td>64</td>
<td>0</td>
<td>64</td>
<td>16</td>
<td>ONLINE</td>
<td></td>
</tr>
<tr>
<td>sdsi-3600c293293c94756c9266eb62c3c9476591 0%</td>
<td>40G</td>
<td>64</td>
<td>0</td>
<td>64</td>
<td>16</td>
<td>ONLINE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The zone (failure domain) specified for the node. Default is 0. See the NexentaEdge Installation Guide for information on zone configuration.

The name of the node. This is the name you specified as the Chef name when you deployed the node. If the node is configured as the Management Controller, then [M] appears next to the node name.

The ID of the Docker container, if applicable.

The server ID or device ID.

The percentage of storage space on the device that is currently in use.

The total amount of storage available on the device.

Latency for GET operations in microseconds.

Latency for PUT operations in microseconds.

Depth of the replication queue for each device. A non-zero number indicates there are outstanding replication tasks within the system. This may be due to asynchronous writes or during recovery.

Depth of the verification queue for each device.

Whether the device is online or in a degraded state.
Displaying Statistics for the Replicast Network

NexentaEdge gathers statistics about the operation of the Replicast network, over which the nodes in the cluster exchange data. You can display these statistics using the NEADM tool. If a node or a device is in a state other than ONLINE, then viewing the Replicast statistics may provide information about what is causing it to be in a degraded state.

To display Replicast statistics for the NexentaEdge cluster:

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).
2. To display Replicast statistics for each node in the cluster, type:
   $ neadm system status -x
   To display Replicast statistics for each VDEV on each node in the cluster, type:
   $ neadm system status -xv

System response:

```
9583202C10D16ACF7B5958767126C scsi-36000c2938d4ca2dd838bdbe1bfad5c15
Verify Avg:                100
Incoming Batch Avg:        100
Ngcount Delay Avg:         139
Ngcount Delay:             528
Get Disk Qdepth:           0
Put Disk Qdepth:           0
Get Net Tx:                0
Put Net Rx:                0
Vbuf Total:                1310720000
Vbuf Queued:               0
Vbuf Reserved:             0
Namedget Active:           2
Namedput Active:           0
Unnamedget Active:         0
Unnamedput Active:         1
Retransmits:               0
Batch Queue Entries:       0
Batch Inc. Queue Entries:  0
Persistent MD Entries:     25
Persistent MD Size:        36864
Temporary MD Entries:      4
Temporary MD Size:          4096
MD Offload Entries:        0
MD Offload Size:           0
MD Offload Capacity:       0
```
Payload Entries: 160177
Payload Size: 36556800

The command displays the following output. Time values are in milliseconds.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify Avg</td>
<td>The average time it takes to process one verification request in the background.</td>
</tr>
<tr>
<td>Incoming Batch Avg</td>
<td>The average processing time for incoming batches for verification.</td>
</tr>
<tr>
<td>Ngcount Delay Avg</td>
<td>The average delay for receiving replies on counting the negotiating group members.</td>
</tr>
<tr>
<td>Ngcount Delay</td>
<td>The delay in counting the negotiating group members</td>
</tr>
<tr>
<td>Get Disk Qdepth</td>
<td>The number of outstanding Get requests for this disk.</td>
</tr>
<tr>
<td>Put Disk Qdepth</td>
<td>The number of outstanding Put requests for this disk.</td>
</tr>
<tr>
<td>Get Net Tx</td>
<td>Count of the current Gets for the incoming network requests.</td>
</tr>
<tr>
<td>Put Net Rx</td>
<td>Count of the current Puts for the incoming network requests.</td>
</tr>
<tr>
<td>Vbuf Total</td>
<td>Total number of virtual buffers pre-allocated for this Replicast network device.</td>
</tr>
<tr>
<td>Vbuf Queued</td>
<td>The number of virtual buffers queued to be reserved, but not reserved yet.</td>
</tr>
<tr>
<td>Vbuf Reserved</td>
<td>Based on the number of data transfers currently in-process in Replicast, this is the number of virtual buffers currently reserved.</td>
</tr>
<tr>
<td>Namedget Active</td>
<td>The number of named-get requests currently active.</td>
</tr>
<tr>
<td>Namedput Active</td>
<td>The number of named-put requests currently active.</td>
</tr>
<tr>
<td>Unnamedget Active</td>
<td>The number of unnamed-get requests for chunks currently active.</td>
</tr>
<tr>
<td>Unnamedput Active</td>
<td>The number of unnamed-put requests for chunks currently active.</td>
</tr>
<tr>
<td>Retransmits</td>
<td>Number of retransmits for data transfer to the clients.</td>
</tr>
<tr>
<td>Batch Queue Entries</td>
<td></td>
</tr>
<tr>
<td>Batch Inc. Queue Entries</td>
<td></td>
</tr>
<tr>
<td>Persistent MD Entries</td>
<td></td>
</tr>
<tr>
<td>Persistent MD Size</td>
<td></td>
</tr>
<tr>
<td>Temporary MD Entries</td>
<td></td>
</tr>
<tr>
<td>Temporary MD Size</td>
<td></td>
</tr>
<tr>
<td>MD Offload Entries</td>
<td></td>
</tr>
<tr>
<td>MD Offload Size</td>
<td></td>
</tr>
</tbody>
</table>
In the context of NexentaEdge, a **zone** refers to a group of servers that for failure considerations can be treated as a single domain; that is, as a failure domain. For example, a zone can be made up of a group of servers in a rack that receive power from a single source. If that power source should fail, all of the servers in the zone will lose power and fail. In this example, the rack is the failure domain. In NexentaEdge, a failure domain is represented as a zone.

When NexentaEdge replicates data across the nodes in the cluster, it makes three copies of each chunk and stores each copy in a different location. To ensure constant availability of the data, each replicated chunk should be stored on a node in a different zone. This ensures that if all the nodes in one zone fail, the data is still accessible from a node in one of the remaining zones.

By default, if you do not assign the data nodes to zones, all of the nodes are considered to be part of a single zone, **zone 0**. NexentaEdge distributes the chunks across the nodes in the cluster without regard to zone assignment.

To configure zoning, you assign the data nodes to zones other than zone 0. When you do this, NexentaEdge takes a node’s zone assignment into account when selecting where it stores each replicated chunk. NexentaEdge distributes copies of a chunk across the zones, so that if the chunk is not available from a given zone (for example, if all of the nodes in the zone fail), it may be available from one of the other zones.

You can assign nodes to zones either during the NexentaEdge deployment process or using NEADM. The procedure below shows how to configure zoning using NEADM. In either case, if you plan to use zones, it is important that you do the configuration prior to system initialization.

For information about how to configure zoning during NexentaEdge deployment using NEDEPLOY, see the *NexentaEdge Installation Guide*.

A zoning configuration requires a minimum of three zones, each consisting of one or more data nodes. Nexenta recommends that the zones be similar in terms of storage, memory, and CPU capacity. This can ensure that any new zones you add have sufficient resources to cover for a failed zone in case of a zone failure.

If you configure new zones with additional nodes after the initial NexentaEdge deployment, the number of nodes in the new zone must be at least 75 percent of the number of nodes in the other zones. For example, if your existing NexentaEdge configuration has 3 zones of 7 nodes each, and you want to add a new 4th zone, the new zone must have at least 5 (75 percent of 7) nodes. This is required to ensure that the new zone has sufficient resources (storage, memory, CPU capacity) to cover for a failed zone in case of a zone failure.

When data nodes are deployed in multiple Docker containers on a single physical server, all of the data nodes on that server are part of the same zone to which the server is assigned.

To configure zoning:

1. From the NexentaEdge deployment workstation, use the following command to assign a node to a zone.
$ neadm system zone <nodename> <zone>

Example:
$ neadm system zone newnode101 1

2. Repeat the previous step for all of the nodes to be included in the zoning configuration.

3. After configuring zones for the data nodes in the NexentaEdge cluster, you can initialize the cluster. See Initializing the Cluster next.

## Initializing the Cluster

When you first deploy the server nodes in the cluster, you must manually initialize the cluster.

**Important:** If you plan to use zones, you must configure them prior to doing the procedure below. See Configuring Zoning for Data Nodes for more information.

- **To initialize the NexentaEdge cluster:**
  1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).
  2. Initialize the cluster:

     ```
     $ neadm system init
     ```

     **System response:**

     enter to continue
     Do you agree to the EULA? "YES / NO" : **YES**
     NAME VALUE
     GenID 149138187453656
     NumRows 16
     HashMask 0xf
     ServerCount 5
     VdevCount 25
     
     STATUS: Cluster in consistent state
     Setting up initial configuration checkpoint, please wait ... 
     Service checkpoint created successfully 
     NexentaEdge cluster initialized successfully.
     System GUID: 590539A1-BFC8-4986-801F-156D1B9649CD

## Configuring Autosupport

The NexentaEdge Autosupport feature collects information about cluster-related processes running on each node. By default, the nodes in the cluster send the collected information to Nexenta, where it may be useful to Nexenta Technical Support personnel when troubleshooting problems with the NexentaEdge cluster and Nexenta Product Development for improving future versions of NexentaEdge.
The Autosupport feature is enabled by default. When the NexentaEdge cluster is initialized, Autosupport starts collecting data and saves it in gzipped, JSON-format report files in the `/opt/nedge/var/lib/nef/reports` directory on each node. This information is sent from each node to Nexenta, specifically to the URL https://logcollector.nexenta.com. You can optionally disable the nodes from sending Autosupport information if necessary.

Note that Autosupport anonymizes the information prior to sending it to Nexenta. No site-specific information, such as node IP addresses or usernames/passwords, is included in Autosupport reports.

Using NEADM, you can optionally direct the nodes to send the report files to a specified collector destination in your network, where the data can be aggregated. In addition, you can optionally configure NexentaEdge to send the aggregated Autosupport information from the collector to an external destination.

- **To configure the nodes to send Autosupport report files to a specified collector destination:**
  1. Log in to the NexentaEdge deployment workstation.
  2. To specify an HTTP server as the collector destination, use the following command:
     ```
     $ neadm system autosupport -c <collector_url>
     ```
     Example:
     ```bash
     $ neadm system autosupport -c https://192.168.1.101:8050
     ```
  3. To specify an e-mail address as the collector destination, use the following command:
     ```
     $ neadm system autosupport -e <collector_e-mail_address>
     ```
     Example:
     ```bash
     $ neadm system autosupport -e support@nexenta.com
     ```

- **To disable the nodes from sending Autosupport report files:**
  1. From the NexentaEdge deployment workstation, enter the following commands:
     ```bash
     $ neadm system autosupport -c disabled
     $ neadm system autosupport -e disabled
     ```

- **To configure how often Autosupport reports are generated:**
  1. From the NexentaEdge deployment workstation, enter the following command:
     ```bash
     $ neadm system autosupport -i <report_interval_in_seconds>
     ```
     By default, Autosupport reports are generated once per hour (3,600 seconds).

- **To specify the maximum amount of time NexentaEdge can use to generate an Autosupport report:**
  1. From the NexentaEdge deployment workstation, enter the following command:
     ```bash
     $ neadm system autosupport -t <report_timeout_in_seconds>
     ```

- **To specify how often NexentaEdge retries sending an Autosupport report if delivery to the collector destination fails:**
  1. From the NexentaEdge deployment workstation, enter the following command:
     ```bash
     $ neadm system autosupport -r <retry_timeout_in_seconds>
     ```
To specify the location where the Autosupport reports are stored on the nodes:

1. From the NexentaEdge deployment workstation, enter the following command:

```
$ neadm system autosupport -d <directory_to_store_reports>
```

By default, Autosupport reports are stored in the `~/opt/nedge/var/lib/nef/reports` directory on each node.

To specify an external destination to send aggregated Autosupport information:

1. Log in to the NexentaEdge deployment workstation.
2. Generate a key and certificate for your server.

```
$ openssl req -new -x509 -days 30 -keyout server.key -out server.pem
```

3. Follow the prompts to generate the key file.
4. Set read/write permissions for the key file.

```
$ chmod 0600 server.key
```
5. Use the following command to specify the external destination to send the Autosupport reports:

```
$ autosupport-server --url <external_destination> --interval <report_interval_in_seconds> [--dir <reports_directory>] [-D]
```

where:

- `<external_destination>` specifies the external URL where the Autosupport reports will be sent.
- `<report_interval_in_seconds>` sets the number of seconds in between reports. The minimum interval is 60 seconds.
- `<reports_directory>` specifies the location on the server where the Autosupport report files are located.
- `-D` causes the process that sends the Autosupport reports to the external location to run in the background.

For example, the following command configures NexentaEdge to send Autosupport reports to `https://logcollector.nexenta.com` at 180-second intervals. The reports are sent from the directory `/opt/nedge/reports`.

```
$ autosupport-server --url https://logcollector.nexenta.com --interval 180 --dir /opt/nedge/reports
```

To list the configured Autosupport settings:

1. From the NexentaEdge deployment workstation, enter the following command:

```
$ neadm system autosupport list
```

System response:

```
Reports Directory:       /opt/nedge/var/lib/nef/reports
Report Interval:         3600
Report Timeout:          60
```
Enabling SNMP Traps

NexentaEdge can generate SNMP traps when the following events occur:

- When a server is added or removed
- When a VDEV is added or removed

To enable SNMP traps for NexentaEdge:

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).
2. Install the required packages.
   - For Ubuntu installations, enter:
     
     ```
     $ apt-get install snmpd snmp
     ```
   - For RedHat/CentOS installations, enter:
     
     ```
     $ yum install net-snmp net-snmp-utils net-snmp-devel net-snmp-agent-libs
     ```
3. Edit settings in `/etc/snmp/snmpd.conf` as appropriate for your installation. For example:
   
   ```
   trap2sink 127.0.0.1 testCommunity
   rocommunity public
   master agentx
   agentXSocket unix:/var/snmp/agentx
   ```
4. Restart the `snmpd` server.
5. To test SNMP traps, open another terminal and start `auditserv` and `snmptrapd`.
   
   ```
   $ snmptrapd -CfLo --disableAuthorization=yes
   ```

Traps will be visible when adding or removing servers or VDEVs.
Managing Logical Clusters

This chapter includes the following topics:

- Overview of Logical Cluster Administration
- Managing Clusters
- Managing Tenants
- Managing Buckets

Overview of Logical Cluster Administration

From a physical standpoint, a NexentaEdge cluster is a collection of servers connected by one or more 10 Gigabit switches. The previous chapters explain how to set up the switches and how to add servers and storage devices to the cluster.

From a logical standpoint, a NexentaEdge cluster is a group of data nodes that store objects and gateway nodes that provide access to those objects. To external clients, the NexentaEdge cluster appears to be an iSCSI or OpenStack Cinder target, or an OpenStack Swift or Amazon S3 object storage system, depending on the storage services enabled for the cluster.

To configure NexentaEdge to provide one of these storage services, you create a logical cluster, add one or more tenants to the cluster, and enable a storage service for a tenant, specifying which server nodes provide access to clients for the storage service.

The procedures in this chapter show how to use the NEADM tool to add and delete clusters, tenants, and buckets. After you perform these tasks, you can enable and configure storage services. Configuration of the individual storage services is covered in the following chapters.

Erasure Coding Support

Erasure Coding is a technique used by storage devices similar to RAID to provide chunk parity protections and can greatly reduce the capacity footprint required by enterprise standard triple replication for data resiliency. It can reduce the footprint dramatically, providing the same resiliency as triple protection while additionally allowing for distributed fault protection and rebuilding of data.

Erasure Coding is implemented as part of version 2.0 of NexentaEdge. We intend to support multiple algorithms and protection schemes including Reed-Solomon, Cauchy and XOR; however only Reed-Solomon is initially released.

A protection scheme is defined as “<K>:<M>:<algorithm>“. For example, “4:2:rs” defines a protection scheme where for each 4 data chunks added 2 parity chunks are generated by the Reed-Solomon algorithm, and all of them will be located in different failure domains.

The replication factor must always be at least the number of parity chunks + 1; in this example we require 3 replicas on ingest.
NexentaEdge implements Erasure Coding once an object has come to rest. This complements traditional mirroring (replication) by allowing ingest data to be replicated in a highly performant manner utilizing high speed Replicast, inline deduplication and compression, and later encoded where space is reclaimed by removal of unnecessary replicas after the protection scheme’s properties have been satisfied.

Currently supported models:
- 4:2:rs  [4 chunk : 2 parity : Reed-Solomon]
- 6:3:rs  [6 chunk : 3 parity : Reed-Solomon]
- 9:3:rs  [9 chunk : 3 parity : Reed-Solomon]

Managing Clusters

Using the NEADM tool, you can create logical clusters, list them, and delete them.

- **To create a cluster:**
  1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).
  2. Use the following command to create a cluster:
     
     ```
     neadm cluster create <cluster-name>
     ```
     
     Example:
     ```
     neadm cluster create clu1
     ```
     
     System response:
     ```
     Cluster clu1 created successfully.
     ```

- **To list the available clusters:**
  1. From the NexentaEdge deployment workstation, use the following command to list the available logical clusters:
     ```
     neadm cluster list
     ```
     
     System response:
     ```
     NAME
     clu1
     ```

- **To delete a cluster:**
  1. From the NexentaEdge deployment workstation, use the following command to delete a cluster:
     ```
     neadm cluster delete <cluster-name>
     ```
     
     Example:
     ```
     neadm cluster delete clu1
     ```
     
     System response:
     ```
     Cluster clu1 deleted successfully.
     ```
To set erasure coding options for the cluster:

1. From the NexentaEdge deployment workstation, use the following command to enable erasure coding for the cluster:

   ```bash
   $ neadm cluster create <cluster-name> -o <options>
   ```

   where `<options>` is a comma-separated list of object-related options that can include the erasure coding options listed below.

   Note that object options passed through `-o <options>` are inherited by tenants and buckets and objects unless overridden on creation time or through subsequent metadata modification.

   - **ccow-ec-enabled**
     Enables (1) or disables (0) erasure coding for objects in the cluster.

