Red Hat Ceph Storage Developer

Use the various application programming interfaces (APIs) for Red Hat Ceph Storage running on AMD64 and Intel 64 architectures.

Ceph RESTful API

As a storage administrator, you can use the Ceph RESTful API, or simply the Ceph API, provided by the Red Hat Ceph Storage Dashboard to interact with the Red Hat Ceph Storage cluster. You can display information about the Ceph Monitors and OSDs, along with their respective configuration options. You can also create or edit Ceph pools.

The Ceph API uses the following standards:

- HTTP 1.1
- JSON
- MIME and HTTP Content Negotiation
- JWT

These standards are OpenAPI 3.0 compliant, regulating the API syntax, semantics, content encoding, versioning, authentication, and authorization.

Prerequisites

- A healthy running Red Hat Ceph Storage cluster.
- Access to the node running the Ceph Manager.

Versioning for the Ceph API

The main goal for the Ceph RESTful API is to provide a stable interface, which is done by versioning.

To achieve a stable interface, the Ceph API is built on the following principles:

- A mandatory explicit default version for all endpoints to avoid implicit defaults.
- Fine-grain change control per-endpoint.
  - The expected version from a specific endpoint is stated in the HTTP header.

    Syntax

    ```
    Accept: application/vnd.ceph.api.vMAJOR.MINOR+json
    ```

    Example

    ```
    Accept: application/vnd.ceph.api.v1.0+json
    ```

    If the current Ceph API server is not able to address that specific version, a `415 - Unsupported Media Type` response will be returned.
- Using semantic versioning.
  - Major changes are backwards incompatible. Changes might result in non-additive changes to the request, and to the response formats for a specific endpoint.
  - Minor changes are backwards and forwards compatible. Changes consist of additive changes to the request or response formats for a specific endpoint.

Authentication and authorization for the Ceph API

Access to the Ceph RESTful API goes through two checkpoints. The first is authenticating that the request is done on the behalf of a valid, and existing user. Secondly, is authorizing the previously authenticated user can do a specific action, such as creating, reading, updating, or deleting, on the target end point.

Before users start using the Ceph API, they need a valid JSON Web Token (JWT). The `/api/auth` endpoint allows you to retrieve this token.

Example

```
[root@mon ~]# curl -X POST "https://example.com:8443/api/auth" \
  -H  "Accept: application/vnd.ceph.api.v1.0+json" \
  -H  "Content-Type: application/json" \
  -d '{"username": user1, "password": password1}'
```

This token must be used together with every API request by placing it within the `Authorization` HTTP header.

Syntax

```
curl -H "Authorization: Bearer TOKEN" ...
```

For more information, see Ceph user management.

Enabling and Securing the Ceph API module

The Red Hat Ceph Storage Dashboard module offers the RESTful API access to the storage cluster over an SSL-secured connection.

> **Important:** If disabling SSL, then user names and passwords are sent unencrypted to the Red Hat Ceph Storage Dashboard.

Prerequisites

- Root-level access to a Ceph Monitor node.
- Ensure that you have at least one `ceph-mgr` daemon active.
- If you use a firewall, ensure that TCP port `8443`, for SSL, and TCP port `8080`, without SSL, are open on the node with the active `ceph-mgr` daemon.

Procedure

1. Log into the `cephadm` shell:

   Example

   ```
   root@host01 ~]# cephadm shell
   ```

2. Enable the RESTful plug-in:

   ```
   [ceph: root@host01 /]# ceph mgr module enable dashboard
   ```

3. Configure an SSL certificate.

   a. If your organization's certificate authority (CA) provides a certificate, then set using the certificate files:

      Syntax

      ```
      ceph dashboard set-ssl-certificate HOST_NAME -i CERT_FILE
      ceph dashboard set-ssl-certificate-key HOST_NAME -i KEY_FILE
      ```

      Example

```
[ceph: root@host01 /]# ceph dashboard set-ssl-certificate -i dashboard.crt
[ceph: root@host01 /]# ceph dashboard set-ssl-certificate-key -i dashboard.key
```

If you want to set unique node-based certificates, then add a *HOST_NAME* to the commands:

Example

```
[ceph: root@host01 /]# ceph dashboard set-ssl-certificate host01 -i dashboard.crt
[ceph: root@host01 /]# ceph dashboard set-ssl-certificate-key host01 -i
dashboard.key
```

b. Alternatively, you can generate a self-signed certificate. However, using a self-signed certificate does not provide full security benefits of the HTTPS protocol:

```
[ceph: root@host01 /]# ceph dashboard create-self-signed-cert
```

> **Warning:** Most modern web browsers will complain about self-signed certificates, which require you to confirm before establishing a secure connection.

4. Create a user, set the password, and set the role:

   Syntax

   ```
   echo -n "PASSWORD" > PATH_TO_FILE/PASSWORDFILE
   ceph dashboard ac-user-create USERNAME -i PASSWORDFILE ROLE
   ```

   Example

   ```
   [ceph: root@host01 /]# echo -n "p@ssw0rd" > /root/dash-password.txt
   [ceph: root@host01 /]# ceph dashboard ac-user-create user1 -i /root/dash-password.txt
   administrator
   ```

   This example creates a user named `user1` with the `administrator` role.

5. Connect to the RESTful plug-in web page. Open a web browser and enter the following URL:

   Syntax

   ```
   https://HOST_NAME:8443
   ```

   Example

   ```
   https://host01:8443
   ```

   If you used a self-signed certificate, confirm a security exception.

Reference

- The **`ceph dashboard --help`** command.
- The `https://`*`HOST_NAME`*`:8443/doc` page, where *HOST_NAME* is the IP address or name of the node with the running `ceph-mgr` instance.
- The *Security hardening* guide for your Red Hat Enterprise Linux version on Red Hat Documentation.

Using the Ceph RESTful API

This section describes how to use the Ceph API to view information about the storage cluster, Ceph Monitors, OSDs, pools, and hosts.

Viewing information can be done by either using the command line interface (CLI), Python, or a web browser.

Viewing cluster configuration options

Understand how to use the RESTful plug-in to view cluster configuration options and their values.

Viewing cluster configuration options using the `curl` command

On the command line, run the following command:

```
curl --silent --user USER 'https://CEPH_MANAGER:CEPH_MANAGER_PORT/api/cluster_conf'
```

Replace:

- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *CEPH_MANAGER_PORT* with the TCP port number.
  The default TCP port number is 8443.

Enter the user's password when prompted.

> **Note:** If using a self-signed certificate, use the `--insecure` option.
>
> ```
> curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/cluster_conf'
> ```

Viewing cluster configuration options using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/cluster_conf', auth=("USER",
"PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *USER* with the username.
- *PASSWORD* with the user's password.

> **Note:** If using a self-signed certificate, use `verify=False`.
>
> ```
> $ python
> >> import requests
> >> result = requests.get('https://CEPH_MANAGER:8080/api/cluster_conf', auth=("USER",
> "PASSWORD"), verify=False)
> >> print result.json()
> ```

Viewing cluster configuration options using a web browser

In the web browser, go to `https://CEPH_MANAGER:8080/api/cluster_conf`.

Replace *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.

Enter the username and password when prompted.

Related information

Configuring

Viewing a particular cluster configuration option

Understand how to view a particular cluster option and its value.

Viewing a particular cluster configuration option using the `curl` command

On the command line, run the following command.

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/cluster_conf/ARGUMENT'
```

Replace:

- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ARGUMENT* with the configuration option you want to view.

Enter the user's password when prompted.

> **Note:** If you used a self-signed certificate, use the `--insecure` option.
>
> ```
> curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/cluster_conf/
> ARGUMENT'
> ```

Viewing a particular cluster configuration option using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/cluster_conf/ARGUMENT',
auth=("USER", "PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ARGUMENT* with the configuration option you want to view.
- *USER* with the username.
- *PASSWORD* with the user's password.

> **Note:** If using a self-signed certificate, use `verify=False`.
>
> ```
> $ python
> >> import requests
> >> result = requests.get('https://CEPH_MANAGER:8080/api/cluster_conf/ARGUMENT',
> auth=("USER", "PASSWORD"), verify=False)
> >> print result.json()
> ```

Viewing a particular cluster configuration option using a web browser

In the web browser, go to `https://CEPH_MANAGER:8080/api/cluster_conf/ARGUMENT`.

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ARGUMENT* with the configuration option you want to view.

Enter the username and password when prompted.

Related information

Viewing all configuration options for OSDs

Understand how to view all configuration options and their values for OSDs.

Viewing all configuration options for OSDs using the `curl` command

On the command line, run the following command.

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/osd/flags'
```

Replace:

- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.

Enter the user's password when prompted.

> **Note:** If you used a self-signed certificate, use the `--insecure` option.
>
> ```
> curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/osd/flags'
> ```

Viewing all configuration options for OSDs using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/flags', auth=("USER",
"PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *USER* with the username.
- *PASSWORD* with the user's password.

If you used a self-signed certificate, use the `verify=False` option.

> **Note:** If using a self-signed certificate, use `verify=False`.
>
> ```
> $ python
> >> import requests
> >> result = requests.get('https://CEPH_MANAGER:8080/api/osd/flags', auth=("USER",
> "PASSWORD"), verify=False))
> >> print result.json()
> ```

Viewing all configuration options for OSDs using a web browser

In the web browser, go to `userhttps://CEPH_MANAGER:8080/api/osd/flags`.

Replace *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.

Enter the username and password when prompted.

Related information

Configuring

Viewing CRUSH rules

Understand how to view CRUSH rules.

Viewing CRUSH rules using the `curl` command

On the command line, run the following command:

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/crush_rule'
```

Replace:

- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short host name of the node with the active `ceph-mgr` instance.

Enter the user's password when prompted.

> **Note:** If using a self-signed certificate, use the `--insecure` option.
>
> ```
> curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/crush_rule'
> ```

Viewing CRUSH rules using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/crush_rule', auth=("USER",
"PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *USER* with the username.
- *PASSWORD* with the user's password.

> **Note:** If using a self-signed certificate, use `verify=False`.
>
> ```
> $ python
> >> import requests
> >> result = requests.get('https://CEPH_MANAGER:8080/api/crush_rule', auth=("USER",
> "PASSWORD"), verify=False)
> >> print result.json()
> ```

Viewing CRUSH rules using a web browser

In the web browser, go to `https://CEPH_MANAGER:8080/api/crush_rule`.

Replace *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.

Enter the username and password when prompted.

Related information

CRUSH rules

Viewing Monitor information

Understand how to use the RESTful plug-in to view Monitor information, such as IP address, name, and quorum status.

Viewing Monitor information using the `curl` command

On the command line, run the following command:

```
curl --silent --user USER 'https://CEPH_MANAGER:CEPH_MANAGER_PORT/api/monitor'
```

Replace:

- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *CEPH_MANAGER_PORT* with the TCP port number.
  The default TCP port number is 8443.

Enter the user's password when prompted.

> **Note:** If using a self-signed certificate, use the `--insecure` option.
>
> ```
> curl --silent --insecure --user USER 'https://CEPH_MANAGER:CEPH_MANAGER_PORT/api/
> monitor'
> ```

Viewing Monitor information using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/monitor', auth=("USER",
"PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *USER* with the username.
- *PASSWORD* with the user's password.

> **Note:** If using a self-signed certificate, use `verify=False`.
>
> ```
> $ python
> >> import requests
> >> result = requests.get('https://CEPH_MANAGER:8080/api/monitor', auth=("USER",
> "PASSWORD"), verify=False)
> >> print result.json()
> ```

Viewing Monitor information using a web browser

In the web browser, go to `https://CEPH_MANAGER:8080/api/monitor`.

Replace *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.

Enter the username and password when prompted.

Viewing information about a particular Monitor

Understand how to use the RESTful plug-in to view information about a particular Monitor, such as IP address, name, and quorum status.

Viewing information about a particular Monitor using the `curl` command

On the command line, run the following command:

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/monitor/MON_NAME'
```

Replace:

- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *MON_NAME* with the short hostname of the Monitor.

Enter the user's password when prompted.

Note: If using a self-signed certificate, use the `--insecure` option.

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/monitor/MON_NAME'
```

Viewing information about a particular Monitor using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/monitor/MON_NAME', auth=("USER",
"PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *MON_NAME* with the short hostname of the Monitor
- *USER* with the username.
- *PASSWORD* with the user's password.

Note: If using a self-signed certificate, use `verify=False`.

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/monitor/MON_NAME',
auth=("USER", "PASSWORD"), verify=False)
>> print result.json()
```

Viewing information about a particular Monitor using a web browser

In the web browser, go to `https://CEPH_MANAGER:8080/api/monitor/MON_NAME`.

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *MON_NAME* with the short hostname of the Monitor

Enter the username and password when prompted.

Viewing OSD information

Understand how to use the RESTful plug-in to view OSD information, such as IP address, Its pools, affinity, and weight.

Viewing OSD information using the `curl` command

On the command line, run the following command:

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/osd'
```

Replace:

- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.

Enter the user's password when prompted.

> **Note:** If using a self-signed certificate, use the `--insecure` option.
>
> ```
> curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/osd'
> ```

Viewing OSD information using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/', auth=("USER", "PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *USER* with the username.
- *PASSWORD* with the user's password.

> **Note:** If using a self-signed certificate, use `verify=False`.
>
> ```
> $ python
> >> import requests
> >> result = requests.get('https://CEPH_MANAGER:8080/api/osd/', auth=("USER",
> "PASSWORD"), verify=False)
> >> print result.json()
> ```

Viewing OSD information using a web browser

In the web browser, go to `https://CEPH_MANAGER:8080/api/osd`.

Replace *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.

Enter the username and password when prompted.

Viewing information about a particular OSD

Understand how to use the RESTful plug-in to view information about a particular OSD, such as IP address, Its pools, affinity, and weight.

Viewing information about a particular OSD using the `curl` command

On the command line, run the following command:

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/osd/ID'
```

Replace:

- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the OSD listed in the `osd` field

Enter the user's password when prompted.

Note: If using a self-signed certificate, use the `--insecure` option.

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/osd/ID'
```

Viewing information about a particular OSD using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/ID', auth=("USER", "PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the OSD listed in the `osd` field
- *USER* with the username.
- *PASSWORD* with the user's password.

Note: If using a self-signed certificate, use `verify=False`.

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/ID', auth=("USER",
"PASSWORD"), verify=False)
>> print result.json()
```

Viewing information about a particular OSD using a web browser

In the web browser, go to `https://CEPH_MANAGER:8080/api/osd/ID`.

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the OSD listed in the `osd` field

Enter the username and password when prompted.

Determining processes that can be scheduled on an OSD

Understand how to use the RESTful plug-in to view which processes, such as scrubbing or deep scrubbing, can be scheduled on an OSD.

Determining processes that can be scheduled on an OSD using the `curl` command

On the command line, run the following command:

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/osd/ID/command'
```

Replace:

- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the OSD listed in the `osd` field

Enter the user's password when prompted.

Note: If using a self-signed certificate, use the `--insecure` option.

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/osd/ID/command'
```

Determining processes that can be scheduled on an OSD using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/ID/command', auth=("USER",
"PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the OSD listed in the `osd` field
- *USER* with the username.
- *PASSWORD* with the user's password.

Note: If using a self-signed certificate, use `verify=False`.

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/ID/command', auth=("USER",
"PASSWORD"), verify=False)
>> print result.json()
```

Determining processes that can be scheduled on an OSD using a web browser

In the web browser, go to `https://CEPH_MANAGER:8080/api/osd/ID/command`.

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the OSD listed in the `osd` field

Enter the username and password when prompted.

Viewing pool information

Understand how to use the RESTful plug-in to view pool information, such as flags, size, and number of placement groups (PGs).

Viewing pool information using the `curl` command

On the command line, run the following command:

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/pool'
```

Replace:

- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.

Enter the user's password when prompted.

Note: If using a self-signed certificate, use the `--insecure` option.

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/pool'
```

Viewing pool information using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/pool', auth=("USER", "PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *USER* with the username.
- *PASSWORD* with the user's password.

Note: If using a self-signed certificate, use `verify=False`.

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/pool', auth=("USER",
"PASSWORD"), verify=False)
>> print result.json()
```

Viewing pool information using a web browser

In the web browser, go to `https://CEPH_MANAGER:8080/api/pool`.

Replace *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.

Enter the username and password when prompted.

Viewing information about a particular pool

Understand how to use the RESTful plug-in to view information about a particular pool, such as flags, size, and number of placement groups (PGs).

Viewing information about a particular pool using the `curl` command

On the command line, run the following command:

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/pool/ID'
```

Replace:

- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the OSD listed in the `osd` field

Enter the user's password when prompted.

Note: If using a self-signed certificate, use the `--insecure` option.

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/pool/ID'
```

Viewing information about a particular pool using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/pool/ID', auth=("USER",
"PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the OSD listed in the `pool` field
- *USER* with the username.
- *PASSWORD* with the user's password.

Note: If using a self-signed certificate, use `verify=False`.