   - **ccow-ec-data-mode**
     Sets the protection scheme for erasure coding.
     A protection scheme is defined as `<K>:<M>:<algorithm>`
     where:
     - `<K>` is the number of original data chunks.
     - `<M>` is the number of redundant (or parity) chunks added to original data chunks to provide data protection. The more parity chunks added, than more failures can be passed without data loss.
     - `<algorithm>` is the type of erasure coding to be used. NexentaEdge supports the Reed-Solomon algorithm, specified as `rs`.
     For example, `4:2:rs` defines a protection scheme where for each 4 data chunks 2 parity chunks are added, generated by a Reed-Solomon algorithm.
     You can specify one of the following protection schemes: `4:2:rs, 6:3:rs, 9:3:rs`.

   Example:
   ```bash
   $ neadm cluster create clu1 -o ccow-ec-enabled=1,ccow-ec-data-mode=4:2:rs
   ```

To list the erasure coding settings for a cluster:

1. From the NexentaEdge deployment workstation, use the following command to list the cluster settings:

   ```bash
   $ neadm cluster list <cluster-name> -x
   ```

   Example:
   ```bash
   $ neadm cluster list clu1 -x
   System response:
   CLUSTER EC_ON EC_MODE REPCNT
   clu1    0     6:3:rs    3
   ```
To set the replication count for objects in the cluster:

1. From the NexentaEdge deployment workstation, enter the following command:
   
   ```bash
   $ neadm cluster create <cluster-name> -o ccow-replication-count=<number>
   ```
   
   where `<number>` is the number of copies of all objects to be stored in different failure domains. Default is 3 copies; maximum is 4 copies.
   
   Example:
   
   ```bash
   $ neadm cluster create clu1 -o ccow-replication-count=2
   ```

To set the chunk size for objects in the cluster:

1. From the NexentaEdge deployment workstation, enter the following command:
   
   ```bash
   $ neadm cluster create <cluster-name> -o ccow-chunkmap-chunk-size=<bytes>
   ```
   
   where `<bytes>` is the chunk size for objects in the cluster. You can specify from 4096 bytes to 4194304 bytes in increments of 2 bytes.
   
   Example:
   
   ```bash
   $ neadm cluster create <cluster-name> -o ccow-chunkmap-chunk-size=8192
   ```

### Enabling and Disabling Syslog

By default, information about NexentaEdge events is written to the system log. This allows the information to be monitored with tools that support syslog format as described in RFC 5424. If necessary, you can disable this functionality.

To enable syslog reporting on a data node:

1. Enter the following command on the data node:
   
   ```bash
   $ echo "export CCOW_LOG_SYSLOG=1" >> /opt/nedge/.local
   ```
   
   2. Restart all services on the data node
   
   ```bash
   $ service nedge restart
   ```

To disable syslog reporting for NexentaEdge events:

1. From the NexentaEdge deployment workstation, enter the following command:
   
   ```bash
   $ neadm cluster create <cluster-name> -o CCOW_LOG_SYSLOG=0
   ```
   
   Example:
   
   ```bash
   $ neadm cluster create clu1 -o CCOW_LOG_SYSLOG=0
   ```

To re-enable syslog reporting for NexentaEdge events:

1. From the NexentaEdge deployment workstation, enter the following command:
   
   ```bash
   $ neadm cluster create <cluster-name> -o CCOW_LOG_SYSLOG=1
   ```
   
   Example:
   
   ```bash
   $ neadm cluster create clu1 -o CCOW_LOG_SYSLOG=1
   ```
Enabling Core Dumps

You can enable core dumps on data nodes. The core dumps will be located in the /opt/nedge/var/cores directory for NexentaEdge nodes, and in the /disks/services/container_id/var/cores directory for service containers.

- To enable core dumps on a data node:
  1. Enter the following command on the data node:
     ```bash
     $ echo "export CCOWD_COREDUMP=1" >> /opt/nedge/.local
     ```

Managing Tenants

After you create a cluster, you add one or more tenants to it. Using the NEADM tool, you can create tenants, list them, delete them, and configure identity services for them.

- To add a tenant to a cluster:
  1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).
  2. Use the following command to add a tenant to a cluster:
     ```bash
     $ neadm tenant create <cluster-name>/<tenant-name>
     ```
     Example:
     ```bash
     $ neadm tenant create clu1/ten1
     ```
     System response:
     Tenant ten1 created successfully.

- To list the tenants within a cluster:
  1. From the NexentaEdge deployment workstation, use the following command to list the tenants within a cluster:
     ```bash
     $ neadm tenant list
     ```
     Example:
     ```bash
     $ neadm tenant list
     ```
     System response:
     CLUSTER TENANT
     clu1    root
     clu1    ten1

- To delete a tenant from a cluster:
  1. From the NexentaEdge deployment workstation, use the following command to delete a tenant:
     ```bash
     $ neadm tenant delete <cluster-name>/<tenant-name>
     ```
Example:

```
$ neadm tenant delete clu1/ten1
System response:
Tenant clu1 deleted successfully.
```

❖ **To set erasure coding options for the tenant:**

1. From the NexentaEdge deployment workstation, use the following command to enable erasure coding for the tenant:

```
$ neadm tenant create <cluster-name>/<tenant-name> -o <options>
```

where `<options>` is a comma-separated list of object-related options that can include the erasure coding options listed below.

Note that object options passed through `-o <options>` are inherited by tenants and buckets and objects unless overridden on creation time or through subsequent metadata modification.

- **ccow-ec-enabled**
  Enables (1) or disables (0) erasure coding for objects in the tenant.
- **ccow-ec-data-mode**
  Sets the protection scheme for erasure coding.
  A protection scheme is defined as `<K>:<M>:<algorithm>`
  where:
  - `<K>` is the number of original data chunks.
  - `<M>` is the number of redundant (or parity) chunks added to original data chunks to provide data protection. The more parity chunks added, than more failures can be passed without data loss.
  - `<algorithm>` is the type of erasure coding to be used. NexentaEdge supports the Reed-Solomon algorithm, specified as `rs`.
  For example, `4:2:rs` defines a protection scheme where for each 4 data chunks 2 parity chunks are added, generated by a Reed-Solomon algorithm.
  You can specify one of the following protection schemes: `4:2:rs, 6:3:rs, 9:3:rs`.

Example:

```
$ neadm tenant create clu1/ten1 -o ccow-ec-enabled=1,ccow-ec-data-mode=4:2:rs
```

❖ **To list the erasure coding settings for a tenant:**

1. From the NexentaEdge deployment workstation, use the following command to list the tenants within a cluster:

```
$ neadm tenant list <cluster-name>/<tenant-name> -x
```

Example:

```
$ neadm tenant list clu1/ten1 -x
```
To set the replication count for objects for the tenant:

1. From the NexentaEdge deployment workstation, enter the following command:

   ```shell
   $ neadm tenant create <cluster-name>/<tenant-name>
   -o ccow-replication-count=<number>
   ```

   where `<number>` is the number of copies of all objects to be stored in different failure domains. Default is 3 copies; maximum is 4 copies.

   Example:

   ```shell
   $ neadm tenant create clu1/ten1 -o ccow-replication-count=2
   ```

Configuring Identity Services for Tenants

To control access to objects stored for a given tenant, you can configure usernames / passwords within the tenant. This allows you to limit access to objects to users that supply the correct credentials.

You can optionally configure the user credentials to be checked with external identity servers. NexentaEdge supports Active Directory (AD), LDAP, and Keystone authentication. The identity service configuration is done on a per-tenant basis, so individual tenants can have different identity service settings.

The system periodically checks a user’s validity with the identity server and blocks access if the user’s credentials become invalid.

To create a user within a tenant:

1. From the NexentaEdge deployment workstation, use the following command to create a user:

   ```shell
   $ neadm tenant user create <cluster-name>/<tenant-name> <user> <password> <gw-type> [administrator]
   ```

   where:

   - `<user>` Is the name of the new user.
   - `<password>` Is the password for the new user.
   - `<gw-type>` Specifies the gateway type. This can be `s3`, `swift`, or `object`. The `object` setting allows the user to access both OpenStack Swift and Amazon S3 services.
   - `administrator` Indicates that the user has an administrative role.

   Example:

   ```shell
   $ neadm tenant user create clu1/ten1 usrl password111 s3 administrator
   System response:
   S3 user usrl created successfully
   Access key: ARIB4POL4SMY0YCCA13R
   Secret key: iXmDqaIVhkOn6FUob1ITacpjT1mLBQvzbX7wi9R
   ```
To list the users within a tenant:

1. From the NexentaEdge deployment workstation, use the following command to list the users configured for a tenant:

```
$ neadm tenant user list <cluster-name>/<tenant-name> [-c <max-number-of-users>] [-n <user-to-start-list>]
```

Example:

```
$ neadm tenant user list clu1/ten1 -c 1 -n usr
```

System response:

```
NAME | TYPE | IDENTITY | ADMINISTRATOR
----- |------ |---------- |----------------
usr1  | s3   | nedge    | Y
```

To configure an AD identity service for a tenant:

1. From the NexentaEdge deployment workstation, use the following command:

```
$ neadm tenant identity create <cluster-name>/<tenant-name> ad
    'url:<ad_url>,baseDN:<baseDN>,username:<ad_admin_user>,
    password:<ad_admin_password>''
```

Note the double-quotes that enclose the AD configuration and the backslash that continues the configuration to the next line.

Example:

```
$ neadm tenant identity create clu1/ten1 ad
    "url:ldap://my.ad.com,baseDN:dc=my,dc=com,username:Admin@my.com,password:passadmin"
```

To configure an LDAP identity service for a tenant:

1. From the NexentaEdge deployment workstation, use the following command:

```
$ neadm tenant identity create <cluster-name>/<tenant-name> ldap
    'url:<ldap_url>:<ldap_port>,
    adminDN:<admin_DN>,
    password:<admin_password>,
    userBaseDN:<user_base_dn>,
    userAttr:<user_attribute>,
    userObjectClass:<user_object_class>,
    groupBaseDN:<group_base_dn>,
    groupAttr:<group_attribute>,
    groupObjectClass:<group_object_class>,
    memberAttr:<member_attribute>''
```

Note the double-quotes that enclose the LDAP configuration and the backslash that continues the configuration to the next line.

Example:

```
$ neadm tenant identity create clu1/ten1 ldap
    "url:ldap://my.ldap.com:389,adminDN:cn=admin,dc=example,dc=com,password:passadmin,userBaseDN:cn=users,ou=groups,dc=example,dc=com,userAttr:uid,"
To configure a Keystone identity service for a tenant:

1. From the NexentaEdge deployment workstation, use the following command:

   ```
   $ neadm tenant identity create <cluster-name>/<tenant-name> keystone
   "host: <keystone_host>,
   port: <keystone_port>,
   protocol: <protocol_type>,
   admin_domain_name: <admin_domain_name>,
   admin_user: <admin_user>,
   admin_password: <admin_password>,
   domain_name: <domain_name>"
   ```

   Note the double-quotes that enclose the Keystone configuration and the backslash that continues the configuration to the next line.

   Example:

   ```
   $ neadm tenant identity create clu1/ten1 keystone
   "host: my.keystone.com,
   port: 35357,
   protocol: http,
   admin_domain_name: Default,
   admin_user: admin,
   admin_password: passadmin,
   domain_name: Default"
   ```

To list the configured identity services for a tenant:

1. From the NexentaEdge deployment workstation, use the following command to list the identity services:

   ```
   $ neadm tenant identity list <cluster-name>/<tenant-name>
   ```

   Example:

   ```
   $ neadm tenant identity list clu1/ten1
   ```

   System response:

   ```
   IDENTITY CONFIG
   keystone
   ldap
   ad
   ```

To delete an identity service for a tenant:

1. From the NexentaEdge deployment workstation, use the following command to delete an identity service:

   ```
   $ neadm tenant identity delete <cluster-name>/<tenant-name> <id-type>
   ```

   Example:

   ```
   $ neadm tenant identity delete clu1/ten1 ldap
   ```
To register an LDAP, AD, or Keystone user within a tenant:

1. From the NexentaEdge deployment workstation, use the following command to register a user:

   ```
   $ neadm tenant user register <cluster-name>/<tenant-name> <user> <password> <gw-type> <id-type> [administrator]
   ```

   where:

   - `<user>` is the name of the user to register.
   - `<password>` is the password for the new user.
   - `<gw-type>` specifies the gateway type. This can be `s3`, `swift`, or `object`. The `object` setting allows the user to access both OpenStack Swift and Amazon S3 services.
   - `<id-type>` specifies the identity service type. This can be `ad`, `ldap`, or `keystone`.
   - `administrator` indicates that the user has an administrative role. Administrative users can set ACLs for other users.

   Example:

   ```
   $ neadm tenant user register clu1/ten1 usr1 password111 object ldap administrator
   ```

   System response:

   object user usr1 registered successfully
   Access key: ARIB4POL4SMY0YCCA13R
   Secret key: iXmDqaIVhkOn6FUoblItacpjTlmLBQvzbX7wi9R

Managing Buckets

Buckets are containers within a tenant namespace that provide organizational structure for objects. Objects can be stored within buckets. Using the NEADM tool, you can create and delete buckets, list the buckets created for a tenant, and set quotas for the size and number of objects in a bucket.

To add a bucket within a tenant:

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).

2. Use the following command to add a bucket for a tenant:

   ```
   $ neadm bucket create <cluster-name>/<tenant-name>/<bucket-name>
   ```

   Example:

   ```
   $ neadm bucket create clu1/ten1/buk1
   ```

   System response:

   Bucket buk1 created successfully.
To list the buckets created for a tenant:

1. From the NexentaEdge deployment workstation, use the following command to list the buckets for a tenant:

   $ neadm bucket list <cluster-name>/<tenant-name>

   Example:
   
   $ neadm bucket list clu1/ten1

   System response:
   
   PATH      OBJECTS
   clu1/ten1 buk1

To delete a bucket:

1. From the NexentaEdge deployment workstation, use the following command to delete a bucket:

   $ neadm bucket delete <cluster-name>/<tenant-name>/<bucket-name>

   Example:
   
   $ neadm bucket delete clu1/ten1/buk1

   System response:
   
   Bucket buk1 deleted successfully.

To set erasure coding options for the bucket:

1. From the NexentaEdge deployment workstation, use the following command to enable erasure coding for the bucket:

   $ neadm bucket create <cluster-name>/<tenant-name>/<bucket-name>
   -o <options>

   where <options> is a comma-separated list of object-related options that can include the erasure coding options listed below.

   Note that object options passed through -o <options> are inherited by tenants and buckets and objects unless overridden on creation time or through subsequent metadata modification.

   ccow-ec-enabled Enables (1) or disables (0) erasure coding for objects in the bucket.
**Example:**

$ neadm bucket create clu1/ten1/buk1 -o ccow-ec-enabled=1,ccow-ec-data-mode=4:2:rs

**To list the erasure coding settings for a bucket:**

1. From the NexentaEdge deployment workstation, use the following command to list the buckets for a tenant:

   $ neadm bucket list <cluster-name>/<tenant-name> -x

   **Example:**

   $ neadm bucket list clu1/ten1 -x

   **System response:**

   CLUSTER/TENANT  EC_ON  EC_MODE  REPCNT  
   clu1/ten1       0       6:3:rs    3

**To set quotas for the size and number of objects that can exist in a bucket:**

1. For a new bucket, use the following command to set quotas:

   $ neadm bucket create <cluster-name>/<tenant-name>/<bucket-name> -o <options>

   **where** <options> **is a comma-separated list of quota options**, described below.

   - **quota**
     Specifies the amount of space allocated to the bucket.
     To specify the space in gigabytes, use the GB suffix (for example, 2GB to specify 2 gigabytes). If you do not specify a suffix, a size in bytes is assumed.

   - **quota-count**
     Sets the maximum number of objects that can be placed in a bucket.
Example:

```
$ neadm bucket create clu1/ten1/buk1 -o quota=1G,quota-count=200
```

2. For an already existing bucket, use the following commands to set quotas:

```
$ neadm bucket quota <cluster-name>/<tenant-name>/<bucket-name> <size>
$ neadm bucket quota-count <cluster-name>/<tenant-name>/<bucket-name> <count>
```

where:

- `<size>` Specifies the amount of space allocated to the bucket. To specify the space in gigabytes, use the GB suffix (for example, 2GB to specify 2 gigabytes). If you do not specify a suffix, a size in bytes is assumed.

- `<count>` Sets the maximum number of objects that can be placed in a bucket.

Example:

```
$ neadm bucket create clu1/ten1/buk1 -o quota=1000,quota-count=100
$ neadm bucket quota clu1/ten1/buk2 1g
$ neadm bucket quota-count clu1/ten1/buk2 1000
```

❖ To set the replication count for objects in the bucket:

1. From the NexentaEdge deployment workstation, enter the following command:

```
$ neadm bucket create <cluster-name>/<tenant-name>/<bucket-name> -o ccow-replication-count=<number>
```

   where `<number>` is the number of copies of all objects to be stored in different failure domains. Default is 3 copies; maximum is 4 copies.

Example:

```
$ neadm bucket create clu1/ten1/buk1 -o ccow-replication-count=2
```
Configuring an iSCSI Storage Service Group

This chapter includes the following topics:

- Overview
- Before You Begin
- Setting Up an iSCSI Storage Service Group
- Setting Resource Limits for the iSCSI Storage Service Group
- Setting iSCSI Target Parameters
- Configuring iSCSI Portals
- Configuring CHAP Authentication for Initiators
- Managing LUNs
- Managing Snapshots
- Configuring an iSCSI Storage Service Group for High Availability
- Managing NBD Devices

Overview

This chapter describes how to set up an iSCSI storage service group, which allows a NexentaEdge cluster to operate as an iSCSI storage system. In an iSCSI storage service group, a gateway node provides iSCSI target functionality for iSCSI initiators, and storage within the cluster is configured as LUNs that are available on the iSCSI target. Using the NEADM tool, you can create and manage LUNs, create snapshots of LUNs, and clone existing LUNs into new ones.

You can deploy an iSCSI storage service group in a high-availability (HA) configuration, with two gateway nodes handling initiator requests sent to a virtual IP (VIP) address; one node actively handles the requests and the other serves as a backup.

Before You Begin

Before setting up an iSCSI storage service group, verify the following:

- NexentaEdge has been deployed on the servers in the cluster that will provide iSCSI target functions (gateway nodes) and the servers that will provide storage (data nodes).
- The servers to be used as gateway nodes (iSCSI targets) have network connectivity to the iSCSI initiators and to the data nodes in the NexentaEdge cluster.
• A NexentaEdge logical cluster has been created, and the logical cluster has at least one tenant. Enter the `neadm tenant list` command on the deployment workstation to verify.

• Within the tenant, at least one bucket has been created. Use the `neadm bucket list` command to verify.

### Setting Up an iSCSI Storage Service Group

**To configure an iSCSI storage service group:**

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).

2. Use the following command to create an iSCSI storage service group:
   
   ```
   neadm service create iscsi <service-group>
   ```

   Example:
   
   ```
   neadm service create iscsi isc01
   ```

   System response:
   
   Service isc01 created

3. List the IDs of the nodes in the NexentaEdge cluster:

   ```
   neadm system status
   ```

   System response:

<table>
<thead>
<tr>
<th>ZONE:HOST:CID</th>
<th>SID</th>
<th>UTIL</th>
<th>CAP</th>
<th>CPU</th>
<th>MEM</th>
<th>DEVs</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:node32:</td>
<td>B5DEFB9B3C690...</td>
<td>0%</td>
<td>160G</td>
<td>2/0.2@2.34Ghz</td>
<td>4.16G/7.8G</td>
<td>4/4</td>
<td>ONLINE</td>
</tr>
<tr>
<td>0:node33:</td>
<td>[M] B3EF9C30B6F79F...</td>
<td>0%</td>
<td>160G</td>
<td>2/0.1@2.34Ghz</td>
<td>4.18G/7.8G</td>
<td>4/4</td>
<td>ONLINE</td>
</tr>
<tr>
<td>0:node34:</td>
<td>155B81C291FA4E...</td>
<td>0%</td>
<td>160G</td>
<td>2/0.1@2.34Ghz</td>
<td>4.18G/7.8G</td>
<td>4/4</td>
<td>ONLINE</td>
</tr>
<tr>
<td>0:node35:</td>
<td>9BB624B94C8E2C...</td>
<td>0%</td>
<td>160G</td>
<td>2/0.1@2.34Ghz</td>
<td>4.15G/7.8G</td>
<td>4/4</td>
<td>ONLINE</td>
</tr>
<tr>
<td>0:node36:</td>
<td>F88C6C3180C34E...</td>
<td>0%</td>
<td>160G</td>
<td>2/0.1@2.34Ghz</td>
<td>4.15G/7.8G</td>
<td>4/4</td>
<td>ONLINE</td>
</tr>
</tbody>
</table>

4. Copy the ID or hostname of the server node that you want to use as a gateway node (that is, as an iSCSI target). If you are deploying the iSCSI storage service group in a [high-availability (HA) configuration](#), you can specify two nodes to be gateway nodes by configuring a virtual IP address (VIP) for the storage service group.

5. Use the following command to make the node a gateway node for the service group. This allows the node to function as an iSCSI target.

   ```
   neadm service add <service-group> <server-id>
   ```

   Example:

   ```
   service add isc01 D586BF84009C4230BFE31CAC961197D
   ```

   System response:

   Service isc01 added to D586BF84009C4230BFE31CAC961197D

6. For Docker deployments, you must configure the Docker network that links the containers to the node on which they are contained.
The following commands show an example of how to configure a Docker network named `client-net` using the Macvlan network driver. You can use other network drivers if necessary.

```
$ ifconfig enp2s0f1 192.168.10.201/24 up
$ modprobe macvlan
$ docker network create -d macvlan --subnet 192.168.10.0/24 --gateway 192.168.10.1 -o parent=enp2s0f1 client-net
$ docker run --net=client-net --ip=192.168.10.11 -it alpine /bin/sh
```

7. For Docker deployments, list the available Docker networks and determine which one you want to attach to the iSCSI storage service group.

   Example:

```
$ docker network ls
```

System response:

<table>
<thead>
<tr>
<th>NETWORK ID</th>
<th>NAME</th>
<th>DRIVER</th>
<th>IPAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>7fca4eb8c647</td>
<td>client-net</td>
<td>bridge</td>
<td></td>
</tr>
<tr>
<td>9f904ee27bf5</td>
<td>none</td>
<td>null</td>
<td></td>
</tr>
<tr>
<td>cf03ee007fb4</td>
<td>host</td>
<td>host</td>
<td></td>
</tr>
<tr>
<td>78b03ee04fc4</td>
<td>multi-host</td>
<td>overlay</td>
<td></td>
</tr>
</tbody>
</table>

8. For Docker deployments, determine subnet information for the selected Docker network.

   Example:

```
$ docker network inspect client-net
```

System response:

```
[
  {
    "Name": "client-net",
    "Id": "b2b1a2cba71761d984383fd68218c70bbbd17d328496885f7c98b0f",
    "Scope": "local",
    "Driver": "bridge",
    "IPAM": {
      "Driver": "default",
      "Config": {
        "Subnet": "192.168.200.1/16",
        "Gateway": "192.168.200.21"
      }
    }
  }
],
```

9. For Docker deployments, display the ID of the gateway node configured for the storage service group.

   Example:

```
$ neadm service show isc01
```
10. Using the ID from the previous step, add the Docker network to the gateway node. Note the use of quotes between the network name and IP address.

Example:

```
$ neadm service configure isc01 X-Container-Network-1985F7E1202CB92F7491B3 "client-net --ip 192.168.200.1"
```

To specify multiple Docker networks, separate them with semicolons; for example:

```
$ neadm service configure isc01 X-Container-Network-1985F7E1202CB92F7491B3 "client-net --ip 192.168.200.1;local-net --ip 192.168.1.23"
```

Note that the client network IP address ranges must be on the same subnet as the physical client interfaces

11. Use the following command to apply the iSCSI storage service group to a specific cluster.

```
$ neadm service serve <service-group> <cluster>
```

Example:

```
$ neadm service serve isc01 clu1
```

System response:

```
Success
```

12. Use the following command to enable the iSCSI storage service group:

```
$ neadm service enable <service-group>
```

Example:

```
$ neadm service enable isc01
```
13. Use the following command to verify that the iSCSI storage service group is enabled.

```
$ neadm service list
```

System response:

```
TYPE  NAME  SERVERID                        STATUS
iscsi  isc01 1985F7E1202CB92F7491B3        enabled
```

14. Use the following command display the properties of the iSCSI storage service group.

```
$ neadm service show <service-group>
```

Example:

```
$ neadm service show isc01
```

System response:

```
FIELD                                                 VALUE
X-ContainerId-0142045DEF9151492F6B60DE76240819        59b98d745138
X-ContainerIPv6-0142045DEF9151492F6B60DE76240819      fd00:ea88:dbf1:3411
X-Service-Name                                       isc01
X-Service-Type                                       iscsi
X-Service-Worker                                     iscsiserv
X-Number-Of-Versions                                 1
X-Auth-Type                                          disabled
X-Description                                        iSCSI Target
X-ISCSI-TargetID                                     44321
X-ISCSI-TargetName                                   ign.2005-11.example.com:
X-ISCSI-AllowedInitiatorAddresses                    ALL
X-Container-Network-0142045DEF9151492F6B60DE76240819  client-net --ip 10.16.11.21
X-Servers                                           1985F7E1202CB92F7491B3
X-Container-Network-1985F7E1202CB92F7491B3           client-net --ip 10.16.110.21
X-VIPS                                               
X-VIP-Version                                        1
X-VIP-Chat                                           224.1.1.1:19874
X-VIP-QuorumNode                                     nx16-edge.xmp.corp
X-VIP-Nodes                                          
X-ContainerId-1985F7E1202CB92F7491B3535FEE1CF7       8b06e28864f5
X-ContainerIPv6-1985F7E1202CB92F7491B3535FEE1CF7     fd00:5238d3bf:ae0e
X-Status                                             enabled
X-Container-Hostname-0142045DEF9151492F6B60DE76240819 2b312804a86e3d15
X-Container-Hostname-1985F7E1202CB92F7491B3535FEE1CF7 1486ee5c5f87de80
```

Service Group Objects:

```
[
```

Setting Resource Limits for the iSCSI Storage Service Group

You can optionally configure limits for memory and CPU usage for containers used by the storage service group. This allows you to limit resource consumption on nodes, as well as set hard limits for nodes that have many containers or services, or in hyper-converged setups.

- **To set memory and CPU resource limits for containers used by an iSCSI storage service group:**

  1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).

  2. Use the following command to limit the amount of memory for a container used by a specified storage service group:
$ neadm service configure <service-group> X-Service-Mem-Limit <bytes>
Example:
$ neadm service configure isc01 X-Service-Mem-Limit 64G
System response:
Service isc01 updated successfully.
Where:

<bytes> Sets a limit in bytes provided to the container before the kernel puts back-pressure on allocations, inclusive of all caches in the service and container.

3. Use the following command to limit the CPU resources for a container used by a specified storage service group:

$ neadm service configure <service-group> X-Service-CPU-Limit <cores>
Example:
$ neadm service configure isc01 X-Service-CPU-Limit 4
System response:
Service isc01 updated successfully.
Where:

<cores> Is the number of cores dedicated to the container before the kernel puts back-pressure on scheduling time for the processes in the container. This is enforced by cgroups, not NUMA.

4. If the storage service group is currently enabled, use the following command to restart (disable and re-enable) it. Note that any in-progress upload/download operations are aborted during the restart; consequently, you may want to disable user access during service configuration.

$ neadm service restart <service-group>
Example:
$ neadm service restart isc01
System response:
Service isc01 restarted

5. If the storage service group is currently disabled, use the following command to enable it.

$ neadm service enable <service-group>
Example:
$ neadm service enable isc01
System response:
Service isc01 enabled
Setting iSCSI Target Parameters

You can set iSCSI target parameters that apply to this iSCSI storage service group.

➢ To set iSCSI target parameters:
  1. Log in to the NexentaEdge deployment workstation.
  2. Use the following command to set an iSCSI target parameter for an iSCSI storage service group:

     $ neadm iscsi config set <service-group> <target-parameter> <value>

    Example:

     $ neadm iscsi config set isc01 MaxQueueCmd 512

    Where:

    <service-group> Is the iSCSI storage service group for which the parameter is being set.
    <target-parameter> Is an iSCSI target parameter to be set.
    <value> Is the value to be set for the iSCSI target parameter.

    System response:
    iSCSI Target configured successfully.

To list the iSCSI target parameters you can set with this command, use the neadm iscsi help command. To list the current values of the iSCSI target parameters, use the neadm iscsi config read <service-group> command.

Configuring iSCSI Portals

Using NEADM, you can add, remove, and list iSCSI portals.

➢ To configure iSCSI portals
  1. Log in to the NexentaEdge deployment workstation.
  2. Use the following command to add a portal for an iSCSI storage service group:

     $ neadm iscsi portal add <service-group> <ipaddr>:<port>

    Example:

     $ neadm iscsi portal add isc01 172.16.0.1:9090

    Where:

    <service-group> Is the iSCSI storage service group for which the portal is being configured.
    <ipaddr>:<port> Are the IP address and port number for the portal.
System response:

to delete an existing iSCSI portal, use the `neadm iscsi portal remove` command. To list the existing iSCSI portals, use the `neadm iscsi portal list` command.

Configuring CHAP Authentication for Initiators

You can configure the iSCSI service group to use CHAP authentication for initiators. The initiator must supply a valid username/password prior to being granted access to devices.

- **To configure CHAP authentication for an iSCSI service group:**

  1. Log in to the NexentaEdge deployment workstation.
  2. Use the following command to add a portal for an iSCSI storage service group:

     ```
     $ neadm iscsi accounts add <service-group> <username> <password>
     ```

     Example:

     ```
     $ neadm iscsi accounts add isc01 init01 Password
     ```

     System response:

     iSCSI account configured successfully.

     To delete an existing username/password for the iSCSI service group, use the `neadm iscsi accounts remove` command. To list the existing usernames, use the `neadm iscsi accounts list` command.

Managing LUNs

Using the NEADM tool, you can create LUNs, which are made available to iSCSI initiators via the iSCSI target enabled by the iSCSI storage service group. Existing LUNs can be resized or deleted.

| Note: | If you are deploying NexentaEdge in an HA configuration, you must create the VIP(s) before creating LUNs. See Configuring an iSCSI Storage Service Group for High Availability. |

- **To create a LUN:**

  1. Log in to the NexentaEdge deployment workstation.
  2. Use the following command to create a LUN for an iSCSI storage service group:

     ```
     ```

     Example:

     ```
     $ neadm iscsi create isc01 clu1/ten1/buk1/LUN01 1G
     ```
Where:

- `<service-group>` is the iSCSI storage service group for which the LUN is being created.
- `<lun-path>` associates the LUN with a specific cluster/tenant/bucket combination for this iSCSI storage service group.
- `<size>` is the size of the new LUN. To specify a size in gigabytes, use the GB suffix (for example, 2GB to specify 2 gigabytes). If you do not specify a suffix, a size in bytes is assumed. Maximum LUN size is 16 petabytes.
- `-b <block-size>` is the block size for the new LUN. The default is 4096 bytes. You can change this in increments of 512 bytes, with a minimum block size of 512 bytes. For ESX, this should be set to 512 bytes.
- `-s <chunk-size>` sets the size of the chunks in the object that backs the LUN; NexentaEdge breaks objects stored for this LUN into chunks of this size. The default is 131,072 bytes. You can specify the chunk size in $2^n$-byte increments; for example; 8192, 16384 and so on. The minimum chunk size is 8,192 bytes, and the maximum is 1,048,576 bytes.
- `-n <lun-number>` is the LUN number for the new LUN. You can specify a number from 1–255.
- `-o <option>` specifies CSV input of metadata.
- `-r <replication-count>` sets the number of times the LUN object is replicated across the cluster. This can be set to 2, 3, or 4. The default replication count for a LUN object is 3.
- `-t <IOPS-limit>` sets a limit in 4K normalized IOPS for the LUN object.

For the `<size>`, `<block-size>`, and `<chunk-size>` parameters, you can optionally specify a suffix to indicate the units (B, KB, MB, GB, or TB). For example, to specify a size in gigabytes, use the GB suffix. If you do not specify a suffix, a size in bytes is assumed.

System response:

iSCSI LUN clu1/ten1/buk1/LUN01 created.

The combination of cluster/tenant/bucket/LUN, clu1/ten1/buk1/LUN01 in this example, is called a LUN path. You use the LUN path when referring to specific LUNs for snapshot and clone operations.

If you created a VIP for deploying the iSCSI storage service group in an HA configuration, the new LUN is associated with the VIP; iSCSI initiators can access the LUN using the VIP as the iSCSI target.

3. Use the `neadm iscsi list` command to verify that the new LUN exists for the iSCSI storage service group.

Example:

```
$ neadm iscsi list isc01
```
4. Use the following command to view the status of the iSCSI storage service group.

```bash
$ neadm iscsi status <service-group>
```

Example:

```bash
$ neadm iscsi status isc01
```

System response:

```
Server: D586BF84009C4230BFEE31CAC961197D:
  System information:
    Driver: iscsi
    State: ready
  L_T nexus information:
  LUN information:
    LUN: 0
      Type: controller
      SCSI ID: IET 00010000
      SCSI SN: beaf10
      Size: 0 MB, Block size: 1
      Online: Yes
      Removable media: No
      Prevent removal: No
      Readonly: No
      SWP: No
      Thin-provisioning: No
      Backing store type: null
      Backing store path: None
      Backing store flags:
    LUN: 1
      Type: disk
      SCSI ID: IET 00010001
      SCSI SN: beaf11
      Size: 1074 MB, Block size: 4096
      Online: Yes
      Removable media: Yes
      Prevent removal: No
      Readonly: No
      SWP: No
      Thin-provisioning: Yes
      Backing store type: ccowbd
      Backing store path: clu3/ten3/buk3/LUN01
      Backing store flags:
      Account information:
      ACL information:
        ALL
```

At this point, the iSCSI target enabled by the storage service group can be discovered by iSCSI initiators.
To resize a LUN:

1. From the NexentaEdge deployment workstation, use the following command to change the size of an existing LUN. You can only increase the size of LUNs; resizing LUNs to a smaller size is not supported.

   $ neadm iscsi resize <service-group> <lun-path> <new-size>

   Example:

   $ neadm iscsi resize isc01 clu1/ten1/buk1/LUN01 2G

   System response:

   iSCSI LUN clu1/ten1/buk1/LUN01 resized.

2. Use the `neadm iscsi list` command to verify the new size of the LUN.

   Example:

   $ neadm iscsi list isc01

   System response:

   LUN HOST SIZE BLOCK CHUNK REPCOUNT PATH
   1 newnode101 1G 4K 32K 3 clu1/ten1/buk1/LUN01

To delete a LUN:

1. From the NexentaEdge deployment workstation, use the following command to delete an existing LUN:

   $ neadm iscsi delete <service-group> <lun-path>

   Example:

   $ neadm iscsi delete isc01 clu1/ten1/buk1/LUN01

   System response:

   iSCSI LUN clu1/ten1/buk1/LUN01 deleted.

   Note that deleting a LUN deletes the listing information for the LUN, as well as its backing object. The LUN data will be lost.

   Note that for Docker deployments, the iSCSI storage service group must be enabled in order to delete a LUN.

   2. Use the `neadm iscsi list` command to verify that the LUN has been deleted.

Managing Snapshots

Using the NEADM tool, you can create snapshots (incremental backups) of LUNs configured for an iSCSI storage service group, clone LUN snapshots to new LUNs, and roll back LUNs to snapshot versions.

Note that when creating a snapshot of a LUN where you have placed a filesystem on the LUN, you must bring the LUN to rest via a filesystem sync, then take the LUN offline prior to the snapshot. Otherwise, filesystem corruption will occur.
To create a snapshot of a LUN:

1. Log in to the NexentaEdge deployment workstation.

2. Use the following command to create a snapshot of a LUN:

   ```shell
   $ neadm iscsi snapshot create <service-group> <lun-path>@<snapshot-name>
   
   Example:
   $ neadm iscsi snapshot create isc01 clu1/ten1/buk1/LUN01@snap01
   
   System response:
   Snapshot snap01 created.
   