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/pool/ID', auth=("USER",
"PASSWORD"), verify=False)
>> print result.json()
```

Viewing information about a particular pool using a web browser

In the web browser, go to `https://CEPH_MANAGER:8080/api/pool/ID`.

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the OSD listed in the `pool` field

Enter the username and password when prompted.

Viewing host information

Understand how to use the RESTful plug-in to view host information, such as hostnames, Ceph daemons and their IDs, and Ceph versions.

Viewing host information using the `curl` command

On the command line, run the following command:

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/host'
```

Replace:

- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.

Enter the user's password when prompted.

Note: If using a self-signed certificate, use the `--insecure` option.

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/host'
```

Viewing host information using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/host', auth=("USER", "PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *USER* with the username.
- *PASSWORD* with the user's password.

Note: If using a self-signed certificate, use `verify=False`.

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/host', auth=("USER",
"PASSWORD"), verify=False)
>> print result.json()
```

Viewing host information using a web browser

In the web browser, go to `https://CEPH_MANAGER:8080/api/host`.

Replace *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.

Enter the username and password when prompted.

Viewing information about a particular host

Understand how to use the RESTful plug-in to view information about a particular host, such as hostnames, Ceph daemons and their IDs, and Ceph versions.

Viewing information about a particular host using the `curl` command

On the command line, run the following command:

```
curl --silent --user USER 'https://CEPH_MANAGER:8080/api/host/HOSTNAME'
```

Replace:

- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *HOSTNAME* with the host name of the host listed in the `host` field.

Enter the user's password when prompted.

Note: If using a self-signed certificate, use the `--insecure` option.

```
curl --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/host/HOSTNAME'
```

Viewing information about particular host using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/host/HOSTNAME', auth=("USER",
"PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *HOSTNAME* with the host name of the host listed in the `host` field
- *USER* with the username.
- *PASSWORD* with the user's password.

Note: If using a self-signed certificate, use `verify=False`.

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/host', auth=("USER",
"PASSWORD"), verify=False)
>> print result.json()
```

Viewing information about a particular host using a web browser

In the web browser, go to `https://CEPH_MANAGER:8080/api/host/HOSTNAME`.

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *HOSTNAME* with the host name of the host listed in the `host` field.

Enter the username and password when prompted.

Changing OSD configurations

Learn how to use the Ceph API to change OSD configuration options, the state of an OSD, and information about pools.

Changing the OSD configurations can be done by using either the command line interface (CLI) or Python.

Changing OSD configuration options

Understand how to use the RESTful plug-in to change OSD configuration options.

Changing OSD configuration options using the `curl` command

On the command line, run the following command:

```
echo -En '{"=OPTION": VALUE}' | curl --request PATCH --data @- --silent --
user USER 'https://CEPH_MANAGER:8080/api/osd/flags'
```

Replace:

- *OPTION* with one of the modifying options:`pause`,`noup`,`nodown`,`noout`,`noin`,`nobackfill`,`norecover`, `noscrub`, or`nodeep-scrub`.
- *VALUE* with`true`or`false`.
- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.

Enter the user's password when prompted.

Note: If using a self-signed certificate, use the`--insecure`option.

```
echo -En '{"=OPTION": VALUE}' | curl --request PATCH --data @- --silent --insecure --
user USER 'https://CEPH_MANAGER:8080/api/osd/flags'
```

Changing OSD configuration options using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/flags', json={"OPTION": VALUE},
auth=("USER", "PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *OPTION* with one of the modifying options:`pause`,`noup`,`nodown`,`noout`,`noin`,`nobackfill`,`norecover`, `noscrub`, ornodeep-scrub.
- *VALUE* with`true`or`false`.
- *USER* with the username.
- *PASSWORD* with the user's password.

Note: If using a self-signed certificate, use `verify=False`.

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/flags',
json={"OPTION": VALUE}, auth=("USER", "PASSWORD"), verify=False)
>> print result.json()
```

Changing an OSD state

Understand how to use the RESTful plug-in to change the state of an OSD.

Changing the state of an OSD using the `curl` command

On the command line, run the following command:

```
echo -En '{"=STATE": VALUE}' | curl --request PATCH --data @- --silent --
user USER 'https://CEPH_MANAGER:8080/api/osd/ID'
```

Replace:

- *STATE* with either`in`or`up`.
- *VALUE* with`true`or`false`.
- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the OSD listed in the `osd` field.

Enter the user's password when prompted.

Note: If using a self-signed certificate, use the`--insecure`option.

```
echo -En '{"=STATE": VALUE}' | curl --request PATCH --data @- --silent --insecure --
user USER 'https://CEPH_MANAGER:8080/api/osd/ID'
```

Changing the state of an OSD using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/ID', json={"STATE": VALUE},
auth=("USER", "PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *STATE* with either `in` or `up`.
- *VALUE* with `true` or `false`.
- *USER* with the username.
- *PASSWORD* with the user's password.
- *ID* with the ID of the OSD listed in the `osd` field.

Note: If using a self-signed certificate, use `verify=False`.

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/ID',
json={"STATE": VALUE}, auth=("USER", "PASSWORD"), verify=False)
>> print result.json()
```

Changing the weight of an OSD

Understand how to use the RESTful plug-in to change the weight of an OSD.

Changing the weight of an OSD using the `curl` command

On the command line, run the following command:

```
echo -En '{"=reweight": VALUE}' | curl --request PATCH --data @- --silent --
user USER 'https://CEPH_MANAGER:8080/api/osd/ID'
```

Replace:

- *VALUE* with `true` or `false`.
- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the OSD listed in the `osd` field.

Enter the user's password when prompted.

Note: If using a self-signed certificate, use the `--insecure` option.

```
echo -En '{"=reweight": VALUE}' | curl --request PATCH --data @- --silent --insecure
--user USER 'https://CEPH_MANAGER:8080/api/osd/ID'
```

Changing the weight of an OSD using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/ID', json={"reweight": VALUE},
auth=("USER", "PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *VALUE* with `true` or `false`.
- *USER* with the username.
- *PASSWORD* with the user's password.
- *ID* with the ID of the OSD listed in the `osd` field.

Note: If using a self-signed certificate, use `verify=False`.

```
$ python
>> import requests
>> result = requests.get('https://CEPH_MANAGER:8080/api/osd/ID',
json={"reweight": VALUE}, auth=("USER", "PASSWORD"), verify=False)
>> print result.json()
```

Changing pool information

Understand how to use the RESTful plug-in to change pool information.

Changing pool information using the `curl` command

On the command line, run the following command:

```
echo -En '{"=OPTION": VALUE}' | curl --request PATCH --data @- --silent --
user USER 'https://CEPH_MANAGER:8080/api/pool/ID'
```

Replace:

- *OPTION* with the option to modify.
- *VALUE* with the new value of the option.
- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the pool listed in the `pool` field.

Enter the user's password when prompted.

Note: If using a self-signed certificate, use the `--insecure` option.

```
echo -En '{"=OPTION": VALUE}' | curl --request PATCH --data @- --silent --insecure --
user USER 'https://CEPH_MANAGER:8080/api/pool/ID'
```

Changing pool information using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.patch('https://CEPH_MANAGER:8080/api/pool/ID', json={"OPTION": VALUE},
auth=("USER", "PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the pool listed in the `pool` field.

- *OPTION* with one of the modifying options:`pause`,`noup`,`nodown`,`noout`,`noin`,`nobackfill`,`norecover`, `noscrub`, or`nodeep-scrub`.
- *VALUE* with`true`or`false`.
- *USER* with the username.
- *PASSWORD* with the user's password.

Note: If using a self-signed certificate, use `verify=False`.

```
$ python
>> import requests
>> result = requests.patch('https://CEPH_MANAGER:8080/api/pool/ID',
json={"OPTION": VALUE}, auth=("USER", "PASSWORD"), verify=False)
>> print result.json()
```

Administering the cluster

Learn how to use the Ceph API to initialize scrubbing or deep scrubbing on an OSD, create a pool or remove data from a pool, remove requests, or create a request.

Running a scheduled process on an OSD

Understand how to use the RESTful plug-in to run a scheduled process on an OSD, such as scrubbing or deep scrubbing.

Running a scheduled process on an OSD using the `curl` command

On the command line, run the following command:

```
echo -En '{"command": "COMMAND"}' | curl --request POST --data @- --silent --
user USER 'https://CEPH_MANAGER:8080/api/osd/ID/command'
```

Replace:

- *COMMAND* with the process you want to start. The options are`scrub`,`deep-scrub`, or`repair`. Verify if the process is supported on the OSD. For more information, see "Determining processes that can be scheduled on an OSD" on page 11.
- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the OSD listed in the `osd` field

Enter the user's password when prompted.

Note: If using a self-signed certificate, use the`--insecure`option.

```
echo -En '{"command": "COMMAND"}' | curl --request POST --data @- --silent --insecure
--user USER 'https://CEPH_MANAGER:8080/api/osd/ID/command'
```

Running a scheduled process on an OSD using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.post('https://CEPH_MANAGER:8080/api/osd/ID/command', json={"command":
"COMMAND"}, auth=("USER", "PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the OSD listed in the `osd` field
- *COMMAND* with the process you want to start. The options are `scrub`, `deep-scrub`, or `repair`. Verify if the process is supported on the OSD. For more information, see "Determining processes that can be scheduled on an OSD" on page 11.
- *USER* with the username.
- *PASSWORD* with the user's password.

Note: If using a self-signed certificate, use `verify=False`.

```
$ python
>> import requests
>> result = requests.post('https://CEPH_MANAGER:8080/api/osd/ID/command',
json={"command": "COMMAND"}, auth=("USER", "PASSWORD"), verify=False)
>> print result.json()
```

Creating a pool

Understand how to use the RESTful plug-in to create a pool.

Creating a pool using the `curl` command

On the command line, run the following command:

```
echo -En '{"name": "NAME", "pg_num": NUMBER}' | curl --request POST --data @- --silent --
user USER 'https://CEPH_MANAGER:8080/api/pool'
```

Replace:

- *NAME* with the name of the new pool.
- *NUMBER* with the number of the placement groups.
- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.

Enter the user's password when prompted.

Note: If using a self-signed certificate, use the `--insecure` option.

```
echo -En '{"name": "NAME", "pg_num": NUMBER}' | curl --request POST --data @- --silent
--insecure --user USER 'https://CEPH_MANAGER:8080/api/pool'
```

Creating a pool using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.post('https://CEPH_MANAGER:8080/api/pool', json={"NAME": NUMBER},
auth=("USER", "PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *NAME* with the name of the new pool.

- *NUMBER* with the number of the placement groups.
- *USER* with the username.
- *PASSWORD* with the user's password.

Note: If using a self-signed certificate, use `verify=False`.

```
$ python
>> import requests
>> result = requests.post('https://CEPH_MANAGER:8080/api/pool', json={"NAME": NUMBER},
auth=("USER", "PASSWORD"), verify=False)
>> print result.json()
```

Removing a pool

Understand how to use the RESTful plug-in to remove a pool.

Removing a pool is not allowed by default. To allow the removal of a pool, in the Ceph configuration file, add the following parameter:

```
mon_allow_pool_delete = true
```

Removing a pool using the `curl` command

On the command line, run the following command:

```
curl --request DELETE --silent --user USER 'https://CEPH_MANAGER:8080/api/pool/ID'
```

Replace:

- *USER* with the username.
- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the pool listed in the `pool` field.

Enter the user's password when prompted.

Note: If using a self-signed certificate, use the `--insecure` option.

```
curl --request DELETE --silent --insecure --user USER 'https://CEPH_MANAGER:8080/api/
pool/ID'
```

Removing a pool using Python

In the Python interpreter, enter the following:

```
$ python
>> import requests
>> result = requests.delete('https://CEPH_MANAGER:8080/api/pool/ID', auth=("USER",
"PASSWORD"))
>> print result.json()
```

Replace:

- *CEPH_MANAGER* with the IP address or short hostname of the node with the active `ceph-mgr` instance.
- *ID* with the ID of the pool listed in the `pool` field.
- *USER* with the username.
- *PASSWORD* with the user's password.

Note: If using a self-signed certificate, use `verify=False`.

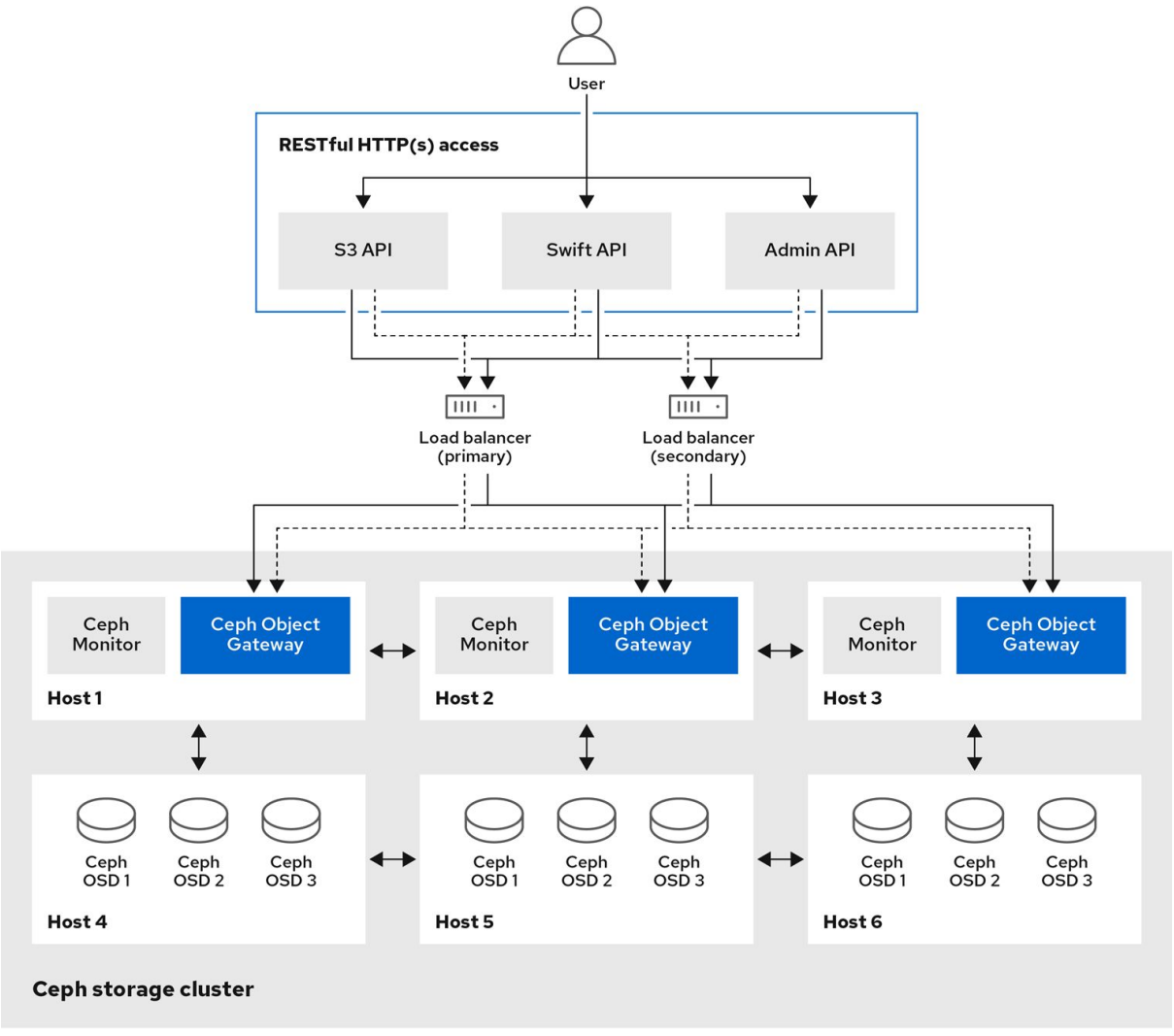
```
$ python
>> import requests
>> result = requests.delete('https://CEPH_MANAGER:8080/api/pool/ID', auth=("USER",
"PASSWORD"), verify=False)
>> print result.json()
```

Ceph Object Gateway administrative API

As a developer, you can administer the Ceph Object Gateway by interacting with the RESTful application programming interface (API). The Ceph Object Gateway makes available the features of the **`radosgw-admin`** command in a RESTful API. You can manage users, data, quotas, and usage which you can integrate with other management platforms.

Note: Use the command-line interface when configuring the Ceph Object Gateway.

Figure: Basic access



Before administering with the RESTful application programming interface (API), be sure that you have both a running Red Hat Ceph Storage cluster and a RESTful client.

The administrative API provides the following functionality:

Administration operations

An administrative Application Programming Interface (API) request will be done on a URI that starts with the configurable *admin* resource entry point. Authorization for the administrative API duplicates the S3 authorization mechanism.

Some operations require that the user holds special administrative capabilities. The response entity type, either XML or JSON, might be specified as the *format* option in the request and defaults to JSON if not specified.

Example