3. Use the following command to verify that the snapshot was created for the LUN. This command lists all of the snapshots that exist for a specified LUN.

   ```shell
   $ neadm iscsi snapshot list <service-group> <lun-path>
   
   Example:
   $ neadm iscsi snapshot list isc01 clu1/ten1/buk1/LUN01
   
   System response:
   SNAPSHOT
   snap01
   
To clone a LUN snapshot to a new LUN:

1. From the NexentaEdge deployment workstation, use the following command to clone a snapshot of an existing LUN to a new LUN:

   ```shell
   $ neadm iscsi snapshot clone <service-group> <lun-path>@<snapshot-name> <new-lun-path>
   
   Example:
   $ neadm iscsi snapshot clone isc01 clu1/ten1/buk1/LUN01@snap01 clu1/ten1/buk1/LUN02
   
   System response:
   Snapshot snap01 cloned into LU clu1/ten1/buk1/LUN02.
   
2. Use the `neadm iscsi list` command to verify that the new LUN has been added to the cluster.

   Example:
   $ neadm iscsi list isc01
   
   System response:
   LUN HOST SIZE BLOCK CHUNK REPCOUNT PATH
   1 newnode101 1G 4K 32K 3 clu1/ten1/buk1/LUN01
   2 newnode101 1G 4K 32K 3 clu1/ten1/buk1/LUN02

To roll back a LUN to a snapshot version of the LUN:

1. From the NexentaEdge deployment workstation, use the following command to list all of the snapshots that exist for a specified LUN.
To replace a LUN with the version of the LUN stored in a snapshot:

```
$ neadm iscsi snapshot rollback <service-group> <lun-path>@<snapshot-name>
```

Example:
```
$ neadm iscsi snapshot rollback isc01 clu1/ten1/buk1/LUN01@snap01
```

System response:
```
iSCSI LU clu1/ten1/buk1/LUN01 rolled back to Snapshot: snap01
```

To delete a LUN snapshot:

1. From the NexentaEdge deployment workstation, use the following command to delete a LUN snapshot:

```
$ neadm iscsi snapshot delete <service-group> <lun-path>@<snapshot-name>
```

Example:
```
$ neadm iscsi snapshot delete isc01 clu1/ten1/buk1/LUN01@snap01
```

System response:
```
Snapshot snap01 deleted.
```

2. Use the following command to verify that the LUN snapshot was deleted.

```
$ neadm iscsi snapshot list <service-group> <lun-path>
```

Example:
```
$ neadm iscsi snapshot list isc01 clu1/ten1/buk1/LUN01
```

System response:
```
No snapshots found for this object
```

Configuring an iSCSI Storage Service Group for High Availability

An iSCSI storage service group can be deployed in a high-availability (HA) configuration, where two gateway nodes provide a single target for initiator requests using a virtual IP (VIP) address. The HA configuration is active-passive: all of the LUNs created for the storage service group are associated with the VIP; one of the gateway nodes provides access to the LUNs, and the other serves as a backup.
The two gateway nodes exchange heartbeat information. If the active gateway node fails, the other gateway node takes over client requests sent to the VIP.

Notes:

• The iSCSI client interface has to be up and configured on the client network.

• Enable multicast using the `route` command or `post-up route` options in network scripts. This must be done before deploying NexentaEdge.

  Example for Ubuntu:

  ```
  auto eth1
  iface eth1 inet static
  post-up route -n add -net 224.0.0.0 netmask 240.0.0.0 dev eth1
  ```

• A minimum of 4 nodes is required in an iSCSI HA configuration.

iSCSI HA Configuration Procedure

- To configure HA for an iSCSI storage service group:
  1. Log in to the NexentaEdge deployment workstation.
  2. Configure an iSCSI storage service group, and add two gateway nodes to the service group.
     
     Example:
     ```
     $ neadm service create iscsi ischA
     $ neadm service add ischA nedgenode0
     $ neadm service add ischA nedgenode1
     ```
  3. For Docker deployments, list the available Docker networks and determine which one you want to attach to the iSCSI HA configuration.
     
     Example:
     ```
     $ docker network ls
     ```
     
     System response:

     | NETWORK ID         | NAME   |
     |-------------------|--------|
     | 7fca4eb8c647       | client-net |
     | bridge            |        |
     | 9f904ee27bf5       | none   |
     | null              |        |
     | cf03ee007fb4       | host   |
     | host              |        |
     | 78b03ee04fc4       | multi-host |
     | overlay           |        |
  4. For Docker deployments, determine subnet information for the selected Docker network.
     
     Example:
     ```
     $ docker network inspect client-net
     ```
System response:

```
[
  {
    "Name": "client-net",
    "Id": "b2b1a2c8a717161d984383fd68218cf70b8bd17d32849685f7c90b0f",
    "Scope": "local",
    "Driver": "bridge",
    "IPAM": {
      "Driver": "default",
      "Config": [
        {
          "Subnet": "192.168.200.1/16",
          "Gateway": "192.168.200.21"
        }
      ]
    }
  }
],
```

5. Use the following command to add a VIP to the storage service group. Note that a storage service group can have only one VIP.

```
$ neadm service vip add <service-group> <quorum-node> <vip-address>/<netmask> [-p <preferred-node>]
```

**Example:**

```
$ neadm service vip add ischA nedgenode2 192.168.200.99/24 -p nedgenode0
```

**Where:**

- `<service-group>` is the name of the iSCSI storage service group to be configured for high availability.
- `<quorum-node>` is the hostname or server ID of a node used as a tiebreaker for split-brain situations. Note that you cannot specify an iSCSI HA node as a quorum node.
- `<vip-address>/<netmask>` is the address and network mask for the VIP.
- `-p <preferred-node>` optionally specifies the hostname or server ID of one of the nodes configured for the iSCSI storage service group to be the active gateway node for the VIP. If not specified, the first node added to the service will be selected.

**System response:**

```
VIP 192.168.200.99/24 created
```

6. For Docker deployments, display the IDs of the gateway nodes configured for the storage service group.

**Example:**

```
$ neadm service show ischA
```

**System response:**

```
FIELD       VALUE
```
7. Using the IDs from the previous step, add the Docker network to both nodes.

Example:

```
$ neadm service configure iscHA X-Container-Network-00049D822EE42F5BA5129BB3118F0502 "client-net --ip 192.168.200.1"
$ neadm service configure iscHA X-Container-Network-000406E4A4C7FF3B182976E1A351CCA9 "client-net --ip 192.168.200.2"
```

To specify multiple Docker networks, separate them with semicolons; for example:

```
$ neadm service configure iscHA X-Container-Network-000406E4A4C7FF3B182976E1A351CCA9 "client-net --ip 192.168.200.1;local-net --ip 192.168.1.23"
```

8. Enable the iSCSI storage service group.

Example:

```
$ neadm service enable iscHA
```

System response:

```
Service iscHA enabled
```

9. Create LUNs for the iSCSI storage service group.

Example:

```
$ neadm iscsi create iscHA clu1/ten1/buk1/LUN01 4G -b 4096 -s 32k
```

System response:

```
iSCSI LUN clu1/ten1/buk1/LUN01 created.
```

10. Display the status of the iSCSI HA configuration, including which node is the preferred [P] node.

Example:

```
$ neadm service vip status iscHA
```

System response:

```
nedgenode0 [P] Up
```
Managing NBD Devices

As an alternative to using iSCSI, Amazon S3, or Openstack Swift protocols, NexentaEdge can provide access to cluster storage via a Network Block Device (NBD) interface.

Using NEADM, you can create an NBD device, which establishes a client endpoint residing as a block device on the target server. This device can subsequently be mounted to the local node and used as a generic block device. The NBD devices are named /dev/nbd<number>; for example /dev/nbd1.

🔗 To create an NBD device:

1. Log in to the NexentaEdge deployment workstation.
2. Use the following command to create an NBD device:

   ```bash
   $ neadm device nbd create <server_id> <device-number> <object-path> <size> [-b <block-size>] [-o <option>] [-s <chunk-size>] [-r <replication-count>]
   ```

   Example:

   ```bash
   $ neadm device nbd create newnode101 1 clu1/ten1/buk1/NBD1 1G
   ```

   Where:

   - `<server_id>` is the ID or hostname of the server where the NBD device is created.
   - `<device-number>` specifies the device NBD device number; for example, specify 1 for NBD1.
   - `<object-path>` specifies the object path for the NBD device; for example, /clu1/ten1/buk1/NBD1. Note that multiple NBD devices cannot use the same object path.
   - `<size>` is the size of the NBD device. To specify a size in gigabytes, use the GB suffix (for example, 2GB to specify 2 gigabytes). If you do not specify a suffix, a size in bytes is assumed.
   - `-b <block-size>` is the block size for the NBD device. The default is 4096 bytes. You can change this in increments of 512 bytes, with a minimum block size of 512 bytes.
   - `-o <option>` specifies CSV input of metadata.
For the `<size>`, `<block-size>`, and `<chunk-size>` parameters, you can optionally specify a suffix to indicate the units (B, KB, MB, GB, or TB). For example, to specify a size in gigabytes, use the GB suffix. If you do not specify a suffix, a size in bytes is assumed.

System response:

```
NBD device clu1/ten1/buk1/NBD1 created.
```

3. Use the `neadm device nbd list` command to verify that the new NBD device exists for the server.

Example:

```
$ neadm device nbd list newnode1
```

```
NBD HOST        SIZE BLOCK CHUNK REPCOUNT PATH
1   newnode101  1G   4096  32768 3        clu1/ten1/buk1/NBD1
```

\* To delete an NBD device:

1. From the NexentaEdge deployment workstation, use the following command to delete an NBD device:

```
$ neadm device nbd delete <server_id> <device-number> <objectPath>
```

Example:

```
$ neadm device nbd delete 681B1B632A85C9DC15DAB0D31CEAF88D 1 clu1/ten1/buk1/NBD1
```

Where:

- `<server_id>` Is the ID or hostname of the server where the NBD device to be deleted is located.
- `<device-number>` Specifies the device NBD device number; for example, specify 1 for NBD1.
- `<object-path>` Specifies the object path for the NBD device; for example, `/clu1/ten1/buk1/NBD1`.

System response:

```
NBD device clu1/ten1/buk1/NBD1 deleted.
```
This chapter describes how to use the NEADM tool to set up an OpenStack Swift storage service group for a NexentaEdge cluster, which allows the cluster to operate as an OpenStack Swift object storage system. When configured for a tenant in a NexentaEdge cluster, an OpenStack Swift storage service provides the following features:

- To clients, the NexentaEdge cluster appears to be an OpenStack Swift storage system. This means that clients can use OpenStack Swift API function calls to perform GET, PUT, and DELETE operations for objects stored in the NexentaEdge cluster. The specific OpenStack Swift API function calls supported by NexentaEdge are listed in Supported OpenStack Swift API Functions In NexentaEdge.

- The NexentaEdge cluster can be configured to support authentication for client requests. Basic authentication, TempAuth authentication, and AD/LDAP/Keystone authentication are supported options. See Configuring Authentication for the OpenStack Swift Storage Service Group.

- The default ports used for making HTTP or HTTP requests to the OpenStack Swift storage service group can be changed if necessary. See Changing the Ports for OpenStack Swift API Calls.

Before You Begin

Before setting up an OpenStack Swift storage service group, verify the following:

- NexentaEdge has been deployed on the servers in the cluster that will provide access to clients (gateway nodes) and the servers that will provide storage (data nodes).
Setting Up an OpenStack Swift Storage Service Group

To configure an OpenStack Swift storage service group:

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).

2. Use the following command to create an OpenStack Swift storage service group:
   
   ```
   $ neadm service create swift <service-group>
   
   Example:
   
   $ neadm service create swift sw01
   
   System response:
   
   Service sw01 created
   ```

3. List the IDs of the NexentaEdge server nodes:
   
   ```
   $ neadm system status
   
   System response:
   
   ZONE:HOST:CID   SID               UTIL  CAP  CPU           MEM       DEVs  STATE
   0:node32:        B5DEF7B9B3C690... 0%   160G 2/0.2@2.34Ghz 4.16G/7.8G 4/4  ONLINE
   0:node33:    [M] B3EF9C30B6F79F... 0%   160G 2/0.1@2.34Ghz 4.18G/7.8G 4/4  ONLINE
   0:node34:        155B81C291FA4E... 0%   160G 2/0.1@2.34Ghz 4.18G/7.8G 4/4  ONLINE
   0:node35:        9BB624B94C8E2C... 0%   160G 2/0.1@2.34Ghz 4.15G/7.8G 4/4  ONLINE
   0:node36:        F88C6C3180C34E... 0%   160G 2/0.1@2.34Ghz 4.15G/7.8G 4/4  ONLINE
   ```

4. Copy the ID of the server node that you want to use as a gateway node.

5. Use the following command to make the node a gateway node for the storage service group. This allows the node to respond to OpenStack Swift API calls from clients.
   
   ```
   $ neadm service add <service-group> <server-id>
   
   Example:
   
   $ neadm service add sw01 B5DEF7B9B3C69055A9079125BC24FD29
   
   System response:
   
   Service sw01 added to B5DEF7B9B3C69055A9079125BC24FD29
   ```

6. Use the following command to apply the OpenStack Swift storage service group to a specific cluster.
   
   ```
   $ neadm service serve <service-group> <cluster>
   ```
Example:
$ neadm service serve sw01 clu1
System response:
Service sw01 now serving path clu1

7. Use the following command to enable the OpenStack Swift storage service group:
$ neadm service enable <service-group>
Example:
$ neadm service enable sw01
System response:
Service sw01 enabled

8. Use the following command to verify that the OpenStack Swift storage service group is enabled.
$ neadm service list
System response:
<table>
<thead>
<tr>
<th>TYPE</th>
<th>NAME</th>
<th>SERVERID</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>swift</td>
<td>sw01</td>
<td>B5DEF7B9B3C69055A9079125BC24FD29</td>
<td>enabled</td>
</tr>
</tbody>
</table>

At this point, clients can use OpenStack Swift API function calls to the gateway node to store and retrieve objects in the NexentaEdge cluster.

You can optionally configure authentication for the client requests using basic access authentication, TempAuth authentication, or AD/LDAP/Keystone authentication. See Configuring Authentication for the OpenStack Swift Storage Service Group.

9. Use the following command display the properties of the OpenStack Swift storage service group.
$ neadm service show <service-group>
Example:
$ neadm service show sw01
System response:
<table>
<thead>
<tr>
<th>FIELD</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Service-Name</td>
<td>sw01</td>
</tr>
<tr>
<td>X-Service-Type</td>
<td>swift</td>
</tr>
<tr>
<td>X-Service-Worker</td>
<td>ccowgw</td>
</tr>
<tr>
<td>X-Number-Of-Versions</td>
<td>1</td>
</tr>
<tr>
<td>X-Auth-Type</td>
<td>disabled</td>
</tr>
<tr>
<td>X-Description</td>
<td>SWIFT Object</td>
</tr>
<tr>
<td>X-Need-MD5</td>
<td>false</td>
</tr>
<tr>
<td>X-ACL-On</td>
<td>false</td>
</tr>
<tr>
<td>X-List-Cache</td>
<td>true</td>
</tr>
<tr>
<td>X-HTTP-Port</td>
<td>9981</td>
</tr>
<tr>
<td>X-HTTPS-Port</td>
<td>443</td>
</tr>
<tr>
<td>X-Auth-TTL</td>
<td>600</td>
</tr>
<tr>
<td>X-Swift-Versioning</td>
<td>disabled</td>
</tr>
<tr>
<td>X-Trust-Proxy</td>
<td>true</td>
</tr>
<tr>
<td>X-Access-Log</td>
<td>false</td>
</tr>
<tr>
<td>X-Servers</td>
<td>B5DEF7B9B3C69055A9079125BC24FD29</td>
</tr>
<tr>
<td>X-ContainerId</td>
<td>B5DEF7B9B3C69055A9079125BC24FD29 80f3c376eccc</td>
</tr>
<tr>
<td>X-ContainerIPv6</td>
<td>B5DEF7B9B3C69055A9079125BC24FD29 fd00:a228:7eb3:9f843:652f:54f4:12be5d25e5</td>
</tr>
<tr>
<td>X-Status</td>
<td>enabled</td>
</tr>
<tr>
<td>X-Container-Hostname</td>
<td>B5DEF7B9B3C69055A9079125BC24FD29 dbe5e27034385164</td>
</tr>
</tbody>
</table>
Configuring Authentication for the OpenStack Swift Storage Service Group

An OpenStack Swift storage service group supports the following mechanisms for authenticating users accessing objects stored in the NexentaEdge cluster:

- Basic access authentication – Client username and password credentials are supplied in an HTTP header included in HTTP requests.
- TempAuth – A user supplies a token that Swift sends to an authentication system. If the authentication system validates the token, it returns an expiration time for the token. Swift allows access to the user until the token expires (for example, after 15 minutes).
- AD/LDAP/Keystone authentication – Access is granted to the user based on interaction with an external host.

By default, authentication is disabled; that is, no authentication is performed for users accessing objects in the cluster.

To apply configuration settings set with the `neadm service configure` command, the storage service must be stopped and restarted. Until this is done, the previous settings remain in effect. For a currently running service, you can change the configuration settings, then stop and restart the service; for a currently disabled service, you can change the configuration settings, then enable the service.