```
PUT /admin/user?caps&format=json HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
Content-Type: text/plain
Authorization: AUTHORIZATION_TOKEN

usage=read
```

Administration authentication requests

Amazon's S3 service uses the access key and a hash of the request header and the secret key to authenticate the request. It has the benefit of providing an authenticated request, especially large uploads, without SSL overhead.

Most use cases for the S3 API involve using open-source S3 clients such as the `AmazonS3Client` in the Amazon SDK for Java or Python Boto. These libraries do not support the Ceph Object Gateway Admin API. You can subclass and extend these libraries to support the Ceph Admin API. Alternatively, you can create a unique Gateway client.

Creating an `execute()` method

The CephAdminAPI example class in this section illustrates how to create an `execute()` method that can take request parameters, authenticate the request, call the Ceph Admin API and receive a response.

> **Important:** The `CephAdminAPI` class example is not supported or intended for commercial use. It is for illustrative purposes only.

Calling the Ceph Object Gateway

The client code contains five calls to the Ceph Object Gateway to demonstrate CRUD operations:

- Create a User
- Get a User
- Modify a User
- Create a Subuser
- Delete a User

To use this example, get the `httpcomponents-client-4.5.3` Apache HTTP components. You can download it for example here: http://hc.apache.org/downloads.cgi. Then unzip the tar file, navigate to its `lib` directory and copy the contents to the `/jre/lib/ext` directory of the `JAVA_HOME` directory, or a custom classpath.

As you examine the CephAdminAPI class example, notice that the `execute()` method takes an HTTP method, a request path, an optional subresource, `null` if not specified, and a map of parameters. To run with subresources, for example, `subuser`, and `key`, you will need to specify the subresource as an argument in the `execute()` method.

The example method:

1. Builds a URI.
2. Builds an HTTP header string.

3. Instantiates an HTTP request, for example, `PUT`, `POST`, `GET`, `DELETE`.
4. Adds the `Date` header to the HTTP header string and the request header.
5. Adds the `Authorization` header to the HTTP request header.
6. Instantiates an HTTP client and passes it the instantiated HTTP request.
7. Makes a request.
8. Returns a response.

Building the header string

Building the header string is the portion of the process that involves Amazon's S3 authentication procedure. Specifically, the example method does the following:

1. Adds a request type, for example, `PUT`, `POST`, `GET`, `DELETE`.
2. Adds the date.
3. Adds the requestPath.

The request type should be uppercase with no leading or trailing white space. If you do not trim white space, authentication will fail. The date MUST be expressed in GMT, or authentication will fail.

The exemplary method does not have any other headers. The Amazon S3 authentication procedure sorts `x-amz` headers lexicographically. So if you are adding `x-amz` headers, be sure to add them lexicographically.

Once you have built the header string, the next step is to instantiate an HTTP request and pass it the URI. The exemplary method uses `PUT` for creating a user and subuser, `GET` for getting a user, `POST` for modifying a user and `DELETE` for deleting a user.

Once you instantiate a request, add the `Date` header followed by the `Authorization` header. Amazon's S3 authentication uses the standard `Authorization` header, and has the following structure:

```
Authorization: AWS _ACCESS_KEY_:_HASH_OF_HEADER_AND_SECRET_
```

The CephAdminAPI example class has a `base64Sha1Hmac()` method, which takes the header string and the secret key for the admin user, and returns a SHA1 HMAC as a base-64 encoded string. Each `execute()` call will invoke the same line of code to build the `Authorization` header:

```
httpRequest.addHeader("Authorization", "AWS " + this.getAccessKey() + ":" +
base64Sha1Hmac(headerString.toString(), this.getSecretKey()));
```

The following `CephAdminAPI` example class requires you to pass the access key, secret key, and an endpoint to the constructor. The class provides accessor methods to change them at runtime.

Example

```
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.time.OffsetDateTime;
import java.time.format.DateTimeFormatter;
import java.time.ZoneId;

import org.apache.http.HttpEntity;
import org.apache.http.NameValuePair;
import org.apache.http.Header;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpRequestBase;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpPut;
import org.apache.http.client.methods.HttpDelete;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.util.EntityUtils;
import org.apache.http.client.utils.URIBuilder;

import java.util.Base64;
import java.util.Base64.Encoder;
```

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.Mac;

import java.util.Map;
import java.util.Iterator;
import java.util.Set;
import java.util.Map.Entry;

public class CephAdminAPI {

        /*
         * Each call must specify an access key, secret key, endpoint and format.
         */
        String accessKey;
        String secretKey;
        String endpoint;
        String scheme = "http"; //http only.
        int port = 80;

        /*
         * A constructor that takes an access key, secret key, endpoint and format.
         */
        public CephAdminAPI(String accessKey, String secretKey, String endpoint){
                this.accessKey = accessKey;
                this.secretKey = secretKey;
                this.endpoint = endpoint;
        }

        /*
         * Accessor methods for access key, secret key, endpoint and format.
         */
        public String getEndpoint(){
                return this.endpoint;
        }

        public void setEndpoint(String endpoint){
                this.endpoint = endpoint;
        }

        public String getAccessKey(){
                return this.accessKey;
        }

        public void setAccessKey(String accessKey){
                this.accessKey = accessKey;
        }

        public String getSecretKey(){
                return this.secretKey;
        }

        public void setSecretKey(String secretKey){
                this.secretKey = secretKey;
        }

        /*
         * Takes an HTTP Method, a resource and a map of arguments and
         * returns a CloseableHTTPResponse.
         */
        public CloseableHttpResponse execute(String HTTPMethod, String resource,
                                        String subresource, Map arguments) {

                String httpMethod = HTTPMethod;
                String requestPath = resource;
                StringBuffer request = new StringBuffer();
                StringBuffer headerString = new StringBuffer();
                HttpRequestBase httpRequest;
                CloseableHttpClient httpclient;
                URI uri;
                CloseableHttpResponse httpResponse = null;

                try {

                        uri = new URIBuilder()
                                .setScheme(this.scheme)
                                .setHost(this.getEndpoint())
                                .setPath(requestPath)
                                .setPort(this.port)
                                .build();
```

```
                if (subresource != null){
                        uri = new URIBuilder(uri)
                                .setCustomQuery(subresource)
                                .build();
                }


                for (Iterator iter = arguments.entrySet().iterator();
                iter.hasNext();) {
                        Entry entry = (Entry)iter.next();
                        uri = new URIBuilder(uri)
                                .setParameter(entry.getKey().toString(),
                         entry.getValue().toString())
                                .build();

                }

                request.append(uri);

                headerString.append(HTTPMethod.toUpperCase().trim() + "\n\n\n");

                OffsetDateTime dateTime = OffsetDateTime.now(ZoneId.of("GMT"));
                DateTimeFormatter formatter = DateTimeFormatter.RFC_1123_DATE_TIME;
                String date = dateTime.format(formatter);

                headerString.append(date + "\n");
                headerString.append(requestPath);

                if (HTTPMethod.equalsIgnoreCase("PUT")){
                        httpRequest = new HttpPut(uri);
                } else if (HTTPMethod.equalsIgnoreCase("POST")){
                        httpRequest = new HttpPost(uri);
                } else if (HTTPMethod.equalsIgnoreCase("GET")){
                        httpRequest = new HttpGet(uri);
                } else if (HTTPMethod.equalsIgnoreCase("DELETE")){
                        httpRequest = new HttpDelete(uri);
                } else {
                        System.err.println("The HTTP Method must be PUT,
                        POST, GET or DELETE.");
                        throw new IOException();
                }

                httpRequest.addHeader("Date", date);
                httpRequest.addHeader("Authorization", "AWS " + this.getAccessKey()
                + ":" + base64Sha1Hmac(headerString.toString(),
                this.getSecretKey()));

                httpclient = HttpClients.createDefault();
                httpResponse = httpclient.execute(httpRequest);

        }       catch  (URISyntaxException e){
                System.err.println("The URI is not formatted properly.");
                e.printStackTrace();
        }  catch (IOException e){
                System.err.println("There was an error making the request.");
                e.printStackTrace();
        }
                return httpResponse;
}

/*
 * Takes a uri and a secret key and returns a base64-encoded
 * SHA-1 HMAC.
 */
public String base64Sha1Hmac(String uri, String secretKey) {
        try {

                byte[] keyBytes = secretKey.getBytes("UTF-8");
                SecretKeySpec signingKey = new SecretKeySpec(keyBytes, "HmacSHA1");

                Mac mac = Mac.getInstance("HmacSHA1");
                mac.init(signingKey);

                byte[] rawHmac = mac.doFinal(uri.getBytes("UTF-8"));

                Encoder base64 = Base64.getEncoder();
                return base64.encodeToString(rawHmac);

        } catch (Exception e) {
```

```
                        throw new RuntimeException(e);
                }
        }

}
```

The subsequent `CephAdminAPIClient` example illustrates how to instantiate the `CephAdminAPI` class, build a map of request parameters, and use the `execute()` method to create, get, update and delete a user.

Example

```
import java.io.IOException;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.HttpEntity;
import org.apache.http.util.EntityUtils;
import java.util.*;


public class CephAdminAPIClient {

        public static void main (String[] args){

                CephAdminAPI adminApi = new CephAdminAPI ("FFC6ZQ6EMIF64194158N",
                                                "Xac39eCAhlTGcCAUreuwe1ZuH5oVQFa51lbEMVoT",
                                                "ceph-client");

                /*
                 * Create a user
                 */
                Map requestArgs = new HashMap();
                requestArgs.put("access", "usage=read, write; users=read, write");
                requestArgs.put("display-name", "New User");
                requestArgs.put("email", "new-user@email.com");
                requestArgs.put("format", "json");
                requestArgs.put("uid", "new-user");

                CloseableHttpResponse response =
                        adminApi.execute("PUT", "/admin/user", null, requestArgs);

                System.out.println(response.getStatusLine());
                HttpEntity entity = response.getEntity();

                try {
                        System.out.println("\nResponse Content is: "
                                + EntityUtils.toString(entity, "UTF-8") + "\n");
                        response.close();
                } catch (IOException e){
                        System.err.println ("Encountered an I/O exception.");
                        e.printStackTrace();
                }

                /*
                 * Get a user
                 */
                requestArgs = new HashMap();
                requestArgs.put("format", "json");
                requestArgs.put("uid", "new-user");

                response = adminApi.execute("GET", "/admin/user", null, requestArgs);

                System.out.println(response.getStatusLine());
                entity = response.getEntity();

                try {
                        System.out.println("\nResponse Content is: "
                                + EntityUtils.toString(entity, "UTF-8") + "\n");
                        response.close();
                } catch (IOException e){
                        System.err.println ("Encountered an I/O exception.");
                        e.printStackTrace();
                }

                /*
                 * Modify a user
                 */
                requestArgs = new HashMap();
                requestArgs.put("display-name", "John Doe");
                requestArgs.put("email", "johndoe@email.com");
                requestArgs.put("format", "json");
```

```
                requestArgs.put("uid", "new-user");
                requestArgs.put("max-buckets", "100");

                response = adminApi.execute("POST", "/admin/user", null, requestArgs);

                System.out.println(response.getStatusLine());
                entity = response.getEntity();

                try {
                        System.out.println("\nResponse Content is: "
                                + EntityUtils.toString(entity, "UTF-8") + "\n");
                        response.close();
                } catch (IOException e){
                        System.err.println ("Encountered an I/O exception.");
                        e.printStackTrace();
                }


                /*
                 * Create a subuser
                 */
                requestArgs = new HashMap();
                requestArgs.put("format", "json");
                requestArgs.put("uid", "new-user");
                requestArgs.put("subuser", "foobar");

                response = adminApi.execute("PUT", "/admin/user", "subuser", requestArgs);
                System.out.println(response.getStatusLine());
                entity = response.getEntity();

                try {
                        System.out.println("\nResponse Content is: "
                                + EntityUtils.toString(entity, "UTF-8") + "\n");
                        response.close();
                } catch (IOException e){
                        System.err.println ("Encountered an I/O exception.");
                        e.printStackTrace();
                }


                /*
                 * Delete a user
                 */
                requestArgs = new HashMap();
                requestArgs.put("format", "json");
                requestArgs.put("uid", "new-user");

                response = adminApi.execute("DELETE", "/admin/user", null, requestArgs);
                System.out.println(response.getStatusLine());
                entity = response.getEntity();

                try {
                        System.out.println("\nResponse Content is: "
                                + EntityUtils.toString(entity, "UTF-8") + "\n");
                        response.close();
                } catch (IOException e){
                        System.err.println ("Encountered an I/O exception.");
                        e.printStackTrace();
                }
        }
}
```

Reference

- For a more extensive explanation of the Amazon S3 authentication procedure, consult the Signing and Authenticating REST Requests section of Amazon Simple Storage Service documentation.
- For more information, see S3 Authentication.

Creating an administrative user

Learn how to create an administrative API user.

Important: To run the **`radosgw-admin`** command from the Ceph Object Gateway node, ensure the node has the admin key. The admin key can be copied from any Ceph Monitor node.

Prerequisites

- Root-level access to the Ceph Object Gateway node.

Procedure

1. Create an object gateway user:

   Syntax

   ```
   radosgw-admin user create --uid="USER_NAME" --display-name="DISPLAY_NAME"
   ```

   Example

   ```
   [user@client ~]$ radosgw-admin user create --uid="admin-api-user" --display-
   name="Admin API User"
   ```

   The `radosgw-admin` command-line interface will return the user.

   Example output

   ```
   {
       "user_id": "admin-api-user",
       "display_name": "Admin API User",
       "email": "",
       "suspended": 0,
       "max_buckets": 1000,
       "auid": 0,
       "subusers": [],
       "keys": [
           {
               "user": "admin-api-user",
               "access_key": "NRWGT19TWMYOB1YDBV1Y",
               "secret_key": "gr1VEGIV7rxcP3xvXDFCo4UDwwl2YoNrmtRlIAty"
           }
       ],
       "swift_keys": [],
       "caps": [],
       "op_mask": "read, write, delete",
       "default_placement": "",
       "placement_tags": [],
       "bucket_quota": {
           "enabled": false,
           "max_size_kb": -1,
           "max_objects": -1
       },
       "user_quota": {
           "enabled": false,
           "max_size_kb": -1,
           "max_objects": -1
       },
       "temp_url_keys": []
   }
   ```

2. Assign administrative capabilities to the user you create:

   Syntax

   ```
   radosgw-admin caps add --uid="USER_NAME" --caps="users=*"
   ```

   Example

   ```
   [user@client ~]$ radosgw-admin caps add --uid=admin-api-user --caps="users=*"
   ```

   The `radosgw-admin` command-line interface will return the user. The `"caps":` will have the capabilities you assigned to the user:

Example output

```
{
    "user_id": "admin-api-user",
    "display_name": "Admin API User",
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "auid": 0,
    "subusers": [],
    "keys": [
        {
            "user": "admin-api-user",
            "access_key": "NRWGT19TWMYOB1YDBV1Y",
            "secret_key": "gr1VEGIV7rxcP3xvXDFCo4UDwwl2YoNrmtRlIAty"
        }
    ],
    "swift_keys": [],
    "caps": [
        {
            "type": "users",
            "perm": "*"
        }
    ],
    "op_mask": "read, write, delete",
    "default_placement": "",
    "placement_tags": [],
    "bucket_quota": {
        "enabled": false,
        "max_size_kb": -1,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "max_size_kb": -1,
        "max_objects": -1
    },
    "temp_url_keys": []
}
```

Now you have a user with administrative privileges.

Get user information

Learn how to get user information.

Cap `users` or `user-info-without-keys` must be set to `read` to run this operation. If cap `user-info-without-keys` is set to `read` or `*`, S3 keys and Swift keys will not be included in the response unless the user running this operation is the system user, an admin user, or the cap `users` is set to `read`.

Capabilities

```
users=read or user-info-without-keys=read
```

Syntax

```
GET /admin/user?format=json HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`uid`

Description

The user for which the information is requested.

Type

String

Example

`foo_user`

Required

Yes

`access-key`

Description

The S3 access key of the user for which the information is requested.

Type

String

Example

`ABCD0EF12GHIJ2K34LMN`

Required

No

Response Entities

`user`

Description

A container for the user data information.

Type

Container

Parent

N/A

`user_id`

Description

The user ID.

Type

String

Parent

`user`

`display_name`

Description

Display name for the user.

Type

String

Parent

`user`

`suspended`

Description

True if the user is suspended.

Type

Boolean

Parent

`user`

`max_buckets`

Description

The maximum number of buckets to be owned by the user.

Type

Integer

Parent

`user`

`subusers`

Description

Subusers associated with this user account.

Type

Container

Parent

`user`

`keys`

Description

S3 keys associated with this user account.

Type

Container

Parent

`user`

`swift_keys`

Description

Swift keys associated with this user account.

Type

Container

Parent

`user`

`caps`

Description

User capabilities.

Type

Container

Parent

`user`

If successful, the response contains the user information.

Special Error Responses

None.

Create a user

Create a new user. By default, an S3 key pair is automatically created and returned in the response.

If only a `access-key` or `secret-key` is provided, the omitted key will be automatically generated. By default, a generated key is added to the keyring without replacing an existing key pair. If `access-key` is specified and refers to an existing key owned by the user then it will be modified.

Capabilities

```
`users=write`
```

Syntax

```
PUT /admin/user?format=json HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`uid`

Description

The user ID to be created.

Type

String

Example

`foo_user`

Required

Yes

`display-name`

Description

The display name of the user to be created.

Type

String

Example

`foo_user`

Required

Yes

`email`

Description

The email address associated with the user.

Type

String

Example

`foo@bar.com`

Required

No

`key-type`

Description

Key type to be generated, options are: swift, s3 (default).

Type

String

Example

`s3` [`S3`]

Required

No

`access-key`

Description

Specify access key.

Type

String

Example

`ABCD0EF12GHIJ2K34LMN`

Required

No

`secret-key`

Description

Specify secret key.

Type

String

Example

`0AbCDEFg1h2i34JklM5nop6QrSTUV+WxyzaBC7D8`

Required

No

`user-caps`

Description

User capabilities.

Type

String

Example

`usage=read, write; users=read`

Required

No

`generate-key`

Description

Generate a new key pair and add to the existing keyring.

Type

Boolean

Example

True [`True`]

Required

No

`max-buckets`

Description

Specify the maximum number of buckets the user can own.

Type

Integer

Example

500 [1000]

Required

No

`suspended`

Description

Specify whether the user should be suspended

Type

Boolean

Example

False [False]

Required

No

Response Entities

`user`

Description

Specify whether the user should be suspended

Type

Boolean

Parent

No

`user_id`

Description

The user ID.

Type

String

Parent

`user`

`display_name`

Description

Display name for the user.

Type

String

Parent

`user`

`suspended`

Description

True if the user is suspended.

Type

Boolean

Parent

`user`

`max_buckets`

Description

The maximum number of buckets to be owned by the user.

Type

Integer

Parent

`user`

`subusers`

Description

Subusers associated with this user account.

Type

Container

Parent

`user`

`keys`

Description

S3 keys associated with this user account.

Type

Container

Parent

`user`

`swift_keys`

Description

Swift keys associated with this user account.

Type

Container

Parent

`user`

`caps`

Description

User capabilities.

Type

Container

Parent

If successful, the response contains the user information.

Special Error Responses

`UserExists`

Description

Attempt to create existing user.

Code

409 Conflict

`InvalidAccessKey`

Description

Invalid access key specified.

Code

400 Bad Request

`InvalidKeyType`

Description

Invalid key type specified.

Code

400 Bad Request

`InvalidSecretKey`

Description

Invalid secret key specified.

Code

400 Bad Request

`KeyExists`

Description

Provided access key exists and belongs to another user.

Code

409 Conflict

`EmailExists`

Description

Provided email address exists.

Code

409 Conflict

`InvalidCap`

Description

Attempt to grant invalid admin capability.

Code

400 Bad Request

Reference

For more information about creating subusers, see “Create a subuser” on page 44.

Modify a user

Modify an existing user.

Capabilities

```
`users=write`
```

Syntax

```
POST /admin/user?format=json HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`uid`

Description

The user ID to be created.

Type

String

Example

`foo_user`

Required

Yes

`display-name`

Description

The display name of the user to be created.

Type

String

Example

`foo_user`

Required

Yes

`email`

Description

The email address associated with the user.

Type

String

Example

`foo@bar.com`

Required

No

`generate-key`

Description

Generate a new key pair and add to the existing keyring.

Type

Boolean

Example

True [False]

Required

No

`access-key`

Description

Specify access key.

Type

String

Example

`ABCD0EF12GHIJ2K34LMN`

Required

No

`secret-key`

Description

Specify secret key.

Type

String

Example

`0AbCDEFg1h2i34JklM5nop6QrSTUV+WxyzaBC7D8`

Required

No

`key-type`

Description

Key type to be generated, options are: swift, s3 (default).

Type

String

Example

`s3`

Required

No

`user-caps`

Description

User capabilities.

Type

String

Example

`usage=read, write; users=read`

Required

No

`max-buckets`

Description

Specify the maximum number of buckets the user can own.

Type

Integer

Example

500 [1000]

Required

No

`suspended`

Description

Specify whether the user should be suspended

Type

Boolean

Example

False [False]

Required

No

Response Entities

`user`

Description

Specify whether the user should be suspended

Type

Boolean

Parent

No

`user_id`

Description

The user ID.

Type

String

Parent

`user`

`display_name`

Description

Display name for the user.

Type

String

Parent

`user`

`suspended`

Description

True if the user is suspended.

Type

Boolean

Parent

user

`max_buckets`

Description

The maximum number of buckets to be owned by the user.

Type

Integer

Parent

user

`subusers`

Description

Subusers associated with this user account.

Type

Container

Parent

user

`keys`

Description

S3 keys associated with this user account.

Type

Container

Parent

user

`swift_keys`

Description

Swift keys associated with this user account.

Type

Container

Parent

user

`caps`

Description

User capabilities.

Type

Container

Parent

If successful, the response contains the user information.

Special Error Responses

`InvalidAccessKey`

Description

Invalid access key specified.

Code

400 Bad Request

`InvalidKeyType`

Description

Invalid key type specified.

Code

400 Bad Request

`InvalidSecretKey`

Description

Invalid secret key specified.

Code

400 Bad Request

`KeyExists`

Description

Provided access key exists and belongs to another user.

Code

409 Conflict

`EmailExists`

Description

Provided email address exists.

Code

409 Conflict

`InvalidCap`

Description

Attempt to grant invalid admin capability.

Code

400 Bad Request

Reference

For more information, see “Modify a subuser” on page 47.

Remove a user

Remove an existing user.

Capabilities

```
`users=write`
```

Syntax

```
DELETE /admin/user?format=json HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`uid`

Description

The user ID to be removed.

Type

String

Example

`foo_user`

Required

Yes

`purge-data`

Description

When specified the buckets and objects belonging to the user will also be removed.

Type

Boolean

Example

True

Required

No

Response Entities

None.

Special Error Responses

None.

Reference

For more information, see Remove a subuser.

Create a subuser

Learn how to create a new subuser, primarily useful for clients using the Swift API.

> **Note:** Either `gen-subuser` or `subuser` is required for a valid request. In general, for a subuser to be useful, it must be granted permissions by specifying `access`. As with user creation if `subuser` is specified without `secret`, then a secret key is automatically generated.

Capabilities

```
`users=write`
```

Syntax

```
PUT /admin/user?subuser&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`uid`

Description

The user ID under which a subuser is to be created.

Type

String

Example

`foo_user`

Required

Yes

`subuser`

Description

Specify the subuser ID to be created.

Type

String

Example

`sub_foo`

Required

Yes (or `gen-subuser`)

`gen-subuser`

Description

Specify the subuser ID to be created.

Type

String

Example

`sub_foo`

Required

Yes (or `gen-subuser`)

`secret-key`

Description

Specify secret key.

Type

String

Example

`0AbCDEFg1h2i34JklM5nop6QrSTUV+WxyzaBC7D8`

Required

No

`key-type`

Description

Key type to be generated, options are: swift (default), s3.

Type

String

Example

`swift` [`swift`]

Required

No

`access`

Description

Set access permissions for sub-user, should be one of `read, write, readwrite, full`.

Type

String

Example

`read`

Required

No

`generate-secret`

Description

Generate the secret key.

Type

Boolean

Example

True [False]

Required

No

Response Entities

`subusers`

Description

Subusers associated with the user account.

Type

Container

Parent

N/A

`permissions`

Description

Subuser access to user account.

Type

String

Parent

`subusers`

If successful, the response contains the subuser information.

`SubuserExists`

Description

Specified subuser exists.

Code

409 Conflict

`InvalidKeyType`

Description

Invalid key type specified.

Code

400 Bad Request

`InvalidSecretKey`

Description

Invalid secret key specified.

Code

400 Bad Request

`InvalidAccess`

Description

Invalid subuser access specified

Code

400 Bad Request

Modify a subuser

Learn how to modify an existing subuser.

Capabilities

```
`users=write`
```

Syntax

```
POST /admin/user?subuser&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`uid`

Description

The user ID under which a subuser is to be created.

Type

String

Example

`foo_user`

Required

Yes

`subuser`

Description

The subuser ID to be modified.

Type

String

Example

`sub_foo`

Required

No

`generate-secret`

Description

Generate a new secret key for the subuser, replacing the existing key.

Type

Boolean

Example

True [False]

Required

No

`secret`

Description

Specify secret key.

Type

String

Example

`0AbCDEFg1h2i34JklM5nop6QrSTUV+WxyzaBC7D8`

Required

No

`key-type`

Description

Key type to be generated, options are: swift (default), s3.

Type

String

Example

`swift` [`swift`]

Required

No

`access`

Description

Set access permissions for sub-user, should be one of `read, write, readwrite, full`.

Type

String

Example

`read`

Required

No

Response Entities

`subusers`

Description

Subusers associated with the user account.

Type

Container

Parent

N/A

`id`

Description

Subuser ID

Type

String

Parent

`subusers`

`permissions`

Description

Subuser access to user account.

Type

String

Parent

`subusers`

If successful, the response contains the subuser information.

`InvalidKeyType`

Description

Invalid key type specified.

Code

400 Bad Request

`InvalidSecretKey`

Description

Invalid secret key specified.

Code

400 Bad Request

`InvalidAccess`

Description

Invalid subuser access specified

Code

400 Bad Request

Remove a subuser

Learn how to remove a subuser.

Capabilities

```
`users=write`
```

Syntax

```
DELETE /admin/user?subuser&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`uid`

Description

The user ID to be removed.

Type

String

Example

`foo_user`

Required

Yes

`subuser`

Description

The subuser ID to be removed.

Type

String

Example

`sub_foo`

Required

Yes

`purge-keys`

Description

Remove keys belonging to the subuser.

Type

Boolean

Example

True [True]

Required

No

Response Entities

None.

Special Error Responses

None.

Add capabilities to a user

Learn how to add an administrative capability to a specified user.

Capabilities

```
`users=write`
```

Syntax

```
PUT /admin/user?caps&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

`uid`

Description

The user ID to add an administrative capability to.

Type

String

Example

`foo_user`

Required

Yes

`user-caps`

Description

The administrative capability to add to the user.

Type

String

Example

`usage=read, write`

Required

Yes

`user`

Description

A container for the user data information.

Type

Container

Parent

N/A

`user_id`

Description

The user ID

Type

String

Parent

`user`

`caps`

Description

User capabilities

Type

Container

Parent

`user`

If successful, the response contains the user's capabilities.

Special Error Responses

`InvalidCap`

Description

Attempt to grant invalid admin capability.

Code

400 Bad Request

Remove capabilities from a user

Learn how to remove an administrative capability from a specified user

Capabilities

```
`users=write`
```

Syntax

```
DELETE /admin/user?caps&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`uid`

Description

The user ID to remove an administrative capability from.

Type

String

Example

`foo_user`

Required

Yes

`user-caps`

Description

The administrative capabilities to remove from the user.

Type

String

Example

`usage=read, write`

Required

Yes

Response Entities

`user`

Description

A container for the user data information.

Type

Container

Parent

N/A

`user_id`

Description

The user ID.

Type

String

Parent

`user`

`caps`

Description

User capabilities.

Type

Container

Parent

`user`

If successful, the response contains the user's capabilities.

`InvalidCap`

Description

Attempt to remove an invalid admin capability.

Code

400 Bad Request

`NoSuchCap`

Description

User does not possess specified capability.

Code

404 Not Found

Create a key

Learn how to create a new key.

If a `subuser` is specified then by default created keys will be swift type. If only one of `access-key` or `secret-key` is provided the committed key will be automatically generated, that is if only `secret-key` is specified then `access-key` will be automatically generated. By default, a generated key is added to the keyring without replacing an existing key pair. If `access-key` is specified and refers to an existing key owned by the user then it will be modified. The response is a container listing all keys of the same type as the key created.

> **Note:** When creating a swift key, specifying the option `access-key` will have no effect. Additionally, only one swift key might be held by each user or subuser.

Capabilities

```
`users=write`
```

Syntax

```
PUT /admin/user?key&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`uid`

Description

The user ID to receive the new key.

Type

String

Example

`foo_user`

Required

Yes

`subuser`

Description

The subuser ID to receive the new key.

Type

String

Example

`sub_foo`

Required

No

`key-type`

Description

Key type to be generated, options are: swift, s3 (default).

Type

String

Example

`s3` [`s3`]

Required

No

`access-key`

Description

Specify access key.

Type

String

Example

`AB01C2D3EF45G6H7IJ8K`

Required

No

`secret-key`

Description

Specify secret key.

Type

String

Example

`0ab/CdeFGhij1klmnopqRSTUv1WxyZabcDEFgHij`

Required

No

`generate-key`

Description

Generate a new key pair and add to the existing keyring.

Type

Boolean

Example

True [`True`]

Required

No

Response Entities

`keys`

Description

Keys of type created associated with this user account.

Type

Container

Parent

N/A

`user`

Description

The user account associated with the key.

Type

String

Parent

`keys`

`access-key`

Description

The access key.

Type

String

Parent

`keys`

`secret-key`

Description

The secret key.

Type

String

Parent

`keys`

Special Error Responses

`InvalidAccessKey`

Description

Invalid access key specified.

Code

400 Bad Request

`InvalidKeyType`

Description

Invalid key type specified.

Code

400 Bad Request

`InvalidSecretKey`

Description

Invalid secret key specified.

Code

400 Bad Request

`InvalidKeyType`

Description

Invalid key type specified.

Code

400 Bad Request

`KeyExists`

Description

Provided access key exists and belongs to another user.

Code

409 Conflict

Remove a key

Learn how to remove a key.