Configuring Basic Access Authentication

To configure basic access authentication for an OpenStack Swift storage service group:

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).
2. Use the following command to set the authentication type for the storage service group to basic.
   ```bash
   $ neadm service configure <service-group> X-Auth-Type basic
   ```
   Example:
   ```bash
   $ neadm service configure sw01 X-Auth-Type basic
   System response:
   Service sw01 updated
   ```
3. If the storage service group is currently enabled, use the following command to restart (disable and re-enable) it. Note that any in-progress upload/download operations are aborted during the restart; consequently, you may want to disable user access during service configuration.
   ```bash
   $ neadm service restart <service-group>
   ```
Example:

```
$ neadm service restart sw01
System response:
Service sw01 restarted
```

4. If the storage service group is currently disabled, use the following command to enable it.

```
$ neadm service enable <service-group>
Example:
$ neadm service enable sw01
System response:
Service sw01 enabled
```

Configuring TempAuth Authentication

To configure TempAuth authentication, you enable it on the NexentaEdge deployment workstation, then specify credentials for users. The users can create tokens by sending requests to /auth/v1.0 on any gateway that is running the service group. The user-created tokens are authenticated and cached in Swift until they expire after 15 minutes.

- **To configure TempAuth authentication:**
  1. Log in to the NexentaEdge deployment workstation.
  2. Use the following command to set the authentication type for the storage service group to TempAuth.
     
     ```
     $ neadm service configure <service-group> X-Auth-Type <auth-type>
     Example:
     $ neadm service configure sw01 X-Auth-Type tempauth
     System response:
     Service sw01 updated
     ```
  3. If the storage service group is currently enabled, use the following command to restart (disable and re-enable) it. Note that any in-progress upload/download operations are aborted during the restart; consequently, you may want to disable user access during service configuration.
     
     ```
     $ neadm service restart <service-group>
     Example:
     $ neadm service restart sw01
     System response:
     Service sw01 restarted
     ```
  4. If the storage service group is currently disabled, use the following command to enable it.

     ```
     $ neadm service enable <service-group>
     ```
Example:

$ neadm service enable sw01
System response:
Service sw01 enabled

5. Use the following command to create a user:

$ neadm tenant user create <cluster-name>/<tenant-name> <user>
<password>
Example:

$ neadm tenant user create clu1/ten1 usr1 password1234
System response:
Swift user usr1 created successfully

You can create up to 1,024 users for a tenant.

Configuring AD, LDAP, and Keystone Authentication

Use the procedure in this section to configure the OpenStack Swift storage service group to use Active Directory (AD), LDAP, or Keystone authentication when accessing objects in the NexentaEdge cluster.

- To configure AD/LDAP/Keystone authentication:

1. Log in to the NexentaEdge deployment workstation.
2. Install the `python-swiftclient` command-line tool. See this link for instructions.

   $ pip install python-swiftclient

3. Configure an AD, LDAP, or Keystone identity service for the tenant. See Configuring Identity Services for Tenants.

4. Use the following commands to use AD authentication for the storage service group.

   $ neadm service configure <service-group> X-Auth-Type ad
   $ neadm service restart <service-group>

   Example:

   $ neadm service configure sw01 X-Auth-Type ad
   $ neadm service restart sw01
   $ swift -A http://my.gateway.com:9981/auth/v1.0 -U ten1:user -K password123 list

5. Use the following commands to use LDAP authentication for the storage service group.

   $ neadm service configure <service-group> X-Auth-Type ldap
   $ neadm service restart <service-group>
$ swift -A http://<gateway_address>:<gateway_port>/auth/v1.0 \
        -U <tenant>:<ldap_user> -K <ldap_user_password> <command>

Example:

$ neadm service configure sw01 X-Auth-Type ldap
$ neadm service restart sw01
$ swift -A http://my.gateway.com:9981/auth/v1.0 -U ten1:user \
        -K password123 list

6. Use the following commands to use Keystone authentication for the storage service group.

$ neadm service configure <service-group> X-Auth-Type keystone
$ neadm service restart <service-group>

$ swift --os-auth-url http://<keystone_address>:<keystone_port>/v3 --auth-version 3\ 
        --os-project-name <tenant> --os-project-domain-name <project_domain> \ 
        --os-username <keystone_user> --os-user-domain-name <keystone_user_domain> \ 
        --os-password <keystone_user_password> \ 
        --os-storage-url http://<gateway_address>:<gateway_port>/v1/<tenant> <command>

Example:

$ neadm service configure sw01 X-Auth-Type keystone
$ neadm service restart sw01
$ swift --os-auth-url http://my.keystone.com:5000/v3 --auth-version 3\ 
        --os-project-name ten1 --os-project-domain-name Default \ 
        --os-username user1 --os-user-domain-name Default \ 
        --os-password password1 \ 
        --os-storage-url http://my.gateway.com:9981/v1/test list

Changing the Ports for OpenStack Swift API Calls

The default ports for OpenStack Swift API calls are 9981 for HTTP and 443 for HTTPS. If necessary, you can change these port numbers. You can also disable HTTP or HTTPS access to the storage service group.

❖ To change the ports used for OpenStack Swift API Calls:

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).

2. Use the following command to change the port used for API function calls:

$ neadm service configure <service-group> <protocol> <new-port-number>

Example:

$ neadm service configure sw01 X-HTTP-Port 8181

Where:

<service-group> Is the OpenStack Swift storage service group for which the API port is being changed.

<protocol> Can be either X-HTTP-Port or X-HTTPS-Port.
3. If the storage service group is currently enabled, use the following command to restart (disable and re-enable) it. Note that any in-progress upload/download operations are aborted during the restart; consequently, you may want to disable user access during service configuration.

```
$ neadm service restart <service-group>
```

Example:

```
$ neadm service restart sw01
System response:
Service sw01 restarted
```

4. If the storage service group is currently disabled, use the following command to enable it.

```
$ neadm service enable <service-group>
```

Example:

```
$ neadm service enable sw01
System response:
Service sw01 enabled
```

Configuring the OpenStack Swift Gateway to Provide MD5 Checksum Values for Objects

The ETag/Checksum field for an object consists of a checksum value that corresponds to a unique identifier for the object at the current point in time. This checksum value is calculable on both the client and the server.

The following approaches can be used to provide the required checksum value:

- Provide an MD5 checksum value of the entire object.
- Use a vendor-defined algorithm (specified with `x-amz-server-side-encryption-customer-algorithm`) to provide the checksum value of the object. By default NexentaEdge uses this approach; the checksum value is a portion of the unique SHA512 hash ID that corresponds to the specific object.

Some applications assume that MD5 is always used, regardless of user request. For these applications, you can configure the OpenStack Swift service gateway node to provide the MD5 checksum value for each object request, rather than the vendor-defined checksum value.
To configure the OpenStack Swift gateway node to provide MD5 checksum values for objects:

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).

2. Use the following command to provide the MD5 checksum value for each object request:
   
   ```bash
   $ neadm service configure <service-group> X-Need-MD5 true
   ```
   
   Example:
   ```bash
   $ neadm service configure sw01 X-Need-MD5 true
   ```
   
   System response:
   Service sw01 updated successfully.

3. If necessary, use the following command to disable providing the MD5 checksum value for each object request:
   
   ```bash
   $ neadm service configure <service-group> X-Need-MD5 false
   ```
   
   Example:
   ```bash
   $ neadm service configure sw01 X-Need-MD5 false
   ```
   
   System response:
   Service sw01 updated successfully.

   When this setting is disabled, the vendor-defined algorithm specified with `x-amz-server-side-encryption-customer-algorithm` is used. This generally provides the same level of safety as MD5, but not all applications support this header.

4. If the storage service group is currently enabled, use the following command to restart (disable and re-enable) it. Note that any in-progress upload/download operations are aborted during the restart; consequently, you may want to disable user access during service configuration.
   
   ```bash
   $ neadm service restart <service-group>
   ```
   
   Example:
   ```bash
   $ neadm service restart sw01
   ```
   
   System response:
   Service sw01 restarted

5. If the storage service group is currently disabled, use the following command to enable it.
   
   ```bash
   $ neadm service enable <service-group>
   ```
   
   Example:
   ```bash
   $ neadm service enable sw01
   ```
   
   System response:
   Service sw01 enabled
Enabling Versioning Support on the OpenStack Swift Gateway

NexentaEdge supports management of multiple versions of objects stored in the cluster. Versioning allows you to create multiple versions of an object; you can copy or download a specific version of an object, as well as delete or restore a deleted version of an object.

To enable versioning for objects stored in an OpenStack Swift storage service group:

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).

2. Use the following command to enable Swift versioning.
   
   $ neadm service configure <service-group> X-Swift-Versioning enabled
   
   Example:
   
   $ neadm service configure sw01 X-Swift-Versioning enabled
   
   System response:
   
   Service sw01 updated successfully.

3. Use the following command to specify the maximum number of versions of an object that NexentaEdge stores in the cluster. You can specify up to 100 versions.
   
   $ neadm service configure <service-group> X-Number-Of-Versions <number>
   
   Example:
   
   $ neadm service configure sw01 X-Number-Of-Versions 3
   
   System response:
   
   Service sw01 updated successfully.

5. To disable NexentaEdge from keeping multiple versions of objects, use the following command.
   
   $ neadm service configure <service-group> X-Swift-Versioning disabled
   
   Example:
   
   $ neadm service configure sw01 X-Swift-Versioning disabled
   
   System response:
   
   Service sw01 updated successfully.

5. If the storage service group is currently enabled, use the following command to restart (disable and re-enable) it. Note that any in-progress upload/download operations are aborted during the restart; consequently, you may want to disable user access during service configuration.
   
   $ neadm service restart <service-group>
   
   Example:
   
   $ neadm service restart sw01
Service sw01 restarted

6. If the storage service group is currently disabled, use the following command to enable it.

$ neadm service enable <service-group>

Example:

$ neadm service enable sw01

System response:

Service sw01 enabled

OpenStack Swift Large Objects Support

NexentaEdge does not impose any restriction on the size of objects uploaded to the cluster. However, Nexenta recommends uploading objects no larger than 1 gigabyte to limit the possibility that a network failure would cause an object upload to abort. NexentaEdge fully supports Swift large objects, documented here.

You can use a Swift client, available here, to manage large objects in the cluster. To use this tool with NexentaEdge, do the following:

1. Configure an OpenStack Swift storage service group.
2. Install the Swift client.
3. Export the following Swift configuration parameters:

   export ST_AUTH=http://<gateway ip>:<swift port>/auth/v1.0
   export ST_USER=<tenant>:<user>
   export ST_KEY=<password>

4. Once you install the Swift client, you can use it to list the contents of buckets, upload and download data, including large objects.

Setting Resource Limits for the OpenStack Swift Storage Service Group

You can optionally configure limits for memory and CPU usage for containers used by the storage service group. This allows you to limit resource consumption on nodes, as well as set hard limits for nodes that have many containers or services, or in hyper-converged setups.

- To set memory and CPU resource limits for containers used by an OpenStack Swift storage service group:
  1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).
2. Use the following command to limit the amount of memory for a container used by a specified storage service group:

\$ neadm service configure <service-group> X-Service-Mem-Limit <bytes>

Example:

\$ neadm service configure sw01 X-Service-Mem-Limit 64G

System response:
Service sw01 updated successfully.

Where:

<bytes> Sets a limit in bytes provided to the container before the kernel puts back-pressure on allocations, inclusive of all caches in the service and container.

3. Use the following command to limit the CPU resources for a container used by a specified storage service group:

\$ neadm service configure <service-group> X-Service-CPU-Limit <cores>

Example:

\$ neadm service configure sw01 X-Service-CPU-Limit 4

System response:
Service sw01 updated successfully.

Where:

<cores> Is the number of cores dedicated to the container before the kernel puts back-pressure on scheduling time for the processes in the container. This is enforced by cgroups, not NUMA.

4. If the storage service group is currently enabled, use the following command to restart (disable and re-enable) it. Note that any in-progress upload/download operations are aborted during the restart; consequently, you may want to disable user access during service configuration.

\$ neadm service restart <service-group>

Example:

\$ neadm service restart sw01

System response:
Service sw01 restarted

5. If the storage service group is currently disabled, use the following command to enable it.

\$ neadm service enable <service-group>

Example:

\$ neadm service enable sw01

System response:
Service sw01 enabled
Configuring an Amazon S3 Storage Service Group

This chapter includes the following topics:

- Overview
- Before You Begin
- Setting Up an Amazon S3 Storage Service Group
- Configuring Authentication for the Amazon S3 Storage Service Group
- Changing the Ports for Amazon S3 API Calls
- Changing the Region Setting Used for Amazon S3 API Calls
- Configuring the Amazon S3 Gateway to Provide Vendor-Defined Checksum Values for Objects
- Enabling Versioning Support on the Amazon S3 Gateway
- Amazon S3 Multipart Upload Support
- Setting Resource Limits for the Amazon S3 Storage Service Group

Overview

This chapter describes how to set up an Amazon S3 storage service group for a NexentaEdge cluster, which allows the cluster to operate as an Amazon S3 object storage system. When configured for a tenant in a NexentaEdge cluster, an Amazon S3 storage service group provides the following features:

- To clients, the NexentaEdge cluster appears to be an Amazon S3 storage system. This means that clients can use Amazon S3 API function calls to perform GET, PUT, and DELETE operations for objects stored in the NexentaEdge cluster. The specific Amazon S3 API function calls supported by NexentaEdge are listed in Supported Amazon S3 API Functions In NexentaEdge.
- The NexentaEdge cluster supports authentication for users accessing objects stored in the cluster. See Configuring Authentication for the Amazon S3 Storage Service Group.
- The default ports used for making HTTP or HTTPS requests to the Amazon S3 storage service group can be changed if necessary. See Changing the Ports for Amazon S3 API Calls.

Before You Begin

Before setting up an Amazon S3 storage service group, verify the following:

- NexentaEdge has been deployed on the servers in the cluster that will provide access to clients (gateway nodes) and the servers that will provide storage (data nodes).
• The servers to be used as gateway nodes have network connectivity to the clients and to the data nodes in the NexentaEdge cluster.

• A NexentaEdge logical cluster has been created, and the logical cluster has at least one tenant. Enter the `neadm tenant list` command on the deployment workstation to verify.

Setting Up an Amazon S3 Storage Service Group

❖ To configure an Amazon S3 storage service group:

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).

2. Use the following command to create an Amazon S3 storage service group for a tenant in a cluster:

   $ neadm service create s3 <service-group>

   Example:

   $ neadm service create s3 s301

   System response:

   Service s301 created

3. List the IDs of the NexentaEdge server nodes:

   $ neadm system status

   System response:

   ZONE:HOST:CID    SID               UTIL CAP  CPU           MEM        DEVs STATE
   0:node32:        B5DEF7B9B3C690... 0%   160G 2/0.2@2.34Ghz 4.16G/7.8G 4/4  ONLINE
   0:node33:    [M] B3EF9C30B6F79F... 0%   160G 2/0.1@2.34Ghz 4.18G/7.8G 4/4  ONLINE
   0:node34:        155B81C291FA4E... 0%   160G 2/0.1@2.34Ghz 4.18G/7.8G 4/4  ONLINE
   0:node35:        9BB624B94C8E2C... 0%   160G 2/0.1@2.34Ghz 4.15G/7.8G 4/4  ONLINE
   0:node36:        F88C6C3180C34E... 0%   160G 2/0.1@2.34Ghz 4.15G/7.8G 4/4  ONLINE

4. Copy the ID of the server node that you want to use as a gateway node.

5. Use the following command to make the server node a gateway node for the storage service group. This allows the node to respond to Amazon S3 API calls from clients.

   $ neadm service add <service-group> <server-id>

   Example:

   $ neadm service add s301 B3EF9C30B6F79F8831BE6FF908333D99

   System response:

   Service s301 added to B3EF9C30B6F79F8831BE6FF908333D99

6. For Docker deployments, you must configure the Docker network that links the containers to the node on which they are contained.

   The following commands show an example of how to configure a Docker network named `client-net` using the Macvlan network driver. You can use other network drivers if necessary.
7. For Docker deployments, list the available Docker networks and determine which one you want to attach to the NFS storage service group. In the example below, the network `client-net` is used.

Example:

```bash
$ docker network ls
```

System response:

```
NETWORK ID            NAME
 DRIVER
 7fc4feb8c647         client-net
 bridge
 9f904ee27bf5          none
 null
 cf03ee07fb4          host
 host
 78b3e04fc4          multi-host
 overlay
```

8. For Docker deployments, determine subnet information for the selected Docker network. In the example below, the network `client-net` is used.

Example:

```bash
$ docker network inspect client-net
```

System response:

```
[
 {
  "Name": "client-net",
  "Id": "b2b1a2cba717161d984383fd68218cf70bbbd17d32849688f7c98b0f",
  "Scope": "local",
  "Driver": "bridge",
  "IPAM": {
    "Driver": "default",
    "Config": [
     {
      "Subnet": "192.168.200.1/16",
      "Gateway": "192.168.200.21"
     }
    ]
   }
  }
],
```

9. For Docker deployments, display the IDs of the gateway node configured for the storage service group.

Example:

```bash
$ neadm service show s301
```
10. Using the ID from the previous step, add the Docker network to the gateway node.

   Example:
   
   $ neadm service configure s301 X-Container-Network- 
   B3EF9C30B6F79F8831BE6FF908333D99 "client-net --ip 192.168.200.1"

   To specify multiple Docker networks, enclose them in quotes and separate them with
   semicolons; for example:

   $ neadm service configure s301 X-Container-Network- 
   B3EF9C30B6F79F8831BE6FF908333D99 "client-net --ip 192.168.200.1;local-net --ip 192.168.1.23"

11. Use the following command to apply the Amazon S3 storage service group to a specific cluster
    and tenant combination. The cluster/tenant combination is referred to as a logical path.