Capabilities

```
`users=write`
```

Syntax

```
DELETE /admin/user?key&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`access-key`

Description

The S3 access key belonging to the S3 key pair to remove.

Type

String

Example

`AB01C2D3EF45G6H7IJ8K`

Required

Yes

`uid`

Description

The user to remove the key from.

Type

String

Example

`foo_user`

Required

No

`subuser`

Description

The subuser to remove the key from.

Type

String

Example

`sub_foo`

Required

No

`key-type`

Description

Key type to be removed, options are: swift, s3.

> **Note:** Required to remove swift key.

Type

String

Example

`swift`

Required

No

Special Error Responses

None.

Response Entities

None.

Bucket notifications

Bucket notification APIs can be used to provide configuration and control interfaces for the bucket notification mechanism.

Bucket notifications associate topics with a specific bucket. For more information about bucket notifications, see "S3 bucket operations" on page 102. Before using bucket notification APIs, create bucket notifications on the Ceph Object Gateway.

Note:

- In all topic actions, the parameters are URL encoded, and sent in the message body using `application/x-www-form-urlencoded` content type.
- Any bucket notification already associated with the topic needs to be re-created for the topic update to take effect.

Bucket notifications provide a way to send information out of the Ceph Object Gateway when certain events happen in the bucket. Bucket notifications can be sent to HTTP, AMQP0.9.1, and Kafka endpoints. A notification entry must be created to send bucket notifications for events on a specific bucket and to a specific topic. A bucket notification can be created on a subset of event types or by default for all event types. The bucket notification can filter out events based on key prefix or suffix, regular expression matching the keys, and the metadata attributes attached to the object, or the object tags. Bucket notifications have a REST API to provide configuration and control interfaces for the bucket notification mechanism.

Sending a bucket notification when an object is synced to a zone lets the external system get information into the zone syncing status at the object level. The bucket notification event types `s3:ObjectSynced:*` and `s3:ObjectSynced:Created`, when configured via the bucket notification mechanism, send a notification event from the synced RGW upon successful sync of an object. Both the topics and the notification configuration should be done separately in each zone from which the notification events are being sent.

Persistent notifications

Persistent notifications enable reliable and asynchronous delivery of notifications from the Ceph Object Gateway to the endpoint configured at the topic.

Regular notifications are also reliable because the delivery to the endpoint is performed synchronously during the request. With persistent notifications, the Ceph Object Gateway retries sending notifications even when the endpoint is down or there are network issues during the operations, that is notifications are retried if not successfully delivered to the endpoint. Notifications are sent only after all other actions related to the notified operation are successful. If an endpoint goes down for a longer duration, the notification queue fills up and the S3 operations that have configured notifications for these endpoints will fail.

Note: With `kafka-ack-level=none`, there is no indication for message failures, and therefore messages sent while broker is down are not retried, when the broker is up again. After the broker is up again, only new notifications are seen.

Creating a topic

Create a topic before creating bucket notifications.

A topic is a Simple Notification Service (SNS) entity and all the topic operations, that is, `create`, `delete`, `list`, and `get`, are SNS operations. The topic needs to have endpoint parameters that are used when a bucket notification is created. Once the request is successful, the response includes the topic Amazon Resource Name (ARN) that can be used later to reference this topic in the bucket notification request.

Note: A `topic_arn` provides the bucket notification configuration and is generated after a topic is created.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access.
- Installation of the Ceph Object Gateway.
- User access key and secret key.
- Endpoint parameters.

1. Create a topic with the following request format:

   Syntax

   ```
   POST
   Action=CreateTopic
   &Name=TOPIC_NAME
   [&Attributes.entry.1.key=amqp-exchange&Attributes.entry.1.value=EXCHANGE]
   [&Attributes.entry.2.key=amqp-ack-level&Attributes.entry.2.value=none|broker|routable]
   [&Attributes.entry.3.key=verify-ssl&Attributes.entry.3.value=true|false]
   [&Attributes.entry.4.key=kafka-ack-level&Attributes.entry.4.value=none|broker]
   [&Attributes.entry.5.key=use-ssl&Attributes.entry.5.value=true|false]
   [&Attributes.entry.6.key=ca-location&Attributes.entry.6.value=FILE_PATH]
   [&Attributes.entry.7.key=OpaqueData&Attributes.entry.7.value=OPAQUE_DATA]
   [&Attributes.entry.8.key=push-endpoint&Attributes.entry.8.value=ENDPOINT]
   [&Attributes.entry.9.key=persistent&Attributes.entry.9.value=true|false]
   ```

   Here are the request parameters:

   - `Endpoint`: URL of an endpoint to send notifications to.
   - `OpaqueData`: opaque data is set in the topic configuration and added to all notifications triggered by the topic.
   - `persistent`: indication of whether notifications to this endpoint are persistent that is asynchronous or not. By default the value is `false`.
   - HTTP endpoint:
     - `URL`: https://*FQDN:PORT*
     - `port defaults to`: Use 80/443 for HTTP[S] accordingly.
     - `verify-ssl`: Indicates whether the server certificate is validated by the client or not. By default , it is `true`.
   - AMQP0.9.1 endpoint:
     - `URL`: amqp://*USER:PASSWORD@FQDN:PORT[/VHOST]*.
     - User and password defaults to: `guest` and `guest` respectively.
     - User and password details should be provided over HTTPS, otherwise the topic creation request is rejected.
     - `port defaults to`: 5672.
     - `vhost` defaults to: “/”
     - `amqp-exchange`: The exchanges must exist and be able to route messages based on topics. This is a mandatory parameter for AMQP0.9.1. Different topics pointing to the same endpoint must use the same exchange.
     - `amqp-ack-level`: No end to end acknowledgment is required, as messages may persist in the broker before being delivered into their final destination. Three acknowledgment methods exist:
       - `none`: Message is considered `delivered` if sent to the broker.

- `broker`: By default, the message is considered `delivered` if acknowledged by the broker.
- `routable`: Message is considered `delivered` if the broker can route to a consumer.

> **Note:**
> - The key and value of a specific parameter do not have to reside in the same line, or in any specific order, but must use the same index. Attribute indexing does not need to be sequential or start from any specific value.
> - The `topic-name` is used for the AMQP topic.

– Kafka endpoint:
  - `URL`: kafka:*USER:PASSWORD@FQDN:PORT*.
  - `use-ssl` is set to `false` by default. If `use-ssl` is set to `true`, secure connection is used for connecting with the broker.
  - If `ca-location` is provided, and secure connection is used, the specified CA will be used, instead of the default one, to authenticate the broker.
  - User and password can only be provided over HTTP[S]. Otherwise, the topic creation request is rejected.
  - User and password may only be provided together with `use-ssl`, otherwise, the connection to the broker will fail.
  - `port defaults to`: 9092.
  - `kafka-ack-level`: no end to end acknowledgment required, as messages may persist in the broker before being delivered into their final destination. Two acknowledgment methods exist:
    - `none`: message is considered `delivered` if sent to the broker.
    - `broker`: By default, the message is considered `delivered` if acknowledged by the broker.

The following is an example of the response format:

Example

```
<CreateTopicResponse xmlns="https://sns.amazonaws.com/doc/2010-03-31/">
    <CreateTopicResult>
        <TopicArn></TopicArn>
    </CreateTopicResult>
    <ResponseMetadata>
        <RequestId></RequestId>
    </ResponseMetadata>
</CreateTopicResponse>
```

> **Note:** The topic Amazon Resource Name (ARN) in the response will have the following format:
> `arn:aws:sns:ZONE_GROUP:TENANT:TOPIC`

The following is an example of AMQP0.9.1 endpoint:

Example

```
client.create_topic(Name='my-topic' , Attributes={'push-endpoint': 'amqp://127.0.0.1:5672',
'amqp-exchange': 'ex1', 'amqp-ack-level': 'broker'}) "
```

Getting topic information

Getting topic information provides information about a specific topic, which can include endpoint information if it is provided.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access.
- Installation of the Ceph Object Gateway.
- User access key and secret key.
- Endpoint parameters.

Procedure

1. Get topic information with the following request format:

   Syntax

   ```
   POST
   Action=GetTopic
   &TopicArn=TOPIC_ARN
   ```

   Here is an example of the response format:

   ```
   <GetTopicResponse>
   <GetTopicRersult>
   <Topic>
   <User></User>
   <Name></Name>
   <EndPoint>
   <EndpointAddress></EndpointAddress>
   <EndpointArgs></EndpointArgs>
   <EndpointTopic></EndpointTopic>
   <HasStoredSecret></HasStoredSecret>
   <Persistent></Persistent>
   </EndPoint>
   <TopicArn></TopicArn>
   <OpaqueData></OpaqueData>
   </Topic>
   </GetTopicResult>
   <ResponseMetadata>
   <RequestId></RequestId>
   </ResponseMetadata>
   </GetTopicResponse>
   ```

   The following are the tags and definitions:

   - `User`: Name of the user that created the topic.
   - `Name`: Name of the topic.
   - JSON formatted endpoints include:
     - `EndpointAddress`: The endpoint URL. If the endpoint URL contains user and password information, the request must be made over HTTPS. Otheriwse, the topic get request is rejected.
       - `EndPointArgs`: The endpoint arguments.
       - `EndpointTopic`: The topic name that is be sent to the endpoint can be different than the example topic name.
       - `HasStoredSecret`: `true` when the endpoint URL contains user and password information.
       - `Persistent`: `true` when the topic is persistent.
   - `TopicArn`: Topic ARN.

- `OpaqueData`: This is an opaque data set on the topic.

Listing topics

Learn how to list topics that a user has defined.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access.
- Installation of the Ceph Object Gateway.
- User access key and secret key.
- Endpoint parameters.

Procedure

1. List topic information with the following request format:

Syntax

```
POST
Action=ListTopics
```

Here is an example of the response format:

```
<ListTopicdResponse xmlns="https://sns.amazonaws.com/doc/2020-03-31/">
<ListTopicsRersult>
<Topics>
<member>
<User></User>
<Name></Name>
<EndPoint>
<EndpointAddress></EndpointAddress>
<EndpointArgs></EndpointArgs>
<EndpointTopic></EndpointTopic>
</EndPoint>
<TopicArn></TopicArn>
<OpaqueData></OpaqueData>
</member>
</Topics>
</ListTopicsResult>
<ResponseMetadata>
<RequestId></RequestId>
</ResponseMetadata>
</ListTopicsResponse>
```

Note: If endpoint URL contains user and password information, in any of the topics, the request must be made over HTTPS. Otherwise, the topic list request is rejected.

Deleting topics

Learn how to delete a topic.

Note: Removing a deleted topic results in no operation and is not a failure

.

Prerequisites

- A running Red Hat Ceph Storage cluster.

- Root-level access.
- Installation of the Ceph Object Gateway.
- User access key and secret key.
- Endpoint parameters.

Procedure

- Delete a topic with the following request format:

  Syntax

  ```
  POST
  Action=DeleteTopic
  &TopicArn=TOPIC_ARN
  ```

  Here is an example of the response format:

  ```
  <DeleteTopicResponse xmlns="https://sns.amazonaws.com/doc/2020-03-31/">
  <ResponseMetadata>
  <RequestId></RequestId>
  </ResponseMetadata>
  </DeleteTopicResponse>
  ```

Using the command-line interface for topic management

List, get, and remove topics by using the command-line interface.

Prerequisites

- Root-level access to the Ceph Object Gateway node.

1. To get a list of all topics of a user:

   Syntax

   ```
   radosgw-admin topic list --uid=USER_ID
   ```

   Example

   ```
   [root@rgw ~]# radosgw-admin topic list --uid=example
   ```

2. To get configuration of a specific topic:

   Syntax

   ```
   radosgw-admin topic get --uid=USER_ID --topic=TOPIC_NAME
   ```

   Example

   ```
   [root@rgw ~]# radosgw-admin topic get --uid=example --topic=example-topic
   ```

3. To remove a specific topic:

   Syntax

   ```
   radosgw-admin topic rm --uid=USER_ID --topic=TOPIC_NAME
   ```

   Example

   ```
   [root@rgw ~]# radosgw-admin topic rm --uid=example --topic=example-topic
   ```

Managing notification configuration

Manage notification configuration of buckets with the command-line interface.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A Red Hat Ceph Storage cluster.
- A Ceph Object Gateway installed.

You can list, get, and remove a bucket notification configuration with the **radosgw-admin** command.

- List all the bucket notification configuration.

```
radosgw-admin notification list --bucket=BUCKET_NAME
```

  For example,

```
[root@host04 ~]# radosgw-admin notification list --bucket bkt2
{
    "notifications": [
        {
            "TopicArn": "arn:aws:sns:default::topic1",
            "Id": "notif1",
            "Events": [
                "s3:ObjectCreated:*",
                "s3:ObjectRemoved:*"
            ],
            "Filter": {
                "S3Key": {},
                "S3Metadata": {},
                "S3Tags": {}
            }
        },
        {
            "TopicArn": "arn:aws:sns:default::topic1",
            "Id": "notif2",
            "Events": [
                "s3:ObjectSynced:*"
            ],
            "Filter": {
                "S3Key": {},
                "S3Metadata": {},
                "S3Tags": {}
            }
        }
    ]
}
```

- Get the bucket notification configuration.

```
radosgw-admin notification get --bucket BUCKET_NAME --notification-id NOTIFICATION_ID
```

  For example,

```
[root@host04 ~]# radosgw-admin notification get --bucket bkt2 --notification-id notif2
{
    "TopicArn": "arn:aws:sns:default::topic1",
    "Id": "notif2",
    "Events": [
        "s3:ObjectSynced:*"
    ],
    "Filter": {
        "S3Key": {},
        "S3Metadata": {},
        "S3Tags": {}
    }
}
```

- Remove a specific bucket notification configuration.

```
radosgw-admin notification rm --bucket BUCKET_NAME [--notification-id NOTIFICATION_ID]
```

Here, *NOTIFICATION_ID* is optional. If it is not specified, the command removes all the notification configurations of that bucket.
For example,

```
[root@host04 ~]# radosgw-admin notification rm --bucket bkt2 --notification-id notif1
```

Event record

An event holds information about the operation done by the Ceph Object Gateway and is sent as a payload over the chosen endpoint, such as HTTP, HTTPS, Kafka, or AMQ0.9.1. The event record is in JSON format.

The following `ObjectLifecycle:Expiration` events are supported:

- `ObjectLifecycle:Expiration:Current`
- `ObjectLifecycle:Expiration:NonCurrent`
- `ObjectLifecycle:Expiration:DeleteMarker`
- `ObjectLifecycle:Expiration:AbortMultipartUpload`

Example

```
{"Records":[
    {
        "eventVersion":"2.1",
        "eventSource":"ceph:s3",
        "awsRegion":"us-east-1",
        "eventTime":"2019-11-22T13:47:35.124724Z",
        "eventName":"ObjectCreated:Put",
        "userIdentity":{
            "principalId":"tester"
        },
        "requestParameters":{
            "sourceIPAddress":""
        },
        "responseElements":{
            "x-amz-request-id":"503a4c37-85eb-47cd-8681-2817e80b4281.5330.903595",
            "x-amz-id-2":"14d2-zone1-zonegroup1"
        },
        "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"mynotif1",
            "bucket":{
                "name":"mybucket1",
                "ownerIdentity":{
                    "principalId":"tester"
                },
                "arn":"arn:aws:s3:us-east-1::mybucket1",
                "id":"503a4c37-85eb-47cd-8681-2817e80b4281.5332.38"
            },
            "object":{
                "key":"myimage1.jpg",
                "size":"1024",
                "eTag":"37b51d194a7513e45b56f6524f2d51f2",
                "versionId":"",
                "sequencer": "F7E6D75DC742D108",
                "metadata":[],
                "tags":[]
            }
        },
        "eventId":"",
        "opaqueData":"me@example.com"
    }
]}
```

These are the event record keys and their definitions:

awsRegion

Zonegroup.

eventTime

Timestamp that indicates when the event was triggered.

eventName

The type of the event. It can be `ObjectCreated`, `ObjectRemovedObjectLifecycle:Expiration`

`userIdentity.principalId`

The identity of the user that triggered the event.

`requestParameters.sourceIPAddress`

The IP address of the client that triggered the event. This field is not supported.

`responseElements.x-amz-request-id`

The request ID that triggered the event.

`responseElements.x_amzID2`

The identity of the Ceph Object Gateway on which the event was triggered. The identity format is RGWID*-ZONE-ZONEGROUP*.

`s3.configurationId`

The notification ID that created the event.

`s3.bucket.name`

The name of the bucket.

`s3.bucket.ownerIdentity.principalId`

The owner of the bucket.

`s3.bucket.arn`

Amazon Resource Name (ARN) of the bucket.

`s3.bucket.id`

Identity of the bucket.

`s3.object.key`

The object key.

`s3.object.size`

The size of the object.

`s3.object.eTag`

The object etag.

`s3.object.version`

The object version in a versioned bucket.

`s3.object.sequencer`

Monotonically increasing identifier of the change per object in the hexadecimal format.

`s3.object.metadata`

Any metadata set on the object sent as `x-amz-meta`.

`s3.object.tags`

Any tags set on the object.

`s3.eventId`

Unique identity of the event.

`s3.opaqueData`

Opaque data is set in the topic configuration and added to all notifications triggered by the topic.

Reference

- See the Event Message Structure for more information.

Supported event types

Get to know the supported event types.

The following event types are supported:

- `s3:ObjectCreated:*`
- `s3:ObjectCreated:Put`
- `s3:ObjectCreated:Post`
- `s3:ObjectCreated:Copy`
- `s3:ObjectCreated:CompleteMultipartUpload`

  > **Note:** In multipart upload, an ObjectCreated:CompleteMultipartUpload notification is sent at the end of the process.

- `s3:ObjectRemoved:*`
- `s3:ObjectRemoved:Delete`
- `s3:ObjectRemoved:DeleteMarkerCreated`
- `s3:ObjectLifecycle:Expiration:Current`
- `s3:ObjectLifecycle:Expiration:NonCurrent`
- `s3:ObjectLifecycle:Expiration:DeleteMarker`
- `s3:ObjectLifecycle:Expiration:AbortMultipartUpload`
- `s3:ObjectLifecycle:Transition:Current`
- `s3:ObjectLifecycle:Transition:NonCurrent`
- `s3:ObjectSynced:Create`

Get bucket information

Get information about a subset of the existing buckets.

If `uid` is specified without `bucket` then all buckets belonging to the user will be returned. If `bucket` alone is specified, information for that particular bucket will be retrieved.

Capabilities

```
`buckets=read`
```

Syntax

```
GET /admin/bucket?format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`bucket`

Description

The bucket to return info on.

Type

String

Example

`foo_bucket`

Required

No

`uid`

Description

The user to retrieve bucket information for.

Type

String

Example

`foo_user`

Required

No

`stats`

Description

Return bucket statistics.

Type

Boolean

Example

True [False]

Required

No

Response Entities

`stats`

Description

Per bucket information.

Type

Container

Parent

N/A

`buckets`

Description

Contains a list of one or more bucket containers.

Type

Container

Parent

`buckets`

`bucket`

Description

Container for single bucket information.

Type

Container

Parent

buckets

`name`

Description

The name of the bucket.

Type

String

Parent

`bucket`

`pool`

Description

The pool the bucket is stored in.

Type

String

Parent

`bucket`

`id`

Description

The unique bucket ID.

Type

String

Parent

`bucket`

`marker`

Description

Internal bucket tag.

Type

String

Parent

`bucket`

`owner`

Description

The user ID of the bucket owner.

Type

String

Parent

`bucket`

`usage`

Description

Storage usage information.

Type

Container

Parent

bucket

index

Description

Status of bucket index.

Type

String

Parent

bucket

If successful, then the request returns a bucket's container with the bucket information.

Special Error Responses

IndexRepairFailed

Description

Bucket index repair failed.

Code

409 Conflict

Check a bucket index

Check the index of an existing bucket.

> **Note:** To check multipart object accounting with `check-objects`, `fix` must be set to True.

Capabilities

buckets=write

Syntax

```
GET /admin/bucket?index&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

bucket

Description

The bucket to return info on.

Type

String

Example

foo_bucket

Required

Yes

check-objects

Description

Check multipart object accounting.

Type

Boolean

Example

True [False]

Required

No

`fix`

Description

Also fix the bucket index when checking.

Type

Boolean

Example

False [False]

Required

No

Response Entities

`index`

Description

Status of bucket index.

Type

String

Special Error Responses

`IndexRepairFailed`

Description

Bucket index repair failed.

Code

409 Conflict

Remove a bucket

Learn how to remove an existing bucket.

Capabilities

```
`buckets=write`
```

Syntax

```
DELETE /admin/bucket?format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`bucket`

Description

The bucket to remove.

Type

String

Example

foo_bucket

Required

Yes

`purge-objects`

Description

Type

Boolean

Example

True [False]

Required

No

Response Entities

None.

Special Error Responses

`BucketNotEmpty`

Description

Attempted to delete non-empty bucket.

Code

409 Conflict

`ObjectRemovalFailed`

Description

Unable to remove objects.

Code

409 Conflict

Link a bucket

Link a bucket to a specified user, unlinking the bucket from any previous user.

Capabilities

```
`buckets=write`
```

Syntax

```
PUT /admin/bucket?format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`bucket`

Description

The bucket to unlink.

Type

String

Example

`foo_bucket`

Required

Yes

`uid`

Description

The user ID to link the bucket to.

Type

String

Example

`foo_user`

Required

Yes

Response Entities

`bucket`

Description

Container for single bucket information.

Type

Container

Parent

N/A

`name`

Description

The name of the bucket.

Type

String

Parent

`bucket`

`pool`

Description

The pool the bucket is stored in.

Type

String

Parent

`bucket`

`id`

Description

The unique bucket ID.

Type

String

Parent

`bucket`

`marker`

Description

Internal bucket tag.

Type

String

Parent

`bucket`

`owner`

Description

The user ID of the bucket owner.

Type

String

Parent

`bucket`

`usage`

Description

Storage usage information.

Type

Container

Parent

`bucket`

`index`

Description

Status of bucket index.

Type

String

Parent

`bucket`

Special Error Responses

`BucketUnlinkFailed`

Description

Unable to unlink bucket from specified user.

Code

409 Conflict

`BucketLinkFailed`

Description

Unable to link bucket to specified user.

Code

409 Conflict

Unlink a bucket

Unlink a bucket from a specified user. Unlinking is primarily useful for changing bucket ownership.

Capabilities

```
`buckets=write`
```

Syntax

```
POST /admin/bucket?format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

`bucket`

Description

The bucket to unlink.

Type

String

Example

`foo_bucket`

Required

Yes

`uid`

Description

The user ID to link the bucket to.

Type

String

Example

`foo_user`

Required

Yes

Response Entities None.

`BucketUnlinkFailed`

Description

Unable to unlink bucket from specified user.

Type

409 Conflict

Get a bucket or object policy

Get the policy of a bucket or object.

Capabilities

```
`buckets=read`
```

Syntax

```
GET /admin/bucket?policy&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`bucket`

Description

The bucket to read the policy from.

Type

String

Example

`foo_bucket`

Required

Yes

`object`

Description

The object to read the policy from.

Type

String

Example

`foo.txt`

Required

No

Response Entities

`policy`

Description

Access control policy.

Type

Container

Parent

N/A

If successful, returns the object or bucket policy.

Special Error Responses

`IncompleteBody`

Description

Either bucket was not specified for a bucket policy request or bucket and object were not specified for an object policy request.

Code

400 Bad Request

Remove an object

Learn how to remove an existing object.

> **Note:** Removing an object does not require owner to be non-suspended.

Capabilities

```
`buckets=write`
```

Syntax

```
DELETE /admin/bucket?object&format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`bucket`

Description

The bucket containing the object to be removed.

Type

String

Example

`foo_bucket`

Required

Yes

`object`

Description

The object to remove

Type

String

Example

`foo.txt`

Required

Yes

Response Entities

None.

Special Error Responses

`NoSuchObject`

Description

Specified object does not exist.

Code

404 Not Found

`ObjectRemovalFailed`

Description

Unable to remove objects.

Code

409 Conflict

Quotas

The administrative Operations API enables you to set quotas on users and on buckets owned by users. Quotas include the maximum number of objects in a bucket and the maximum storage size in megabytes.

To view quotas, the user must have a `users=read` capability. To set, modify or disable a quota, the user must have `users=write` capability.

Valid parameters for quotas include:

Bucket

The `bucket` option allows you to specify a quota for buckets owned by a user.

Maximum Objects

The `max-objects` setting allows you to specify the maximum number of objects. A negative value disables this setting.

Maximum Size

The `max-size` option allows you to specify a quota for the maximum number of bytes. A negative value disables this setting.

Quota Scope

The `quota-scope` option sets the scope for the quota. The options are `bucket` and `user`.

Get a user quota

To get a quota, the user must have `users` capability set with `read` permission.

Syntax

```
GET /admin/user?quota&uid=UID&quota-type=user
```

Set a user quota

To set a quota, the user must have `users` capability set with `write` permission.

Syntax

```
PUT /admin/user?quota&uid=UID&quota-type=user
```

The content must include a JSON representation of the quota settings as encoded in the corresponding read operation.

Get a bucket quota

Get information about a subset of the existing buckets.

If `uid` is specified without `bucket` then all buckets belonging to the user will be returned. If `bucket` alone is specified, information for that particular bucket will be retrieved.

Capabilities

```
`buckets=read`
```

Syntax

```
GET /admin/bucket?format=json HTTP/1.1
Host FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`bucket`

Description

The bucket to return info on.

Type

String

Example

`foo_bucket`

Required

No

`uid`

Description

The user to retrieve bucket information for.

Type

String

Example

`foo_user`

Required

No

`stats`

Description

Return bucket statistics.

Type

Boolean

Example

True [False]

Required

No

Response Entities

`stats`

Description

Per bucket information.

Type

Container

Parent

N/A

`buckets`

Description

Contains a list of one or more bucket containers.

Type

Container

Parent

N/A

`bucket`

Description

Container for single bucket information.

Type

Container

Parent

`buckets`

`name`

Description

The name of the bucket.

Type

String

Parent

`bucket`

`pool`

Description

The pool the bucket is stored in.

Type

String

Parent

`bucket`

`id`

Description

The unique bucket ID.

Type

String

Parent

`bucket`

`marker`

Description

Internal bucket tag.

Type

String

Parent

`bucket`

`owner`

Description

The user ID of the bucket owner.

Type

String

Parent

`bucket`

`usage`

Description

Storage usage information.

Type

Container

Parent

`bucket`

`index`

Description

Status of bucket index.

Type

String

Parent

`bucket`

If successful, then the request returns a bucket's container with the bucket information.

Special Error Responses

`IndexRepairFailed`

Description

Bucket index repair failed.

Code

409 Conflict

Set a bucket quota

To set a quota, the user must have `users` capability set with `write` permission.

Syntax

```
PUT /admin/user?quota&uid=UID&quota-type=bucket
```

The content must include a JSON representation of the quota settings as encoded in the corresponding read operation.

Get usage information

Learn how to get bandwidth usage information.

Capabilities

```
`usage=read`
```

Syntax

```
GET /admin/usage?format=json HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`uid`

Description

The user for which the information is requested.

Type

String

Required

Yes

`start`

Description

The date, and optionally, the time of when the data request started. For example, `2012-09-25 16:00:00`.

Type

String

Required

No

`end`

Description

The date, and optionally, the time of when the data request ended. For example, `2012-09-25 16:00:00`.

Type

String

Required

No

`show-entries`

Description

Specifies whether data entries should be returned.

Type

Boolean

Required

No

`show-summary`

Description

Specifies whether data entries should be returned.

Type

Boolean

Required

No

Response Entities

`usage`

Description

A container for the usage information.

Type

Container

`entries`

Description

A container for the usage entries information.

Type

Container

`user`

Description

A container for the user data information.

Type

Container

`owner`

Description

The name of the user that owns the buckets.

Type

String

`bucket`

Description

The bucket name.

Type

String

`time`

Description

Time lower bound for which data is being specified that is rounded to the beginning of the first relevant hour.

Type

String

`epoch`

Description

The time specified in seconds since `1/1/1970`.

Type

String

`categories`

Description

A container for stats categories.

Type

Container

`entry`

Description

A container for stats entry.

Type

Container

`category`

Description

Name of request category for which the stats are provided.

Type

String

`bytes_sent`

Description

Number of bytes sent by the Ceph Object Gateway.

Type

Integer

`bytes_received`

Description

Number of bytes received by the Ceph Object Gateway.

Type

Integer

`ops`

Description

Number of operations.

Type

Integer

`successful_ops`

Description

Number of successful operations.

Type

Integer

`summary`

Description

Number of successful operations.

Type

Container

`total`

Description

A container for stats summary aggregated total.

Type

Container

If successful, the response contains the requested information.

Remove usage information

Learn how to remove usage information.

> **Important:** If no dates are specified, all usage information is removed.

Capabilities

```
`usage=write`
```

Syntax

```
DELETE /admin/usage?format=json HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
```

Request Parameters

`uid`

Description

The user for which the information is requested.

Type

String

Example

`foo_user`

Required

Yes

`start`

Description

The date, and optionally, the time of when the data request started.

Type

String

Example

`2012-09-25 16:00:00`

Required

No

`end`

Description

The date, and optionally, the time of when the data request ended.

Type

String

Example

`2012-09-25 16:00:00`

Required

No

`remove-all`

Description

Required when `uid` is not specified, in order to acknowledge multi-user data removal.

Type

Boolean

Example

True [False]

Required

No

Standard error responses

Get to know the standard error responses and their descriptions.

`AccessDenied`

Description

Access denied.

Code

403 Forbidden

`InternalError`

Description

Internal server error.

Code

500 Internal Server Error

`NoSuchUser`

Description

User does not exist.

Code

404 Not Found

`NoSuchBucket`

Description

Bucket does not exist.

Code

404 Not Found

`NoSuchKey`

Description

No such access key.

Code

404 Not Found

Ceph Object Gateway and the S3 API

As a developer, you can use a RESTful application programming interface (API) that is compatible with the Amazon S3 data access model. You can manage the buckets and objects stored in an Red Hat Ceph Storage cluster through the Ceph Object Gateway.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A RESTful client.

S3 limitations

Understand Amazon S3 limitations.

Important: These limitations should be used with caution as there are implications that are related to your hardware selections. Always discuss these requirements with your Red Hat account team.

Maximum object size when using Amazon S3

Individual Amazon S3 objects can range in size from a minimum of 0 B to a maximum of 5 TB. The largest object that can be uploaded in a single `PUT` is 5 GB. For objects larger than 100 MB, consider using the Multipart Upload capability.

Maximum metadata size when using Amazon S3

There is no defined limit on the total size of user metadata that can be applied to an object, but a single HTTP request is limited to 16,000 bytes.

The amount of data overhead Red Hat Ceph Storage cluster produces to store S3 objects and metadata

The estimate is 200-300 bytes plus the length of the object name. Versioned objects use additional space proportional to the number of versions. Also, transient overhead is produced during multi-part upload and other transactional updates, but these overheads are recovered during garbage collection.

For more information, see "S3 unsupported header fields" on page 211.

Accessing the Ceph Object Gateway with the S3 API

As a developer, you must configure access to the Ceph Object Gateway and the Secure Token Service (STS) before you can start using the Amazon S3 API.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A running Ceph Object Gateway.
- A RESTful client.

S3 authentication

Requests to the Ceph Object Gateway can be either authenticated or unauthenticated. Ceph Object Gateway assumes that unauthenticated requests are sent by an anonymous user. Ceph Object Gateway supports canned ACLs.

For most use cases, clients use existing open source libraries like the Amazon SDK's `AmazonS3Client` for Java, and Python Boto. With open source libraries, you pass in the access key and secret key and the library builds the request header and authentication signature for you. However, you can create requests and sign them too.

Authenticating a request requires including an access key and a base 64-encoded Hash-based Message Authentication Code (HMAC) in the request before it is sent to the Ceph Object Gateway server. Ceph Object Gateway uses an S3-compatible authentication approach.

Example

```
HTTP/1.1
PUT /buckets/bucket/object.mpeg
Host: cname.domain.com
Date: Mon, 2 Jan 2012 00:01:01 +0000
Content-Encoding: mpeg
Content-Length: 9999999

Authorization: AWS _ACCESS_KEY_:_HASH_OF_HEADER_AND_SECRET_
```

In the example, replace `ACCESS_KEY` with the value for the access key ID followed by a colon (`:`). Replace `HASH_OF_HEADER_AND_SECRET` with a hash of a canonicalized header string and the secret corresponding to the access key ID.

Generate a hash of header string and secret.

To generate the hash of the header string and secret:

1. Get the value of the header string.

2. Normalize the request header string into canonical form.
3. Generate an HMAC with an SHA-1 hashing algorithm.
4. Encode the `hmac` result as base-64.

Normalizing header

Normalize the header into canonical form.

1. Get all `content-` headers.
2. Remove all `content-` headers except for `content-type` and `content-md5`.
3. Ensure the `content-` header names are lowercase.
4. Sort the `content-` headers lexicographically.
5. Ensure you have a `Date` header AND ensure the specified date uses GMT and not an offset.
6. Get all headers that begins with `x-amz-`.
7. Ensure that the `x-amz-` headers are all lowercase.
8. Sort the `x-amz-` headers lexicographically.
9. Combine multiple instances of the same field name into a single field and separate the field values with a comma.
10. Replace white space and line breaks in header values with a single space.
11. Remove white space before and after colons.
12. Append a new line after each header.
13. Merge the headers back into the request header.

Replace the *HASH_OF_HEADER_AND_SECRET* with the base-64 encoded HMAC string.

Reference

- Signing and Authenticating REST Requests

S3 server-side encryption

The Ceph Object Gateway supports server-side encryption of uploaded objects for the S3 application programming interface (API). Server-side encryption means that the S3 client sends data over HTTP in its unencrypted form, and the Ceph Object Gateway stores that data in the Red Hat Ceph Storage cluster in encrypted form.

Important: To use encryption, client requests must send requests over an SSL connection. S3 encryption from a client is not supported unless the Ceph Object Gateway uses SSL. For testing purposes, administrators can disable SSL during testing. SSL can be disabled by setting the `rgw_crypt_require_ssl` configuration setting to `false` at run time by using the **`ceph config set client.rgw`** command and then restarting the Ceph Object Gateway instance.

Note: S3 object encryption of Static Large Object (SLO) and Dynamic Large Object (DLO) are not supported.

In a production environment, it might not be possible to send encrypted requests over SSL. In such a case, send requests by using HTTP with server-side encryption.

There are two options for the management of encryption keys:

Customer-provided keys

When using customer-provided keys, the S3 client passes an encryption key along with each request to read or write encrypted data. It is the customer's responsibility to manage those keys. Customers must remember which key the Ceph Object Gateway used to encrypt each object.

Ceph Object Gateway implements the customer-provided key behavior in the S3 API according to the Amazon SSE-C specification.

Since the customer handles the key management and the S3 client passes keys to the Ceph Object Gateway, the Ceph Object Gateway requires no special configuration to support this encryption mode.

Key management service

> **Note:** For the latest supported key managers, see "Compatibility matrix for Red Hat Ceph Storage 9.0" on page 0.

When using a key management service, the secure key management service stores the keys and the Ceph Object Gateway retrieves them on demand to serve requests to encrypt or decrypt data.

Ceph Object Gateway implements the key management service behavior in the S3 API according to the Amazon SSE-KMS specification.

For more information, see Configuring server-side encryption and HashiCorp Vault.

Related information

Amazon SSE-C
Amazon SSE-KMS

S3 access control lists

Ceph Object Gateway supports S3-compatible Access Control Lists (ACL) functionality.

An ACL is a list of access grants that specify which operations a user can perform on a bucket or on an object. Each grant has a different meaning when applied to a bucket versus applied to an object, as listed in "User operations" on page 89.

User operations		
Permission	**Bucket**	**Object**
`READ`	Grantee can list the objects in the bucket.	Grantee can read the object.
`WRITE`	Grantee can write or delete objects in the bucket.	N/A
`READ_ACP`	Grantee can read bucket ACL.	Grantee can read the object ACL.
`WRITE_ACP`	Grantee can write bucket ACL.	Grantee can write to the object ACL.
`FULL_CONTROL`	Grantee has full permissions for object in the bucket.	Grantee can read or write to the object ACL.

Preparing access to the Ceph Object Gateway using S3

Prepare the Ceph Object Gatway node before attempting to access the gateway server.

Prerequisites

- Installation of the Ceph Object Gateway software.
- Root-level access to the Ceph Object Gateway node.