   Example:

   $ neadm service serve <service-group> <logical-path>

   Example:

   $ neadm service serve s301 clu1/ten1

   System response:

   Service s301 now serving path clu1/ten1

12. Use the following command to enable the Amazon S3 storage service group:

   Example:

   $ neadm service enable <service-group>

   Example:

   $ neadm service enable s301

   System response:

   Service s301 enabled

13. Use the following command to verify that the Amazon S3 storage service group is enabled:

   Example:

   $ neadm service list

   System response:

   TYPE NAME SERVERID STATUS
   s3  s301 B3EF9C30B6F79F8831BE6FF908333D99 enabled
At this point, clients can use Amazon S3 API function calls to the gateway node to store and retrieve objects in the NexentaEdge cluster. You can optionally configure authentication for the client requests. See Configuring Authentication for the Amazon S3 Storage Service Group.

14. Use the following command display the properties of the Amazon S3 storage service group.

   $ neadm service show <service-group>

Example:

   $ neadm service show s301

System response:

   X-Service-Name                                        s301
   X-Service-Type                                        s3
   X-Service-Worker                                      ccowgws3
   X-Number-Of-Versions                                  1
   X-Auth-Type                                           disabled
   X-Description                                         S3 Object
   X-Need-MD5                                            true
   X-ACL-On                                               false
   X-List-Cache                                          true
   X-HTTP-Port                                           9982
   X-HTTPS-Port                                          9443
   X-Region                                               clu1
   X-Trust-Proxy                                         true
   X-Access-Log                                          false
   X-Servers                                              B3EF9C30B6F79F8831BE6FF908333D99
   X-ContainerId-B3EF9C30B6F79F8831BE6FF908333D99        062d04b807c5
   X-ContainerIPv6-B3EF9C30B6F79F8831BE6FF908333D99      fd00::961f:6bc7:30d0
   X-Status                                              enabled
   X-Container-Hostname-B3EF9C30B6F79F8831BE6FF908333D99  936d138e8b6513f5

   Service Group Objects:
   [ "clu1/ten1"
   ]

Configuring Authentication for the Amazon S3 Storage Service Group

The Amazon S3 storage service group supports access key / secret key (key_secret) authentication for users accessing objects stored in the NexentaEdge cluster. The user can enter the access key / secret key parameters in S3 client configuration.

By default, authentication is disabled; that is, no authentication is performed for users accessing objects in the cluster.

❖ To configure key_secret authentication for an Amazon S3 storage service group:

   1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).

   2. Create an S3 user and get an access key and secret key.

      $ neadm tenant user create <cluster-name>/<tenant-name> <user> <password> s3 [administrator]
Example:

$ neadm tenant user create clu1/ten1 user1 pass1 s3
System response:
S3 user user1 created successfully
Access key: 45BYD1CYRTNJAAN2GSAQ
Secret key: g17MkcKh5yN3iiDnFwrsTC6U8ahM13EAfehRH7As

3. Set the authentication type for the S3 storage service group to key_secret.

$ neadm service configure <service-group> X-Auth-Type key_secret
Example:

$ neadm service configure s301 X-Auth-Type key_secret
System response:
Service s301 updated

4. If the storage service group is currently enabled, use the following command to restart (disable and re-enable) it. Note that any in-progress upload/download operations are aborted during the restart; consequently, you may want to disable user access during service configuration.

$ neadm service restart <service-group>
Example:

$ neadm service restart s301
System response:
Service s301 restarted

5. If the storage service group is currently disabled, use the following command to enable it.

$ neadm service enable <service-group>
Example:

$ neadm service enable s301
System response:
Service s301 enabled

Installing the S3cmd Tool

You can use the freeware S3cmd tool, available here, to manage data in the cluster.

- **To use the S3cmd tool with NexentaEdge:**

1. Configure an Amazon S3 storage service group. See Setting Up an Amazon S3 Storage Service Group.
2. Configure the Amazon S3 gateway to provide checksum values for objects. See Configuring the Amazon S3 Gateway to Provide Vendor-Defined Checksum Values for Objects.
3. Install Python 2.6 or greater.
4. Install the S3cmd tool, supplying access key and secret key for the S3 storage service group.
5. Edit the `.s3cfg` configuration file and change the following lines:

```plaintext
host_base = <gateway ip>:<s3 Port>
host_bucket = <gateway ip>:<s3 Port>/%(bucket)
bucket_location = <region>
```

Example:

```plaintext
host_base = 192.168.1.10:9982
host_bucket = 192.168.1.10:9982/%(bucket)
bucket_location = us-east-1
```

6. Once you install the S3cmd tool, you can use the `s3cmd ls`, `s3cmd put`, and `s3cmd get` commands to manage data in the cluster.

### Setting the Authentication Version for the Amazon S3 Storage Service Group

If you have the S3cmd tool installed, you can set the authentication version used for the Amazon S3 storage service group to either version 2 or version 4. Version 4 authentication takes the region setting into account; only standard Amazon regions work.

- **To set the authentication version for the Amazon S3 storage service group:**
  1. Configure key_secret authentication for the Amazon S3 storage service group. See [Configuring Authentication for the Amazon S3 Storage Service Group](#).
  2. To set version 2 authentication, edit the `.s3cfg` configuration file and make sure the following line exists in the file:
     ```plaintext
     signature_v2 = True
     ```
  3. To set version 4 authentication (the default), edit the `.s3cfg` configuration file and set `signature_v2` to `False` and specify the region.
     ```plaintext
     signature_v2 = False
     bucket_location = us-west-1
     ```
  4. You can also set the region using NEADM using the following commands:
     ```bash
     $ neadm service configure <service-group> X-Region <region-name>
     $ neadm service restart <service-group>
     ```
     Example:
     ```bash
     $ neadm service configure s301 X-Region us-west-1
     $ neadm service restart s301
     ```

### Enabling Access Control Lists for the Amazon S3 Storage Service Group

NexentaEdge supports Access Control Lists (ACLs) for Amazon S3 storage service groups. ACLs are supported at the bucket level, as shown in the examples below.
To enable an ACL for the Amazon S3 storage service group:

1. Create and register administrator and other S3 users. The administrator user can create new buckets, set up quotas (if required), and grant access rights to other users.
   
   Example:
   
   $ neadm tenant user create clu1/ten1 user1 nexenta s3 admin
   $ neadm tenant user create clu1/ten1 user2 nexenta s3

2. Configure key_secret authentication for the Amazon S3 storage service group. See Configuring Authentication for the Amazon S3 Storage Service Group.

3. Enable ACLs for the Amazon S3 storage service group using the following commands:

   $ neadm service configure <service-group> X-ACL-On true
   $ neadm service restart <service-group>

   Example:
   
   $ neadm service configure s301 X-ACL-On true
   $ neadm service restart s301

4. Create a new bucket. You can do this either with NEADM or the S3cmd tool.

   Using NEADM, you can optionally set quotas for the size and number of objects that can exist in the bucket. To do this, use the following command:

   $ neadm bucket create <cluster-name>/<tenant-name>/<bucket-name> -o <options>

   where <options> is a comma-separated list of quota options, described below.

   quota
   Specifies the amount of space allocated to the bucket. To specify the space in gigabytes, use the GB suffix (for example, 2GB to specify 2 gigabytes). If you do not specify a suffix, a size in bytes is assumed.

   quota-count
   Sets the maximum number of objects that can be placed in a bucket.

   Example:
   
   $ neadm bucket create clu1/ten1/buk1 -o quota=1G,quota-count=200

   To create a bucket using the S3cmd tool, use the following command:

   $ s3cmd --access_key=<admin_key> --secret_key=<admin_secret> mb s3://bk1

   For <admin_key> and <admin_secret>, specify the access key and secret key of the administrator user.

5. Using a tool that supports Amazon PUT bucket-level ACLs (see this link), such as the S3cmd tool, configure bucket-level ACLs for the Amazon S3 storage service group.

   For example, the following ACL makes the S3 public access bucket available only to authenticated users:
Changing the Ports for Amazon S3 API Calls

The default ports for Amazon S3 API calls are 9982 for HTTP and 8443 for HTTPS. If necessary, you can change these port numbers using the NEADM tool. You can also disable HTTP or HTTPS access to the storage service group.

To change the ports used for Amazon S3 API calls:

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).
2. Use the following command to change the port used for API function calls:

   ```
   $ neadm service configure <service-group> <protocol> <new-port-number>
   ```

   Example:
   ```
   $ neadm service configure s301 X-HTTP-Port 8181
   ```

   Where:

   - `<service-group>`: Is the Amazon S3 storage service group for which the API port is being changed.
   - `<protocol>`: Can be either `X-HTTP-Port` or `X-HTTPS-Port`.
   - `<new-port-number>`: Is the new port number for Amazon S3 API function calls made to the NexentaEdge cluster for the specified protocol.
     If you specify 0, then access using the specified protocol is disabled.

   System response:
   ```
   Service s301 updated successfully.
   ```

3. If the storage service group is currently enabled, use the following command to restart (disable and re-enable) it. Note that any in-progress upload/download operations are aborted during the restart; consequently, you may want to disable user access during service configuration.

   ```
   $ neadm service restart <service-group>
   ```

   Example:
   ```
   $ neadm service restart s301
   ```
4. If the storage service group is currently disabled, use the following command to enable it.

$$\text{
neadm service enable <service-group>
}$$

Example:

$$\text{
neadm service enable s301
}$$

System response:

Service s301 enabled

Changing the Region Setting Used for Amazon S3 API Calls

The region setting for the Amazon S3 API is used in Amazon V4 authentication. By default, NexentaEdge sets the region name to the cluster name; make sure your client is using the same region value. If necessary, you can change the region setting to a different value.

- To change the region setting for Amazon S3 API calls:

  1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).
  2. Use the following command to change the region used for API function calls:

$$\text{
neadm service configure <service-group> X-Region <region-name>
}$$

Example:

$$\text{
neadm service configure s301 X-Region us-east-1
}$$

Where:

- `<service-group>` Is the Amazon S3 storage service group for which the API port is being changed.
- `<region-name>` Is the region name to be used; default is the cluster name.

System response:

Service s301 updated successfully.

3. If the storage service group is currently enabled, use the following command to restart (disable and re-enable) it. Note that any in-progress upload/download operations are aborted during the restart; consequently, you may want to disable user access during service configuration.

$$\text{
neadm service restart <service-group>
}$$

Example:

$$\text{
neadm service restart s301
}$$

System response:

Service s301 restarted
4. If the storage service group is currently disabled, use the following command to enable it.

```
$ neadm service enable <service-group>
```

Example:

```
$ neadm service enable s301
```

System response:

Service s301 enabled

Configuring the Amazon S3 Gateway to Provide Vendor-Defined Checksum Values for Objects

Amazon S3 requires that the ETag/Checksum field for an object consist of a checksum value that corresponds to a unique identifier for the object at the current point in time. This checksum value is calculable on both the client and the server.

The following approaches can be used to provide the required checksum value:

- Provide an MD5 checksum value of the entire object. By default, NexentaEdge uses this approach.
- Use a vendor-defined algorithm (specified with `x-amz-server-side-encryption-customer-algorithm`) to provide the checksum value of the object. When this approach is used, the checksum value is a portion of the unique SHA512 hash ID that corresponds to the specific object.

Some applications assume that MD5 is always used, regardless of user request. By default, the S3 gateway node provides the MD5 checksum value for each object request, rather than the vendor-defined checksum value. You can optionally configure the S3 gateway node to provide the vendor-defined checksum value if necessary.

✔️ To configure the S3 gateway node to provide vendor-defined checksum values for objects:

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).

2. Use the following command to disable the default setting of providing the MD5 checksum value for each object request:

```
$ neadm service configure <service-group> X-Need-MD5 false
```

Example:

```
$ neadm service configure s301 X-Need-MD5 false
```

System response:

Service s301 updated successfully.

When this setting is disabled, the vendor-defined algorithm specified with `x-amz-server-side-encryption-customer-algorithm` is used. This generally provides the same level of safety as MD5, but not all applications support this header.

3. If you disable the default setting (that is, providing the MD5 checksum value for each object request) you can use the following command to revert back to it if necessary:
Enabling Versioning Support on the Amazon S3 Gateway

NexentaEdge supports management of multiple versions of objects stored in the cluster. Versioning allows you to create multiple versions of an object; you can copy or download a specific version of an object, as well as delete or restore a deleted version of an object.

- To enable versioning for objects stored in an Amazon S3 storage service group:
  1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).
  2. Use the following command to specify the maximum number of versions of an object that NexentaEdge stores in the cluster. You can specify up to 100 versions.
     
     ```bash
     $ neadm service configure <service-group> X-Number-Of-Versions <number>
     ```
     
     Example:
     ```bash
     $ neadm service configure s301 X-Number-Of-Versions 3
     ```
     System response:
     ```
     Service s301 updated successfully.
     ```
3. To disable NexentaEdge from keeping multiple versions of objects, set the X-Number-Of-Versions parameter to 1.
   Example:
   
   $ neadm service configure s301 X-Number-Of-Versions 1
   
   System response:
   
   Service s301 updated successfully.

4. If the storage service group is currently enabled, use the following command to restart (disable and re-enable) it. Note that any in-progress upload/download operations are aborted during the restart; consequently, you may want to disable user access during service configuration.
   $ neadm service restart <service-group>
   
   Example:
   
   $ neadm service restart s301
   
   System response:
   
   Service s301 restarted

5. If the storage service group is currently disabled, use the following command to enable it.
   $ neadm service enable <service-group>
   
   Example:
   
   $ neadm service enable s301
   
   System response:
   
   Service s301 enabled

6. Enable or suspend versioning for individual buckets using the standard Amazon S3 API. See this link for information.

Amazon S3 Multipart Upload Support

NexentaEdge does not impose any restriction on the size of objects uploaded to the cluster. However, Nexenta recommends uploading objects no larger than 1 gigabyte to limit the possibility that a network failure would cause an object upload to abort. NexentaEdge fully supports the Amazon S3 multipart upload API, which is documented here.

To utilize Amazon S3 multipart functionality, you can use the freeware S3 Browser utility, available here. Specify the IP address and port for the S3 gateway node, and the access key and secret key for the S3 storage service group, as shown in the example below:
You can also use the S3cmd tool (see Installing the S3cmd Tool) to configure multipart upload settings.

- **To configure multipart upload settings with the S3cmd tool**:
  
  1. Edit the .s3cfg configuration file and change the following lines:

     host_base = <gateway ip>:<s3 Port>
     host_bucket = <gateway ip>:<s3 Port>/%(bucket)
     bucket_location = <region>
     ...
     multipart_chunk_size_mb = <the chunk size for multipart upload>
     multipart_max_chunks = <maximum number of chunks for multipart upload>

     Example:

     host_base = 192.168.1.10:9982
     host_bucket = 192.168.1.10:9982/%(bucket)
     bucket_location = us-east-1
     ...
     multipart_chunk_size_mb = 1024
     multipart_max_chunks = 1000

### Setting Resource Limits for the Amazon S3 Storage Service Group

You can optionally configure limits for memory and CPU usage for containers used by the storage service group. This allows you to limit resource consumption on nodes, as well as set hard limits for nodes that have many containers or services, or in hyper-converged setups.
To set memory and CPU resource limits for containers used by an Amazon S3 storage service group:

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).

2. Use the following command to limit the amount of memory for a container used by a specified storage service group:

   $ neadm service configure <service-group> X-Service-Mem-Limit <bytes>

   Example:
   
   $ neadm service configure s301 X-Service-Mem-Limit 64G

   System response:
   
   Service s301 updated successfully.

   Where:

   <bytes> Sets a limit in bytes provided to the container before the kernel puts back-pressure on allocations, inclusive of all caches in the service and container.

3. Use the following command to limit the CPU resources for a container used by a specified storage service group:

   $ neadm service configure <service-group> X-Service-CPU-Limit <cores>

   Example:
   
   $ neadm service s301 configure X-Service-CPU-Limit 4

   System response:
   
   Service s301 updated successfully.

   Where:

   <cores> Is the number of cores dedicated to the container before the kernel puts back-pressure on scheduling time for the processes in the container. This is enforced by cgroups, not NUMA.

4. If the storage service group is currently enabled, use the following command to restart (disable and re-enable) it. Note that any in-progress upload/download operations are aborted during the restart; consequently, you may want to disable user access during service configuration.

   $ neadm service restart <service-group>

   Example:
   
   $ neadm service restart s301

   System response:
   
   Service s301 restarted

5. If the storage service group is currently disabled, use the following command to enable it.

   $ neadm service enable <service-group>
Example:

$ neadm service enable s301

System response:

Service s301 enabled
Configuring an NFS Storage Service Group

This chapter includes the following topics:

- Overview
- Before You Begin
- Setting Up an NFS Storage Service Group
- Setting Resource Limits for the NFS Storage Service Group
- Configuring Access Control Lists for the NFS Storage Service Group

Overview

This chapter describes how to use the NEADM tool to set up an NFS storage service group for a NexentaEdge cluster, which allows buckets to be exported as NFS filesystems. When configured for a tenant in a NexentaEdge cluster, an NFS storage service provides the following features:

- To clients, the NexentaEdge cluster appears to be an NFS storage system.
- The bucket name is used as the name of the NFS share; for example, the bucket path `clu1/ten1/buk1` is exported as `hostname:/buk1`.
- Since the object and file namespaces are shared within the NexentaEdge cluster, files stored in an NFS share are also accessible as objects by OpenStack Swift and Amazon S3 clients.

NFS file names and S3 object names combined cannot be longer than 256 characters. It is not possible to have longer names than 256 characters.

Sparse Files in NexentaEdge

Sparse files are files that use metadata to represent empty space in the files, rather than allocating actual blocks to the empty space. Within a sparse file, the blocks with empty space are referred to as “holes”.

When reporting the size of sparse files in a NFS filesystem, NexentaEdge considers the holes to be allocated space, rather than metadata representing the empty space. This means that NexentaEdge reports the size of a sparse file as the blocks that the file actually uses, plus the blocks that make up the holes.