Procedure

1. As `root`, open port `8080` on the firewall:

```
[root@rgw ~]# firewall-cmd --zone=public --add-port=8080/tcp --permanent
[root@rgw ~]# firewall-cmd --reload
```

2. Add a wildcard to the DNS server that you are using for the gateway, as mentioned in Add a wildcard to the DNS.

   You can also set up the gateway node for local DNS caching. To do so, run the following steps:

   a. As `root`, install and setup `dnsmasq`:

```
[root@rgw ]# yum install dnsmasq
[root@rgw ]# echo "address=/.FQDN_OF_GATEWAY_NODE/IP_OF_GATEWAY_NODE" | tee --
append /etc/dnsmasq.conf
[root@rgw ]# systemctl start dnsmasq
[root@rgw ]# systemctl enable dnsmasq
```

   Replace `IP_OF_GATEWAY_NODE` and `FQDN_OF_GATEWAY_NODE` with the IP address and FQDN of the gateway node.

   b. As `root`, stop NetworkManager:

```
[root@rgw ~]# systemctl stop NetworkManager
[root@rgw ~]# systemctl disable NetworkManager
```

   c. As `root`, set the gateway server's IP as the nameserver:

```
[root@rgw ]# echo "DNS1=IP_OF_GATEWAY_NODE" | tee --append /etc/sysconfig/
network-scripts/ifcfg-eth0
[root@rgw ]# echo "IP_OF_GATEWAY_NODE FQDN_OF_GATEWAY_NODE" | tee --append /etc/
hosts
[root@rgw ]# systemctl restart network
[root@rgw ]# systemctl enable network
[root@rgw ~]# systemctl restart dnsmasq
```

   Replace `IP_OF_GATEWAY_NODE` and `FQDN_OF_GATEWAY_NODE` with the IP address and FQDN of the gateway node.

   d. Verify subdomain requests:

```
[user@rgw ~]$ ping mybucket.FQDN_OF_GATEWAY_NODE
```

   Replace `FQDN_OF_GATEWAY_NODE` with the FQDN of the gateway node.

> **Warning:** Setting up the gateway server for local DNS caching is for testing purposes only. You won't be able to access the outside network after doing this. It is strongly recommended to use a proper DNS server for the Red Hat Ceph Storage cluster and gateway node.

3. Create the `radosgw` user for `S3` access carefully and copy the generated `access_key` and `secret_key`.You will need these keys for `S3` access and subsequent bucket management tasks. For more information, see Create an S3 user

Accessing the Ceph Object Gateway that uses Ruby AWS S3

Use Ruby programming language along with `aws-s3` gem for `S3` access. Run the steps that are described on the node that is used for accessing the Ceph Object Gateway server with `Ruby AWS::S3`.

Prerequisites

- User-level access to Ceph Object Gateway.
- Root-level access to the node that can access the Ceph Object Gateway.

- Internet access.

Procedure

1. Install the `ruby` package:

```
[root@dev ~]# yum install ruby
```

> **Note:** The command installs `ruby` and its essential dependencies like `rubygems` and `ruby-libs`. If somehow the command does not install all the dependencies, install them separately.

2. Install the `aws-s3` Ruby package:

```
[root@dev ~]# gem install aws-s3
```

3. Create a project directory:

```
[user@dev ~]$ mkdir ruby_aws_s3
[user@dev ~]$ cd ruby_aws_s3
```

4. Create the connection file:

```
[user@dev ~]$ vim conn.rb
```

5. Paste the following contents into the `conn.rb` file:

   Syntax

```
#!/usr/bin/env ruby

require aws/s3
require resolv-replace

AWS::S3::Base.establish_connection!(
        :server            => FQDN_OF_GATEWAY_NODE,
        :port           => 8080,
        :access_key_id     => MY_ACCESS_KEY,
        :secret_access_key => MY_SECRET_KEY
)
```

   Replace `FQDN_OF_GATEWAY_NODE` with the FQDN of the Ceph Object Gateway node.

   Example

```
#!/usr/bin/env ruby

require 'aws/s3'
require 'resolv-replace'

AWS::S3::Base.establish_connection!(
        :server            => 'testclient.englab.pnq.redhat.com',
        :port           => '8080',
        :access_key_id     => '98J4R9P22P5CDL65HKP8',
        :secret_access_key => '6C+jcaP0dp0+FZfrRNgyGA9EzRy25pURldwje049'
)
```

   Save the file and exit the editor.

6. Make the file executable:

```
[user@dev ~]$ chmod +x conn.rb
```

7. Run the file:

```
[user@dev ~]$ ./conn.rb | echo $?
```

If you have provided the values correctly in the file, the output of the command is `0`.

8. Create a file for creating a bucket:

```
[user@dev ~]$ vim create_bucket.rb
```

Paste the following contents into the file:

```
#!/usr/bin/env ruby

load 'conn.rb'

AWS::S3::Bucket.create('my-new-bucket1')
```

Save the file and exit the editor.

9. Make the file executable:

```
[user@dev ~]$ chmod +x create_bucket.rb
```

10. Run the file:

```
[user@dev ~]$ ./create_bucket.rb
```

If the output of the command is `true` it would mean that a bucket `my-new-bucket1` was created successfully.

11. Create a file for listing owned buckets:

```
[user@dev ~]$ vim list_owned_buckets.rb
```

Paste the following content into the file:

```
#!/usr/bin/env ruby

load 'conn.rb'

AWS::S3::Service.buckets.each do |bucket|
        puts "{bucket.name}\t{bucket.creation_date}"
end
```

Save the file and exit the editor.

12. Make the file executable:

```
[user@dev ~]$ chmod +x list_owned_buckets.rb
```

13. Run the file:

```
[user@dev ~]$ ./list_owned_buckets.rb
```

The output should look something like this:

```
my-new-bucket1 2020-01-21 10:33:19 UTC
```

14. Create a file for creating an object:

```
[user@dev ~]$ vim create_object.rb
```

Paste the following contents into the file:

```
#!/usr/bin/env ruby

load 'conn.rb'

AWS::S3::S3Object.store(
```

```
        'hello.txt',
        'Hello World!',
        'my-new-bucket1',
        :content_type => 'text/plain'
)
```

Save the file and exit the editor.

15. Make the file executable:

```
[user@dev ~]$ chmod +x create_object.rb
```

16. Run the file:

```
[user@dev ~]$ ./create_object.rb
```

This command creates a file `hello.txt` with the string `Hello World!`.

17. Create a new file for listing a bucket's content:

```
[user@dev ~]$ vim list_bucket_content.rb
```

Paste the following content into the file:

```
#!/usr/bin/env ruby

load 'conn.rb'

new_bucket = AWS::S3::Bucket.find('my-new-bucket1')
new_bucket.each do |object|
        puts "{object.key}\t{object.about['content-length']}\t{object.about['last-
modified']}"
end
```

Save the file and exit the editor.

18. Make the file executable.

```
[user@dev ~]$ chmod +x list_bucket_content.rb
```

19. Run the file:

```
[user@dev ~]$ ./list_bucket_content.rb
```

The output looks something like this:

```
hello.txt    12    Fri, 22 Jan 2020 15:54:52 GMT
```

20. Create a file for deleting an empty bucket:

```
[user@dev ~]$ vim del_empty_bucket.rb
```

Paste the following contents into the file:

```
#!/usr/bin/env ruby

load 'conn.rb'

AWS::S3::Bucket.delete('my-new-bucket1')
```

Save the file and exit the editor.

21. Make the file executable:

```
[user@dev ~]$ chmod +x del_empty_bucket.rb
```

22. Run the file:

```
[user@dev ~]$ ./del_empty_bucket.rb | echo $?
```

If the bucket is successfully deleted, the command returns `0` as output.

> **Note:** Edit the `create_bucket.rb` file to create empty buckets, for example, `my-new-bucket4`, `my-new-bucket5`. Next, edit the `del_empty_bucket.rb` file before deleting the empty buckets.

23. Create a file for deleting nonempty buckets:

```
[user@dev ~]$ vim del_non_empty_bucket.rb
```

Paste the following contents into the file:

```
#!/usr/bin/env ruby
load 'conn.rb'
AWS::S3::Bucket.delete('my-new-bucket1', :force => true)
```

Save the file and exit the editor.

24. Make the file executable:

```
[user@dev ~]$ chmod +x del_non_empty_bucket.rb
```

25. Run the file:

```
[user@dev ~]$ ./del_non_empty_bucket.rb | echo $?
```

If the bucket is successfully deleted, the command returns `0` as output.

26. Create a file for deleting an object:

```
[user@dev ~]$ vim delete_object.rb
```

Paste the following contents into the file:

```
#!/usr/bin/env ruby
load 'conn.rb'
AWS::S3::S3Object.delete('hello.txt', 'my-new-bucket1')
```

Save the file and exit the editor.

27. Make the file executable:

```
[user@dev ~]$ chmod +x delete_object.rb
```

28. Run the file:

```
[user@dev ~]$ ./delete_object.rb
```

This command deletes the object `hello.txt`.

Accessing the Ceph Object Gateway using Ruby AWS SDK

Use the Ruby programming language along with `aws-sdk` gem for `S3` access. Run the steps on the node used for accessing the Ceph Object Gateway server with `Ruby AWS::SDK`.

Prerequisites

- User-level access to Ceph Object Gateway.
- Root-level access to the node accessing the Ceph Object Gateway.
- Internet access.

Procedure

1. Install the `ruby` package:

```
[root@dev ~]# yum install ruby
```

> **Note:** This command installs `ruby` and its essential dependencies like `rubygems` and `ruby-libs`. If somehow the command does not install all the dependencies, install them separately.

2. Install the `aws-sdk` Ruby package:

```
[root@dev ~]# gem install aws-sdk
```

3. Create a project directory:

```
[user@dev ~]$ mkdir ruby_aws_sdk
[user@dev ~]$ cd ruby_aws_sdk
```

4. Create the connection file:

```
[user@dev ~]$ vim conn.rb
```

5. Paste the following contents into the `conn.rb` file:

   Syntax

```
#!/usr/bin/env ruby

require aws-sdk
require resolv-replace

Aws.config.update(
        endpoint: http://FQDN_OF_GATEWAY_NODE:8080,
        access_key_id: MY_ACCESS_KEY,
        secret_access_key: MY_SECRET_KEY,
        force_path_style: true,
        region: us-east-1
)
```

   Replace `FQDN_OF_GATEWAY_NODE` with the FQDN of the Ceph Object Gateway node. Replace `MY_ACCESS_KEY` and `MY_SECRET_KEY` with the `access_key` and `secret_key` that were generated when you created the `radosgw` user for S3 access as mentioned in the Create an S3 user section.

   Example

```
#!/usr/bin/env ruby

require 'aws-sdk'
require 'resolv-replace'

Aws.config.update(
        endpoint: 'http://testclient.englab.pnq.redhat.com:8080',
        access_key_id: '98J4R9P22P5CDL65HKP8',
        secret_access_key: '6C+jcaP0dp0+FZfrRNgyGA9EzRy25pURldwje049',
        force_path_style: true,
        region: 'us-east-1'
)
```

   Save the file and exit the editor.

6. Make the file executable:

```
[user@dev ~]$ chmod +x conn.rb
```

7. Run the file:

```
[user@dev ~]$ ./conn.rb | echo $?
```

If you have provided the values correctly in the file, the output of the command will be `0`.

8. Create a file for creating a bucket:

```
[user@dev ~]$ vim create_bucket.rb
```

Paste the following contents into the file:

Syntax

```
#!/usr/bin/env ruby

load 'conn.rb'

s3_client = Aws::S3::Client.new
s3_client.create_bucket(bucket: 'my-new-bucket2')
```

Save the file and exit the editor.

9. Make the file executable:

```
[user@dev ~]$ chmod +x create_bucket.rb
```

10. Run the file:

```
[user@dev ~]$ ./create_bucket.rb
```

If the output of the command is `true`, this means that bucket `my-new-bucket2` was created successfully.

11. Create a file for listing owned buckets:

```
[user@dev ~]$ vim list_owned_buckets.rb
```

Paste the following content into the file:

```
#!/usr/bin/env ruby

load 'conn.rb'

s3_client = Aws::S3::Client.new
s3_client.list_buckets.buckets.each do |bucket|
        puts "{bucket.name}\t{bucket.creation_date}"
end
```

Save the file and exit the editor.

12. Make the file executable:

```
[user@dev ~]$ chmod +x list_owned_buckets.rb
```

13. Run the file:

```
[user@dev ~]$ ./list_owned_buckets.rb
```

The output should look something like this:

```
my-new-bucket2 2020-01-21 10:33:19 UTC
```

14. Create a file for creating an object:

```
[user@dev ~]$ vim create_object.rb
```

Paste the following contents into the file:

```
#!/usr/bin/env ruby

load 'conn.rb'

s3_client = Aws::S3::Client.new
s3_client.put_object(
        key: 'hello.txt',
        body: 'Hello World!',
        bucket: 'my-new-bucket2',
        content_type: 'text/plain'
)
```

Save the file and exit the editor.

15. Make the file executable:

```
[user@dev ~]$ chmod +x create_object.rb
```

16. Run the file:

```
[user@dev ~]$ ./create_object.rb
```

This will create a file `hello.txt` with the string `Hello World!`.

17. Create a file for listing a bucket's content:

```
[user@dev ~]$ vim list_bucket_content.rb
```

Paste the following content into the file:

```
#!/usr/bin/env ruby

load 'conn.rb'

s3_client = Aws::S3::Client.new
s3_client.list_objects(bucket: 'my-new-bucket2').contents.each do |object|
        puts "{object.key}\t{object.size}"
end
```

Save the file and exit the editor.

18. Make the file executable.

```
[user@dev ~]$ chmod +x list_bucket_content.rb
```

19. Run the file:

```
[user@dev ~]$ ./list_bucket_content.rb
```

The output will look something like this:

```
hello.txt    12    Fri, 22 Jan 2020 15:54:52 GMT
```

20. Create a file for deleting an empty bucket:

```
[user@dev ~]$ vim del_empty_bucket.rb
```

Paste the following contents into the file:

```
#!/usr/bin/env ruby
```

```
load 'conn.rb'

s3_client = Aws::S3::Client.new
s3_client.delete_bucket(bucket: 'my-new-bucket2')
```

Save the file and exit the editor.

21. Make the file executable:

```
[user@dev ~]$ chmod +x del_empty_bucket.rb
```

22. Run the file:

```
[user@dev ~]$ ./del_empty_bucket.rb | echo $?
```

If the bucket is successfully deleted, the command will return `0` as output.

> **Note:** Edit the `create_bucket.rb` file to create empty buckets, for example, `my-new-bucket6`, `my-new-bucket7`. Next, edit the `del_empty_bucket.rb` file prior to deleting the empty buckets.

23. Create a file for deleting a non-empty bucket:

```
[user@dev ~]$ vim del_non_empty_bucket.rb
```

Paste the following contents into the file:

```
#!/usr/bin/env ruby

load 'conn.rb'

s3_client = Aws::S3::Client.new
Aws::S3::Bucket.new('my-new-bucket2', client: s3_client).clear!
s3_client.delete_bucket(bucket: 'my-new-bucket2')
```

Save the file and exit the editor.

24. Make the file executable:

```
[user@dev ~]$ chmod +x del_non_empty_bucket.rb
```

25. Run the file:

```
[user@dev ~]$ ./del_non_empty_bucket.rb | echo $?
```

If the bucket is successfully deleted, the command will return `0` as output.

26. Create a file for deleting an object:

```
[user@dev ~]$ vim delete_object.rb
```

Paste the following contents into the file:

```
#!/usr/bin/env ruby

load 'conn.rb'

s3_client = Aws::S3::Client.new
s3_client.delete_object(key: 'hello.txt', bucket: 'my-new-bucket2')
```

Save the file and exit the editor.

27. Make the file executable:

```
[user@dev ~]$ chmod +x delete_object.rb
```

28. Run the file:

```
[user@dev ~]$ ./delete_object.rb
```

This will delete the object `hello.txt`.

Accessing the Ceph Object Gateway using PHP

Use PHP scripts for S3 access. This procedure provides some example PHP scripts to do various tasks, such as deleting a bucket or an object.

> **Important:** The examples are tested against `php v5.4.16` and `aws-sdk v2.8.24`.

Prerequisites

- Root-level access to a development workstation.
- Internet access.

1. Install the `php` package:

```
[root@dev ~]# yum install php
```

2. Download the zip archive of `aws-sdk` for PHP and extract it.
3. Create a project directory:

```
[user@dev ~]$ mkdir php_s3
[user@dev ~]$ cd php_s3
```

4. Copy the extracted `aws` directory to the project directory. For example:

```
[user@dev ~]$ cp -r ~/Downloads/aws/ ~/php_s3/
```

5. Create the connection file:

```
[user@dev ~]$ vim conn.php
```

6. Paste the following contents in the `conn.php` file:

Syntax

```
<?php
define(AWS_KEY, MY_ACCESS_KEY);
define(AWS_SECRET_KEY, MY_SECRET_KEY);
define(HOST, FQDN_OF_GATEWAY_NODE);
define(PORT, 8080);

// require the AWS SDK for php library
require /PATH_TO_AWS/aws-autoloader.php;

use Aws\S3\S3Client;

// Establish connection with host using S3 Client
client = S3Client::factory(array(
    base_url => HOST,
    port => PORT,
    key      => AWS_KEY,
    secret   => AWS_SECRET_KEY
));
?>
```

Replace `FQDN_OF_GATEWAY_NODE` with the FQDN of the gateway node. Replace `PATH_TO_AWS` with the absolute path to the extracted `aws` directory that you copied to the `php` project directory.