Before You Begin

Before setting up an NFS storage service group, verify the following:
• NexentaEdge has been deployed on the servers in the cluster that will provide access to clients (gateway nodes) and the servers that will provide storage (data nodes).

• The servers to be used as gateway nodes have network connectivity to the clients and to the data nodes in the NexentaEdge cluster.

• A NexentaEdge logical cluster has been created, and the logical cluster has at least one tenant. Enter the neadm tenant list command on the deployment workstation to verify.

Setting Up an NFS Storage Service Group

To configure an NFS storage service group:

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).

2. Use the following command to create an NFS storage service group:

   
   
   
   ```
   $ neadm service create nfs <service-group>
   ```

   Example:

   
   ```
   $ neadm service create nfs nfs01
   ```

   System response:

   
   ```
   Service nfs01 created
   ```

3. List the IDs of the NexentaEdge server nodes:

   ```
   $ neadm system status
   ```

   System response:

   
   ```
   ZONE:HOST:CID SID UTIL CAP CPU MEM DEVs STATE
   0:node32: B5DEF7B9B3C690... 0% 160G 2/0.2@2.34Ghz 4.16G/7.8G 4/4 ONLINE
   0:node33: [M] B3EF9C30B6F79F... 0% 160G 2/0.1@2.34Ghz 4.18G/7.8G 4/4 ONLINE
   0:node34: 155B81C291FA4E... 0% 160G 2/0.1@2.34Ghz 4.18G/7.8G 4/4 ONLINE
   0:node35: 9BB624B94C8E2C... 0% 160G 2/0.1@2.34Ghz 4.15G/7.8G 4/4 ONLINE
   0:node36: F88C6C3180C34E... 0% 160G 2/0.1@2.34Ghz 4.15G/7.8G 4/4 ONLINE
   ```

4. Copy the ID of the server node that you want to use as a gateway node.

5. Use the following command to make the node a gateway node for the storage service group. This allows the node to respond to NFS API calls from clients.

   ```
   $ neadm service add <service-group> <server-id>
   ```

   Example:

   ```
   $ neadm service add nfs01 D586BF84009C4230BFEE31CAC961197D
   ```

   System response:

   ```
   Service nfs01 added to D586BF84009C4230BFEE31CAC961197D
   ```

6. For Docker deployments, you must configure the Docker network that links the containers to the node on which they are contained.
The following commands show an example of how to configure a Docker network named `client-net` using the Macvlan network driver. You can use other network drivers if necessary.

```
$ ifconfig enp2s0f1 192.168.10.201/24 up
$ modprobe macvlan
$ docker network create -d macvlan --subnet 192.168.10.0/24 --gateway 192.168.10.1 -o parent=enp2s0f1 client-net
$ docker run --net=client-net --ip=192.168.10.11 -it alpine /bin/sh
```

7. For Docker deployments, list the available Docker networks and determine which one you want to attach to the NFS storage service group. In the example below, the network `client-net` is used.

Example:
```
$ docker network ls
```

System response:
```
<table>
<thead>
<tr>
<th>NETWORK ID</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>7fca4eb8c647</td>
<td>client-net</td>
</tr>
<tr>
<td>bridge</td>
<td></td>
</tr>
<tr>
<td>9f904ee27bf5</td>
<td>none</td>
</tr>
<tr>
<td>null</td>
<td>host</td>
</tr>
<tr>
<td>cf03ee007fb4</td>
<td>multi-host</td>
</tr>
<tr>
<td>host</td>
<td></td>
</tr>
<tr>
<td>78b03ee04fc4</td>
<td>overlay</td>
</tr>
</tbody>
</table>
```

8. For Docker deployments, determine subnet information for the selected Docker network. In the example below, the network `client-net` is used.

Example:
```
$ docker network inspect client-net
```

System response:
```
[
  {
    "Name": "client-net",
    "Id": "b2b1a2cba717161d984383fd68218cf70bbbd17d328496885f7c98b0f",
    "Scope": "local",
    "Driver": "bridge",
    "IPAM": {
      "Driver": "default",
      "Config": [
        {
          "Subnet": "192.168.200.1/16",
          "Gateway": "192.168.200.21"
        }
      ]
    }
  },
```

9. For Docker deployments, display the IDs of the gateway node configured for the storage service group.
Example:

$ neadm service show nfs01

System response:

<table>
<thead>
<tr>
<th>FIELD</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Service-Name</td>
<td>nfs01</td>
</tr>
<tr>
<td>X-Service-Type</td>
<td>nfs</td>
</tr>
<tr>
<td>X-Service-Worker</td>
<td>nfsserv</td>
</tr>
<tr>
<td>X-Number-Of-Versions</td>
<td>1</td>
</tr>
<tr>
<td>X-Auth-Type</td>
<td>disabled</td>
</tr>
<tr>
<td>X-Description</td>
<td>NFS Server</td>
</tr>
<tr>
<td>X-Servers</td>
<td>000406E4A4C7FF3B182976E1A351CCCA9</td>
</tr>
<tr>
<td>X-Status</td>
<td>enabled</td>
</tr>
</tbody>
</table>

Service Group Objects:

```
[
]
```

10. Using the ID from the previous step, add the Docker network to the gateway node.

Example:

$ neadm service configure nfs01 X-Container-Network-000406E4A4C7FF3B182976E1A351CCCA9 "client-net --ip 192.168.200.1"

To specify multiple Docker networks, enclose them in quotes and separate them with semicolons; for example:

$ neadm service configure nfs01 X-Container-Network-000406E4A4C7FF3B182976E1A351CCCA9 "client-net --ip 192.168.200.1;local-net --ip 192.168.1.23"

11. Use the following command to apply the NFS storage service group to a specific cluster.

```
$ neadm service serve <service-group> <cluster>
```

Example:

$ neadm service serve nfs01 clu1

System response:

Service nfs01 now serving path clu1

12. Use the following command to enable the NFS storage service group:

```
$ neadm service enable <service-group>
```

Example:

$ neadm service enable nfs01

System response:

Service nfs01 enabled

13. Use the following command to verify that the NFS storage service group is enabled.

```
$ neadm service list
```

System response:

```
<table>
<thead>
<tr>
<th>TYPE</th>
<th>NAME</th>
<th>SERVERID</th>
<th>STATUS</th>
</tr>
</thead>
</table>
```
nfs   nfs01  D586BF84009C4230BFEE31CAC961197D enabled

14. Use the following command to display the properties of the NFS storage service group.

   $ neadm service show <service-group>

   Example:
   $ neadm service show nfs01

   System response:

<table>
<thead>
<tr>
<th>FIELD</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Service-Name</td>
<td>nfs01</td>
</tr>
<tr>
<td>X-Service-Type</td>
<td>nfs</td>
</tr>
<tr>
<td>X-Service-Worker</td>
<td>nfsserv</td>
</tr>
<tr>
<td>X-Number-Of-Versions</td>
<td>1</td>
</tr>
<tr>
<td>X-Auth-Type</td>
<td>disabled</td>
</tr>
<tr>
<td>X-Description</td>
<td>NFS Server</td>
</tr>
<tr>
<td>X-Servers</td>
<td>000406E4A4C7FF3B182976E1A351CCA9</td>
</tr>
<tr>
<td>X-Status</td>
<td>enabled</td>
</tr>
</tbody>
</table>

   Service Group Objects:
   [ "clu1/ten1"
   ]

15. Use the following command to export a bucket as an NFS share.

   $ neadm nfs share <service-group> <bucket-path> [ <chunk-size> ]

   Example:
   $ neadm nfs share nfs01 clu1/ten1/buk1

   Where:

   <service-group>         Is the NFS storage service group for which the share is being created.
   <bucket-path>           Is the combination of existing cluster/tenant/bucket for the NFS share; for example clu1/ten1/buk1.
   <chunk-size>            Optionally sets the size of the chunks for the NFS share. NexentaEdge breaks objects stored for this NFS share into chunks of this size. The default is 32 KB. You can specify the chunk size as a power of 2 in 512-byte increments. The minimum chunk size is 4 KB, and the maximum is 1 MB.

   System response:

   Service nfs01 updated successfully.

16. Use the following command to display the status of the servers in the NFS storage service group.

   $ neadm nfs status <service-group>
Example:

$ neadm nfs status nfs01

17. Use the following command to display a list of exported buckets for the NFS storage service group.

$ neadm nfs status <service-group>

Example:

$ neadm nfs list nfs01

System response:

EXPORT   BUCKET
clu1/ten1

ten1

18. Use the following command to remove an existing NFS share.

$ neadm nfs unshare <service-group> <bucket-path>

Example:

$ neadm nfs unshare nfs01 clu1/ten1/buk1

System response:

Service nfs01 updated successfully.

Setting Resource Limits for the NFS Storage Service Group

You can optionally configure limits for memory and CPU usage for containers used by the storage service group. This allows you to limit resource consumption on nodes, as well as set hard limits for nodes that have many containers or services, or in hyper-converged setups.

❖ To set memory and CPU resource limits for containers used by an NFS storage service group:

1. Log in to the NexentaEdge deployment workstation (that is, the workstation where the NEADM tool is installed).

2. Use the following command to limit the amount of memory for a container used by a specified storage service group:

$ neadm service configure <service-group> X-Service-Mem-Limit <bytes>

Example:

$ neadm service configure nfs01 X-Service-Mem-Limit 64G

System response:

Service nfs01 updated successfully.

Where:

<bytes> Sets a limit in bytes provided to the container before the kernel puts back-pressure on allocations, inclusive of all caches in the service and container.
3. Use the following command to limit the CPU resources for a container used by a specified storage service group:

   $ neadm service configure <service-group> X-Service-CPU-Limit <cores>

   Example:
   $ neadm service configure nfs01 X-Service-CPU-Limit 4
   System response:
   Service nfs01 updated successfully.

   Where:
   <cores> Is the number of cores dedicated to the container before the kernel puts back-pressure on scheduling time for the processes in the container. This is enforced by cgroups, not NUMA.

4. If the storage service group is currently enabled, use the following command to restart (disable and re-enable) it. Note that any in-progress upload/download operations are aborted during the restart; consequently, you may want to disable user access during service configuration.

   $ neadm service restart <service-group>

   Example:
   $ neadm service restart nfs01
   System response:
   Service nfs01 restarted

5. If the storage service group is currently disabled, use the following command to enable it.

   $ neadm service enable <service-group>

   Example:
   $ neadm service enable nfs01
   System response:
   Service nfs01 enabled

Configuring Access Control Lists for the NFS Storage Service Group

NexentaEdge supports Access Control Lists (ACLs) for NFS storage service groups. You can specify which clients have read/write, read-only, or no access to a specified NFS share. Wildcards can be specified with an asterisk (*).

To configure an ACL for an NFS storage service group:

1. Use the following command to specify the NFS storage service group, NFS share, and IP addresses to which the ACL applies:
$ neadm service configure <service-group> X-NFS-ACL-<tenant>/<bucket> 'RW=<client-net-IPs> RO=<client-net-IPs> NONE=<client-net-IPs> ' 

Where:

X-NFS-ACL-<tenant>/<bucket> Specifies the name of the NFS share.
RW=<client-net-IPs> Specifies a list of IP addresses to have read/write access to the NFS share.
RO=<client-net-IPs> Specifies a list of IP addresses to have read-only access to the NFS share.
NONE=<client-net-IPs> Specifies a list of IP addresses to have no access to the NFS share.

Example:

$ neadm service configure nfs01 X-NFS-ACL-ten1/buk1 'RW=192.168.0.1 RO=192.168.0.12* NONE=**'

System response:
Service nfs01 updated successfully.

This example grants read/write access to the NFS share ten1/buk1 to 192.168.0.1, read-only access to 192.168.0.12x addresses, and no access to all other addresses.

2. Use the following command to apply the ACL to the NFS share:

$ neadm service update-share <service-group> <bucket-path>

Example:

$ neadm service update-share nfs01 clu1/ten1/buk1

System response:
Service nfs01 updated successfully.
NexentaEdge CLI Commands

This chapter includes the following topics:

- Overview
- NEDEPLOY Commands
- NEADM Commands

Overview

NexentaEdge provides two command-line configuration tools on the deployment workstation: NEDEPLOY and NEADM. NEDEPLOY is used for the initial deployment of the software to the cluster nodes, and NEADM is used for subsequent configuration and monitoring.

This chapter provides a high-level description of the NEDEPLOY and NEADM commands. To display descriptions of the subcommands and parameters for NEDEPLOY and NEADM, use the command-line help. For example, to display the subcommands and parameters for the neadm service command, enter `neadm service help` or `neadm service ?` at the command line.
### NEDEPLOY Commands

Table 9-1: NEDEPLOY Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nedeploy deploy</td>
<td>Adds new server nodes to the cluster.</td>
</tr>
<tr>
<td>nedeploy help</td>
<td>Displays help text for NEDEPLOY; enter nedeploy &lt;command&gt; help to display parameters for individual commands.</td>
</tr>
<tr>
<td>nedeploy precheck</td>
<td>Connects to a specified node and checks whether the operating system, network interface speed, available disks, and installed RAM meet the requirements for NexentaEdge.</td>
</tr>
<tr>
<td>nedeploy version</td>
<td>Displays the version number of the installed NEDEPLOY tool.</td>
</tr>
</tbody>
</table>

### NEADM Commands

Table 9-2: NEADM Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>neadm bucket &lt;subcommand&gt;</td>
<td>Creates, deletes, and lists buckets for a specified tenant within a cluster.</td>
</tr>
<tr>
<td>neadm cluster &lt;subcommand&gt;</td>
<td>Creates, deletes, and lists NexentaEdge logical clusters.</td>
</tr>
<tr>
<td>neadm device &lt;subcommand&gt;</td>
<td>Adds, removes, or lists devices on a specified node in the cluster.</td>
</tr>
<tr>
<td>neadm help</td>
<td>Displays help text for NEADM; enter neadm &lt;command&gt; help to display parameters for individual commands.</td>
</tr>
<tr>
<td>neadm iscsi &lt;subcommand&gt;</td>
<td>Creates, deletes, and configures iSCSI LUNs for an iSCSI storage service.</td>
</tr>
<tr>
<td>neadm iscsi snapshot &lt;subcommand&gt;</td>
<td>Creates, configures, and lists snapshots of LUNs for an iSCSI storage service.</td>
</tr>
<tr>
<td>neadm iscsi status</td>
<td>Displays information about the iSCSI target.</td>
</tr>
<tr>
<td>neadm service &lt;subcommand&gt;</td>
<td>Creates, configures, and lists storage services.</td>
</tr>
<tr>
<td>neadm service vip &lt;subcommand&gt;</td>
<td>Configures virtual IP (VIP) addresses to enable high availability for a storage service.</td>
</tr>
<tr>
<td>neadm system autosupport &lt;subcommand&gt;</td>
<td>Configures delivery via HTTP of NexentaEdge logs and statistics to a specified collector server.</td>
</tr>
<tr>
<td>neadm system check &lt;subcommand&gt;</td>
<td>Displays status and statistics for a NexentaEdge cluster.</td>
</tr>
<tr>
<td>neadm system clear</td>
<td>Clears the statistics gathered for the NexentaEdge cluster.</td>
</tr>
</tbody>
</table>
Table 9-2: NEADM Command Summary (Continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>neadm system init</td>
<td>Initializes the NexentaEdge cluster following deployment.</td>
</tr>
<tr>
<td>neadm system license &lt;subcommand&gt;</td>
<td>Sets or displays the NexentaEdge license.</td>
</tr>
<tr>
<td>neadm system mgmt set</td>
<td>Sets or unsets servers as additional REST proxy endpoints to provide high availability for REST management of the NexentaEdge system.</td>
</tr>
<tr>
<td>neadm system passwd</td>
<td>Sets the password for the NexentaEdge management REST API.</td>
</tr>
<tr>
<td>neadm system service-checkpoint &lt;subcommand&gt;</td>
<td>Displays, sets, or refreshes service checkpoints.</td>
</tr>
<tr>
<td>neadm system status</td>
<td>Displays hostname, server ID, and status for each node in the NexentaEdge cluster.</td>
</tr>
<tr>
<td>neadm system summary</td>
<td>Displays usage and status information about the NexentaEdge cluster.</td>
</tr>
<tr>
<td>neadm system version</td>
<td>Displays the NexentaEdge version and internal build number running on the node.</td>
</tr>
<tr>
<td>neadm tenant &lt;subcommand&gt;</td>
<td>Creates, deletes, and lists tenants for a specified NexentaEdge cluster.</td>
</tr>
<tr>
<td>neadm tenant user &lt;subcommand&gt;</td>
<td>Specifies credentials for users in the context of this cluster/tenant. See Configuring TempAuth Authentication.</td>
</tr>
<tr>
<td>neadm version</td>
<td>Displays the version number of the installed NEADM tool.</td>
</tr>
</tbody>
</table>
Supported OpenStack Swift API Functions In NexentaEdge

This chapter includes the following topics:

- Account API Functions
- Container API Functions
- Object API Functions
Account API Functions

- GET Method
- POST Method
- HEAD Method
GET Method

Table 10-1: Supported API Parameters for Account GET Requests and Responses

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Header parameters</td>
<td>X-Auth-Token</td>
</tr>
<tr>
<td>Request</td>
<td>Header parameters</td>
<td>Accept</td>
</tr>
<tr>
<td>Request</td>
<td>URI parameter</td>
<td>{account}</td>
</tr>
<tr>
<td>Request</td>
<td>Query parameters</td>
<td>limit</td>
</tr>
<tr>
<td>Request</td>
<td>Query parameters</td>
<td>marker</td>
</tr>
<tr>
<td>Request</td>
<td>Query parameters</td>
<td>format</td>
</tr>
<tr>
<td>Request</td>
<td>Query parameters</td>
<td>prefix</td>
</tr>
<tr>
<td>Request</td>
<td>Query parameters</td>
<td>delimiter</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>Content-Length</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>Content-Type</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>X-Account-Object-Count</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>X-Account-Bytes-Used</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>X-Account-Container-Count</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>X-Account-Meta-name</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>X-Trans-Id</td>
</tr>
<tr>
<td>Response</td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>

POST Method

Table 10-2: Supported API Parameters for Account POST Requests and Responses

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Header parameters</td>
<td>X-Auth-Token</td>
</tr>
<tr>
<td>Request</td>
<td>Header parameters</td>
<td>X-Account-Meta-name</td>
</tr>
<tr>
<td>Request</td>
<td>Header parameters</td>
<td>Content-Type</td>
</tr>
<tr>
<td>Request</td>
<td>URI parameter</td>
<td>{account}</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>Content-Length</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>Content-Type</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>X-Trans-Id</td>
</tr>
<tr>
<td>Response</td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>
# HEAD Method

Table 10-3: Supported API Parameters for Account HEAD Requests and Responses

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Request</strong></td>
<td>Header parameters</td>
<td>X-Auth-Token</td>
</tr>
<tr>
<td></td>
<td>URI parameter</td>
<td>{account}</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>Header parameters</td>
<td>X- Account-Object-Count</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X- Account-Container-Count</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X- Account-Bytes-Used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X- Account-Meta-name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Trans-Id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>
Container API Functions

- GET Method
- PUT Method
- DELETE Method
- POST Method
- HEAD Method
GET Method

Table 10-4: Supported API Parameters for Container GET Requests and Responses

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Header parameters</td>
<td>X-Auth-Token</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accept</td>
</tr>
<tr>
<td></td>
<td>URI parameter</td>
<td>{account}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{container}</td>
</tr>
<tr>
<td></td>
<td>Query parameters</td>
<td>limit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>marker</td>
</tr>
<tr>
<td></td>
<td></td>
<td>format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>prefix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>delimiter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>path</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Container-Object-Count</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Container-Meta-name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Container-Bytes-Used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Trans-Id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>

PUT Method

Table 10-5: Supported API Parameters for Container PUT Requests and Responses

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Header parameters</td>
<td>X-Auth-Token</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Container-Meta-name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td>URI parameters</td>
<td>{account}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{container}</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Trans-Id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>
DELETE Method

Table 10-6: Supported API Parameters for Container DELETE Requests and Responses

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Header parameter</td>
<td>X-Auth-Token</td>
</tr>
<tr>
<td></td>
<td>URI parameters</td>
<td>{account}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{container}</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Trans-Id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>

POST Method

Table 10-7: Supported API Parameters for Container POST Requests and Responses

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Header parameters</td>
<td>X-Auth-Token</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Container-Meta-name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td>URI parameters</td>
<td>{account}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{container}</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Trans-Id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>

HEAD Method

Table 10-8: Supported API Parameters for Container HEAD Requests and Responses

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Header parameters</td>
<td>X-Auth-Token</td>
</tr>
<tr>
<td></td>
<td>URI parameters</td>
<td>{account}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{container}</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Container-Object-Count</td>
</tr>
</tbody>
</table>
Table 10-8: Supported API Parameters for Container HEAD Requests and Responses

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>X-Container-Meta-name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Container-Bytes-Used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Trans-Id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>
Object API Functions

- GET Method
- PUT Method
- COPY Method
- DELETE Method
- POST Method
- HEAD Method
## GET Method

Table 10-9: Supported API Parameters for Object GET Requests and Responses

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Header parameters</td>
<td>X-Auth-Token</td>
</tr>
<tr>
<td></td>
<td>URI parameters</td>
<td>{account}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{container}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{object}</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Last-Modified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ETag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Encoding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Disposition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Object-Meta-name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Trans-Id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>

## PUT Method

Table 10-10: Supported API Parameters for Object PUT Requests and Responses

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Header parameters</td>
<td>X-Auth-Token</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transfer-Encoding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Copy-From</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ETag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Disposition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Encoding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Object-Meta-name</td>
</tr>
<tr>
<td></td>
<td>URI parameters</td>
<td>{account}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{container}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{object}</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>Content-Length</td>
</tr>
</tbody>
</table>
### Table 10-10: Supported API Parameters for Object PUT Requests and Responses

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ETag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Trans-Id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>
COPY Method

Table 10-11: Supported API Parameters for Object COPY Requests and Responses

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Request</strong></td>
<td>Header parameters</td>
<td>X-Auth-Token</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Destination</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Encoding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Disposition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Object-Meta-name</td>
</tr>
<tr>
<td></td>
<td>URI parameters</td>
<td>{account}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{container}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{object}</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>Header parameters</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Copied-From</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Last-Modified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ETag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Object-Meta-name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Trans-Id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>

DELETE Method

Table 10-12: Supported API Parameters for Object DELETE Requests and Responses

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Request</strong></td>
<td>Header parameter</td>
<td>X-Auth-Token</td>
</tr>
<tr>
<td></td>
<td>URI parameters</td>
<td>{account}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{container}</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>Header parameters</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Trans-Id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>
### POST Method

Table 10-13: Supported API Parameters for Object POST Requests and Responses

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Header parameters</td>
<td>X-Auth-Token</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Container-Meta-name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td>URI parameters</td>
<td>{account}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{container}</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Trans-Id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>

### HEAD Method

Table 10-14: Supported API Parameters for Object HEAD Requests and Responses

<table>
<thead>
<tr>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Header parameters</td>
<td>X-Auth-Token</td>
</tr>
<tr>
<td></td>
<td>URI parameters</td>
<td>{account}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{container}</td>
</tr>
<tr>
<td>Response</td>
<td>Header parameters</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Container-Object-Count</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Container-Meta-name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Container-Bytes-Used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X-Trans-Id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>
Supported Amazon S3 API Functions In NexentaEdge

This chapter includes the following topics:

- Service API Functions
- Bucket API Functions
- Object API Functions
Service API Functions

- GET Operations

GET Operations

Table 11-1: Supported API Parameters for Service GET Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET Service</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Host</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-security-token</td>
</tr>
<tr>
<td>Response</td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td>Response elements</td>
<td></td>
<td></td>
<td>Bucket</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buckets</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CreationDate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ListAllMyBucketsResult</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Name</td>
</tr>
</tbody>
</table>
Bucket API Functions

- GET Operations
- PUT Bucket Operations
- DELETE Bucket Operations
- HEAD Operations
GET Operations
<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET Bucket</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td>(List Objects)</td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Host</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Request parameters</td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Request parameters</td>
<td>x-amz-date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response headers</td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response headers</td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>Contents</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>CommonPrefixes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>Delimiter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>Encoding-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>ETag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>IsTruncated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>Key</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>LastModified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>Marker</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>MaxKeys</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>NextMarker</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>Prefix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>Size</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>StorageClass</td>
</tr>
</tbody>
</table>
### GET acl

Table 11-3: Supported API Parameters for Bucket GET acl Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET Bucket acl</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Host</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Response elements</td>
<td>AccessControlPolicy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AccessControlList</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Owner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DisplayName</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Grantee</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Permission</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Grant</td>
</tr>
</tbody>
</table>
### GET cors

Table 11-4: Supported API Parameters for Bucket GET cors Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET Bucket cors</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Host</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td>Response</td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td>Response</td>
<td>Response</td>
<td>Response elements</td>
<td>CORSConfiguration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CORSRule</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AllowedMethod</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AllowedOrigin</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AllowedHeader</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MaxAgeSeconds</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ExposeHeader</td>
</tr>
</tbody>
</table>
# GET Bucket (Object Version)

Table 11-5: Supported API Parameters for Bucket GET Object Version Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET Bucket (Object Versions)</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Host</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td></td>
<td>Request parameters</td>
<td>delimiter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>encoding-type</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>marker</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>max-keys</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>prefix</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>version-id-marker</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td>Response elements</td>
<td>DeleteMarker</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DisplayName</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Encoding-Type</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ETag</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ID</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>IsLatest</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>IsTruncated</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Key</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>KeyMarker</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LastModified</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ListVersionsResult</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MaxKeys</td>
<td></td>
</tr>
</tbody>
</table>
### Table 11-5: Supported API Parameters for Bucket GET Object Version Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td></td>
<td>Name</td>
</tr>
<tr>
<td>NextKeyMarker</td>
<td></td>
<td></td>
<td>NextKeyMarker</td>
</tr>
<tr>
<td>NextVersionIdMarker</td>
<td></td>
<td></td>
<td>NextVersionIdMarker</td>
</tr>
<tr>
<td>Owner</td>
<td></td>
<td></td>
<td>Owner</td>
</tr>
<tr>
<td>Prefix</td>
<td></td>
<td></td>
<td>Prefix</td>
</tr>
<tr>
<td>Size</td>
<td></td>
<td></td>
<td>Size</td>
</tr>
<tr>
<td>StorageClass</td>
<td></td>
<td></td>
<td>StorageClass</td>
</tr>
<tr>
<td>Version</td>
<td></td>
<td></td>
<td>Version</td>
</tr>
<tr>
<td>VersionId</td>
<td></td>
<td></td>
<td>VersionId</td>
</tr>
<tr>
<td>VersionIdMarker</td>
<td></td>
<td></td>
<td>VersionIdMarker</td>
</tr>
</tbody>
</table>
GET Bucket versioning

Table 11-6: Supported API Parameters for Bucket GET versioning Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET Bucket</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td>versioning</td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Host</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td>Response</td>
<td>Response headers</td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td>Response elements</td>
<td></td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VersioningConfiguration</td>
</tr>
</tbody>
</table>
### PUT Bucket Operations

Table 11-7: Supported API Parameters for Bucket PUT Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT Bucket</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Host</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td>Response</td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Connection</td>
</tr>
</tbody>
</table>
## PUT Bucket acl Operations

Table 11-8: Supported API Parameters for Bucket PUT acl Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT Bucket acl</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Host</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Request elements</td>
<td>AccessControlList</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AccessControlPolicy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DisplayName</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Grant</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Grantee</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Owner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Permission</td>
</tr>
<tr>
<td>Response</td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>
## PUT Bucket cors Operations

Table 11-9: Supported API Parameters for Bucket PUT cors Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT Bucket cors</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Host</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Request elements</td>
<td>CORSConfiguration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CORSRule</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AllowedMethod</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AllowedOrigin</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AllowedHeader</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MaxAgeSeconds</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ExposeHeader</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Connection</td>
</tr>
</tbody>
</table>
## DELETE Bucket Operations

Table 11-10: Supported API Parameters for Bucket DELETE Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE Bucket</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td>Response</td>
<td></td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>

## DELETE Bucket cors

Table 11-11: Supported API Parameters for Bucket DELETE cors Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE Bucket cors</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td>Response</td>
<td></td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>
# HEAD Operations

Table 11-12: Supported API Parameters for Bucket HEAD Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEAD Bucket</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization, Content-Length, Content-Type, Content-MD5, Date, Expect, Host, x-amz-content-sha256, x-amz-date</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length, Content-Type, Date, ETag, Server</td>
</tr>
</tbody>
</table>
## List Multipart Upload Operations

Table 11-13: Supported API Parameters for List Multipart Upload Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>List Multipart Uploads</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Host</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Request parameters</td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>ListMultipartUploadsResult</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bucket</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>KeyMarker</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UploadIdMarker</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NextKeyMarker</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NextUploadIdMarker</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Encoding-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MaxUploads</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IsTruncated</td>
</tr>
<tr>
<td>Name</td>
<td>Operation</td>
<td>Parameter Type</td>
<td>Parameter Name</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------</td>
<td>----------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>Upload</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UploadId</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initiator</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DisplayName</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Owner</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StorageClass</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initiated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ListMultipartUploadsResult.Prefix</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delimiter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CommonPrefixes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CommonPrefixes.Prefix</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### PUT Bucket versioning Operations

#### Table 11-14: Supported API Parameters for Bucket PUT versioning Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT Bucket versioning</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Host</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td></td>
<td>Request</td>
<td>Request elements</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VersioningConfiguration</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Response elements</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VersioningConfiguration</td>
</tr>
</tbody>
</table>
Object API Functions

- GET Object Operations
- HEAD Operations
- POST Operations
- PUT Operations
- Upload Part Operations
- DELETE Operations
### GET Object Operations

#### Table 11-15: Supported API Parameters for Object GET Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET Object</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Disposition</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Encoding</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Host</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Range</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-meta-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If-Modified-Since</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If-Unmodified-Since</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If-Match</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If-None-Match</td>
</tr>
<tr>
<td>Response</td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ETag</td>
</tr>
</tbody>
</table>
### GET Object acl Operations

Table 11-16: Supported API Parameters for Object GET acl Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET Object acl</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td>Response</td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td>Response</td>
<td>Response</td>
<td>Response elements</td>
<td>AccessControlPolicy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AccessControlList</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Owner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DisplayName</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Grantee</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Permission</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Grant</td>
</tr>
</tbody>
</table>
## HEAD Operations

Table 11-17: Supported API Parameters for Object HEAD Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEAD Object</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization, Content-Length,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type, Content-MD5, Content-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Disposition, Content-Encoding,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date, Expect, Host, Range,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-sha256, x-amz-date,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-meta-*, If-Modified-Since,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If-Unmodified-Since, If-Match,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If-None-Match</td>
</tr>
<tr>
<td>Response</td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length, Content-Type, Date,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ETag</td>
</tr>
</tbody>
</table>
OPTIONS Object Operations

Table 11-18: Supported API Parameters for Object OPTIONS Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTIONS Object</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Origin</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Access-Control-Request-Method</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Access-Control-Request-Headers</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Access-Control-Allow-Origin</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Access-Control-Max-Age</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Access-Control-Allow-Methods</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Access-Control-Allow-Headers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Access-Control-Expose-Headers</td>
</tr>
</tbody>
</table>
## POST Operations

Table 11-19: Supported API Parameters for Object POST Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST Object</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization, Content-Length, Content-Type, Content-MD5, Date, Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Form fields</td>
<td>AWSAccessKeyId, Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires, file, key, success_action_redirect, redirect, success_action_status, x-amz-storage-class, x-amz-meta-*</td>
</tr>
<tr>
<td>Response</td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length, Content-Type, Connection, Date, x-amz-version-id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response elements</td>
<td>Bucket, ETag, Key</td>
</tr>
</tbody>
</table>
## PUT Operations

Table 11-20: Supported API Parameters for Object PUT Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT Object</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Disposition</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Encoding</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response headers</td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-meta-</td>
</tr>
<tr>
<td>Response</td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ETag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-version-id</td>
</tr>
</tbody>
</table>
# PUT copy Operations

Table 11-21: Supported API Parameters for Object PUT copy Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT Object copy Request</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Disposition</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Encoding</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-meta-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-copy-source</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-metadata-directive</td>
</tr>
<tr>
<td>Response</td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-version-id</td>
</tr>
<tr>
<td>Response elements</td>
<td></td>
<td></td>
<td>CopyObjectResult</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ETag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LastModified</td>
</tr>
</tbody>
</table>
# Initiate Multipart Upload Operations

Table 11-22: Supported API Parameters for Initiate Multipart Upload Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiate Multipart Upload</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Disposition</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Encoding</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-meta-*</td>
</tr>
<tr>
<td>Response</td>
<td>Response headers</td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td>Response elements</td>
<td></td>
<td>InitiateMultipartUploadResult</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bucket</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Key</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UploadId</td>
</tr>
</tbody>
</table>
## Upload Part Operations

Table 11-23: Supported API Parameters for Object Upload Part Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload Part</td>
<td>Request</td>
<td>Request parameters</td>
<td>partNumber</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>uploadId</td>
</tr>
<tr>
<td></td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>
# Complete Multipart Upload Operations

Table 11-24: Supported API Parameters for Complete Multipart Upload Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete Multipart Upload</td>
<td>Request</td>
<td>Request parameters</td>
<td>uploadId</td>
</tr>
<tr>
<td></td>
<td>Request headers</td>
<td>Authorization</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-MD5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Expect</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>x-amz-date</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Request elements</td>
<td>CompleteMultipartUpload</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Part</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PartNumber</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ETag</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content-Type</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Response elements</td>
<td>CompleteMultipartUploadResult</td>
<td>Location</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bucket</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Key</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ETag</td>
<td></td>
</tr>
</tbody>
</table>
Abort Multipart Upload Operations

Table 11-25: Supported API Parameters for Abort Multipart Upload Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abort Multipart Upload</td>
<td>Request</td>
<td>Request parameters</td>
<td>uploadId</td>
</tr>
<tr>
<td></td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>
List Parts Operations
### Table 11-26: Supported API Parameters for List Parts Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>List Parts</td>
<td>Request</td>
<td>Request parameters</td>
<td>encoding-type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>uploadId</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>max-parts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>part-number-marker</td>
</tr>
<tr>
<td></td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-date</td>
</tr>
<tr>
<td>Response</td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td>Response</td>
<td>Response</td>
<td>Response elements</td>
<td>ListPartsResult</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bucket</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Encoding-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Key</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UploadId</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Initiator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DisplayName</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Owner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PartNumberMarker</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NextPartNumberMarker</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MaxParts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IsTruncated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Part</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PartNumber</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LastModified</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ETag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Size</td>
</tr>
</tbody>
</table>
# DELETE Operations

Table 11-27: Supported API Parameters for Object DELETE Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE Object</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Host</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x-amz-content-sha256</td>
</tr>
<tr>
<td>Response</td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Date</td>
</tr>
</tbody>
</table>
Multiple Object DELETE Operations

Table 11-28: Supported API Parameters for Multiple Object DELETE Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameter Type</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE Multiple Objects</td>
<td>Request</td>
<td>Request headers</td>
<td>Authorization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-MD5</td>
</tr>
<tr>
<td></td>
<td>Request elements</td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Expect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Host</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Response headers</td>
<td>Content-Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Content-Type</td>
</tr>
<tr>
<td></td>
<td>Response elements</td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DeleteResult</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Deleted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Key</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VersionId</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DeleteMarker</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DeleteMarkerVersionId</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Key</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VersionId</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Message</td>
</tr>
</tbody>
</table>