Save the file and exit the editor.

7. Run the file:

```
[user@dev ~]$ php -f conn.php | echo $?
```

If you have provided the values correctly in the file, the output of the command will be `0`.

8. Create a new file for creating a bucket:

```
[user@dev ~]$ vim create_bucket.php
```

Paste the following contents into the new file:

Syntax

```
<?php

include 'conn.php';

client->createBucket(array('Bucket' => 'my-new-bucket3'));

?>
```

Save the file and exit the editor.

9. Run the file:

```
[user@dev ~]$ php -f create_bucket.php
```

10. Create a new file for listing owned buckets:

```
[user@dev ~]$ vim list_owned_buckets.php
```

Paste the following content into the file:

Syntax

```
<?php

include 'conn.php';

blist = client->listBuckets();
echo "Buckets belonging to " . blist['Owner']['ID'] . ":\n";
foreach (blist['Buckets'] as b) {
    echo "{b['Name']}\t{b['CreationDate']}\n";
}

?>
```

Save the file and exit the editor.

11. Run the file:

```
[user@dev ~]$ php -f list_owned_buckets.php
```

The output should look similar to this:

```
my-new-bucket3 2020-01-21 10:33:19 UTC
```

12. Create an object by first creating a source file named `hello.txt`:

```
[user@dev ~]$ echo "Hello World!" > hello.txt
```

13. Create a new php file:

```
[user@dev ~]$ vim create_object.php
```

Paste the following contents into the file:

Syntax

```
<?php

include 'conn.php';

key         = 'hello.txt';
source_file = './hello.txt';
acl         = 'private';
bucket      = 'my-new-bucket3';
client->upload(bucket, key, fopen(source_file, 'r'), acl);

?>
```

Save the file and exit the editor.

14. Run the file:

```
[user@dev ~]$ php -f create_object.php
```

This will create the object `hello.txt` in bucket `my-new-bucket3`.

15. Create a new file for listing a bucket's content:

```
[user@dev ~]$ vim list_bucket_content.php
```

Paste the following content into the file:

Syntax

```
<?php

include 'conn.php';

o_iter = client->getIterator('ListObjects', array(
    'Bucket' => 'my-new-bucket3'
));
foreach (o_iter as o) {
    echo "{o['Key']}\t{o['Size']}\t{o['LastModified']}\n";
}
?>
```

Save the file and exit the editor.

16. Run the file:

```
[user@dev ~]$ php -f list_bucket_content.php
```

The output will look similar to this:

```
hello.txt    12    Fri, 22 Jan 2020 15:54:52 GMT
```

17. Create a new file for deleting an empty bucket:

```
[user@dev ~]$ vim del_empty_bucket.php
```

Paste the following contents into the file:

Syntax

```
<?php

include 'conn.php';

client->deleteBucket(array('Bucket' => 'my-new-bucket3'));
?>
```

Save the file and exit the editor.

18. Run the file:

```
[user@dev ~]$ php -f del_empty_bucket.php | echo $?
```

If the bucket is successfully deleted, the command will return `0` as output.

> **Important:** Deleting a non-empty bucket is currently not supported in PHP 2 and newer versions of `aws-sdk`.

> **Note:** Edit the `create_bucket.php` file to create empty buckets, for example, `my-new-bucket4`, `my-new-bucket5`. Next, edit the `del_empty_bucket.php` file accordingly prior to deleting the empty buckets.

19. Create a new file for deleting an object:

```
[user@dev ~]$ vim delete_object.php
```

Paste the following contents into the file:

Syntax

```
<?php

include 'conn.php';

client->deleteObject(array(
    'Bucket' => 'my-new-bucket3',
    'Key'    => 'hello.txt',
));
?>
```

Save the file and exit the editor.

20. Run the file:

```
[user@dev ~]$ php -f delete_object.php
```

This will delete the object `hello.txt`.

S3 bucket operations

Perform bucket operations with the Amazon S3 application programming interface (API) through the Ceph Object Gateway.

"S3 functional operations and support status for buckets" on page 102 describes the Amazon S3 functional operations for buckets, along with the function's support status.

S3 functional operations and support status for buckets

Feature	Status	Notes
List Buckets	Supported	
Create a Bucket	Supported	Different set of canned ACLs.
Put Bucket Website	Supported	
Get Bucket Website	Supported	
Delete Bucket Website	Supported	
Bucket Lifecycle	Partially Supported	`Expiration`, `NoncurrentVersionExpiration` and `AbortIncompleteMultipartUpload` supported.

Feature	Status	Notes
Put Bucket Lifecycle	Partially Supported	`Expiration`, `NoncurrentVersionExpiration` and `AbortIncompleteMultipartUpload` supported.
Delete Bucket Lifecycle	Supported	
Get Bucket Objects	Supported	
Bucket Location	Supported	
Get Bucket Version	Supported	
Put Bucket Version	Supported	
Delete Bucket	Supported	
Get Bucket ACLs	Supported	Different set of canned ACLs
Put Bucket ACLs	Supported	Different set of canned ACLs
Get Bucket cors	Supported	
Put Bucket cors	Supported	
Delete Bucket cors	Supported	
List Bucket Object Versions	Supported	
Head Bucket	Supported	
List Bucket Multipart Uploads	Supported	
Bucket Policies	Partially Supported	
Get a Bucket Request Payment	Supported	
Put a Bucket Request Payment	Supported	
GET `PublicAccessBlock`	Supported	
PUT `PublicAccessBlock`	Supported	
Delete `PublicAccessBlock`	Supported	

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A RESTful client.

S3 create bucket notifications

Create bucket notifications at the bucket level. The notification configuration has the Red Hat Ceph Storage Object Gateway S3 events, `ObjectCreated`, `ObjectRemoved`, and `ObjectLifecycle:Expiration`.

These need to be published and the destination to send the bucket notifications. Bucket notifications are S3 operations.

To create a bucket notification for `s3:objectCreate`, `s3:objectRemove`, and `s3:ObjectLifecycle:Expiration` events, use PUT:

Example

```
client.put_bucket_notification_configuration(
   Bucket=bucket_name,
   NotificationConfiguration={
       'TopicConfigurations': [
           {
               'Id': notification_name,
               'TopicArn': topic_arn,
               'Events': ['s3:ObjectCreated:*', 's3:ObjectRemoved:*',
's3:ObjectLifecycle:Expiration:*']
           }]})
```

> **Important:** `ObjectCreate` events, such as `put`, `post`, `multipartUpload`, and `copy` are supported. `ObjectRemove` events, such as `object_delete` and `s3_multi_object_delete` are also supported.

Request Entities

`NotificationConfiguration`

Description

list of `TopicConfiguration` entities.

Type

Container

Required

Yes

`TopicConfiguration`

Description

`Id`, `Topic`, and `list` of Event entities.

Type

Container

Required

Yes

`id`

Description

Name of the notification.

Type

String

Required

Yes

`Topic`

Description

Topic Amazon Resource Name(ARN)

> **Note:** The topic must be created beforehand.

Type

String

Required

Yes

`Event`

Description

List of supported events. Multiple event entities can be used. If omitted, all events are handled.

Type

String

Required

No

`Filter`

Description

`S3Key`, `S3Metadata` and `S3Tags` entities.

Type

Container

Required

No

`S3Key`

Description

A list of `FilterRule` entities, for filtering based on the object key. At most, 3 entities may be in the list, for example `Name` would be `prefix`, `suffix`, or `regex`. All filter rules in the list must match for the filter to match.

Type

Container

Required

No

`S3Metadata`

Description

A list of `FilterRule` entities, for filtering based on object metadata. All filter rules in the list must match the metadata defined on the object. However, the object still matches if it has other metadata entries not listed in the filter.

Type

Container

Required

No

`S3Tags`

Description

A list of `FilterRule` entities, for filtering based on object tags. All filter rules in the list must match the tags defined on the object. However, the object still matches if it has other tags not listed in the filter.

Type

Container

Required

No

`S3Key.FilterRule`

Description

`Name` and `Value` entities. Name is : `prefix`, `suffix`, or `regex`. The `Value` would hold the key prefix, key suffix, or a regular expression for matching the key, accordingly.

Type

Container

Required

Yes

`S3Metadata.FilterRule`

Description

`Name` and `Value` entities. Name is the name of the metadata attribute for example `x-amz-meta-xxx`. The value is the expected value for this attribute.

Type

Container

Required

Yes

`S3Tags.FilterRule`

Description

`Name` and `Value` entities. Name is the tag key, and the value is the tag value.

Type

Container

Required

Yes

HTTP response

`400`

Status Code

`MalformedXML`

Description

The XML is not well-formed.

`400`

Status Code

`InvalidArgument`

Description

Missing Id or missing or invalid topic ARN or invalid event.

`404`

Status Code

`NoSuchBucket`

Description

The bucket does not exist.

`404`

Status Code

`NoSuchKey`

Description

The topic does not exist.

S3 get bucket notifications

Get a specific notification or list all the notifications configured on a bucket.

Syntax

```
Get /_BUCKET_?notification=_NOTIFICATIONID HTTP/1.1
Host: cname.domain.com
```

```
Date: date
Authorization: AWS _ACCESS_KEY_:_HASH_OF_HEADER_AND_SECRET_
```

Example

```
Get /testbucket?notification=testnotificationID HTTP/1.1
Host: cname.domain.com
Date: date
Authorization: AWS _ACCESS_KEY_:_HASH_OF_HEADER_AND_SECRET_
```

Example Response

```
<NotificationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <TopicConfiguration>
        <Id></Id>
        <Topic></Topic>
        <Event></Event>
        <Filter>
            <S3Key>
                <FilterRule>
                    <Name></Name>
                    <Value></Value>
                </FilterRule>
                 </S3Key>
             <S3Metadata>
                 <FilterRule>
                     <Name></Name>
                     <Value></Value>
                 </FilterRule>
             </S3Metadata>
             <S3Tags>
                 <FilterRule>
                     <Name></Name>
                     <Value></Value>
                 </FilterRule>
             </S3Tags>
         </Filter>
    </TopicConfiguration>
</NotificationConfiguration>
```

> **Note:** The `notification` subresource returns the bucket notification configuration or an empty `NotificationConfiguration` element. The caller must be the bucket owner.

Request Entities

`notification-id`

Description

Name of the notification. All notifications are listed if the ID is not provided.

Type

String

`NotificationConfiguration`

Description

list of `TopicConfiguration` entities.

Type

Container

Required

Yes

`TopicConfiguration`

Description

`Id`, `Topic`, and `list` of Event entities.

Type

Container

Required

Yes

`id`

Description

Name of the notification.

Type

String

Required

Yes

`Topic`

Description

Topic Amazon Resource Name(ARN)

> **Note:** The topic must be created beforehand.

Type

String

Required

Yes

`Event`

Description

Handled event. Multiple event entities may exist.

Type

String

Required

Yes

`Filter`

Description

The filters for the specified configuration.

Type

Container

Required

No

HTTP response

`404`

Status Code

`NoSuchBucket`

Description

The bucket does not exist.

`404`

Status Code

`NoSuchKey`

Description

The notification does not exist if it has been provided.

S3 delete bucket notifications

Delete a specific or all notifications from a bucket.

> **Note:** Notification deletion is an extension to the S3 notification API. Any defined notifications on a bucket are deleted when the bucket is deleted. Deleting an unknown notification for example `double delete`, is not considered an error.

To delete a specific or all notifications use DELETE:

Syntax

```
DELETE /BUCKET?notification=NOTIFICATION_ID HTTP/1.1
```

Example

```
DELETE /testbucket?notification=testnotificationID HTTP/1.1
```

Request Entities

`notification-id`

Description

Name of the notification. All notifications on the bucket are deleted if the notification ID is not provided.

Type

String

HTTP response

`404`

Status Code

`NoSuchBucket`

Description

The bucket does not exist.

Accessing bucket host names

There are two different modes of accessing the buckets. The first, and preferred method identifies the bucket as the first level directory in the URL.

Example

```
GET /mybucket HTTP/1.1
Host: cname.domain.com
```

The second method identifies the bucket via a virtual bucket host name.

Example

```
GET / HTTP/1.1
Host: mybucket.cname.domain.com
```

Tip: While both methods are available, it is preferable to use the first method, because the second method requires expensive domain certification and DNS wild cards.

S3 list buckets

`GET /` returns a list of buckets created by the user making the request. `GET /` only returns buckets created by an authenticated user. You cannot make an anonymous request.

Syntax

```
GET / HTTP/1.1
Host: cname.domain.com

Authorization: AWS_ACCESS_KEY_:_HASH_OF_HEADER_AND_SECRET_
```

Response Entities

`Buckets`

Description

Container for list of buckets.

Type

Container

`Bucket`

Description

Container for bucket information.

Type

Container

`Name`

Description

Bucket name.

Type

String

`CreationDate`

Description

UTC time when the bucket was created.

Type

Date

`ListAllMyBucketsResult`

Description

A container for the result.

Type

Container

`Owner`

Description

A container for the bucket owner's `ID` and `DisplayName`.

Type

Container

`ID`

Description

The bucket owner's ID.

Type

String

`DisplayName`

Description

The bucket owner's display name.

Type

String

S3 return a list of bucket objects

Returns a list of bucket objects.

Syntax

```
GET /BUCKET?max-keys=25 HTTP/1.1
Host: cname.domain.com
```

Parameters

`prefix`

Description

Only returns objects that contain the specified prefix.

Type

String

`delimiter`

Description

The delimiter between the prefix and the rest of the object name.

Type

String

`marker`

Description

A beginning index for the list of objects returned.

Type

String

`max-keys`

Description

The maximum number of keys to return. Default is 1000.

Type

Integer

HTTP Response

`200`

Status Code

`OK`

Description

Buckets retrieved.

`GET /_BUCKET` returns a container for buckets with the following fields:

Bucket Response Entities

`ListBucketResult`

Description

The container for the list of objects.

Type

Entity

`Name`

Description

The name of the bucket whose contents will be returned.

Type

String

`Prefix`

Description

A prefix for the object keys.

Type

String

`Marker`

Description

A beginning index for the list of objects returned.

Type

String

`MaxKeys`

Description

The maximum number of keys returned.

Type

Integer

`Delimiter`

Description

If set, objects with the same prefix will appear in the `CommonPrefixes` list.

Type

String

`IsTruncated`

Description

If `true`, only a subset of the bucket's contents were returned.

Type

Boolean

`CommonPrefixes`

Description

If multiple objects contain the same prefix, they will appear in this list.

Type

Container

The `ListBucketResult` contains objects, where each object is within a `Contents` container.

Object Response Entities

`Contents`

Description

A container for the object.

Type

Object

`Key`

Description

The object's key.

Type

String

`LastModified`

Description

The object's last-modified date and time.

Type

Date

`ETag`

Description

An MD-5 hash of the object. Etag is an entity tag.

Type

String

`Size`

Description

The object's size.

Type

Integer

`StorageClass`

Description

Should always return `STANDARD`.

Type

String

S3 create a new bucket

Create a new bucket. To create a bucket, you must have a user ID and a valid AWS Access Key ID to authenticate requests. You cannot create buckets as an anonymous user.

Constraints

In general, bucket names follow domain name constraints.

- Bucket names must be unique.
- Bucket names cannot be formatted as IP address.
- Bucket names can be between 3 and 63 characters long.
- Bucket names must not contain uppercase characters or underscores.
- Bucket names must start with a lowercase letter or number.
- Bucket names can contain a dash (-).
- Bucket names must be a series of one or more labels. Adjacent labels are separated by a single period (.). Bucket names can contain lowercase letters, numbers, and hyphens. Each label must start and end with a lowercase letter or a number.

> **Note:** The constraints are relaxed if `rgw_relaxed_s3_bucket_names` is set to `true`. The bucket names must still be unique, cannot be formatted as IP address, and can contain letters, numbers, periods, dashes, and underscores of up to 255 characters long.

Syntax

```
PUT /_BUCKET_ HTTP/1.1
Host: cname.domain.com
x-amz-acl: public-read-write

Authorization: AWS _ACCESS_KEY_:_HASH_OF_HEADER_AND_SECRET_
```

Parameters `x-amz-acl`

Description

Canned ACLs.

Valid Values

`private`, `public-read`,`public-read-write`, `authenticated-read`

Required

No

HTTP Response

If the bucket name is unique, within constraints, and unused, the operation succeeds. If a bucket with the same name exists and the user is the bucket owner, the operation will succeed. If the bucket name is already in use, the operation fails.

`409`

Status Code

`BucketAlreadyExists`

Description

Bucket exists under different user's ownership.

S3 put bucket website

The put bucket website API sets the configuration of the website that is specified in the `website` subresource. To configure a bucket as a website, the `website` subresource can be added on the bucket.

> **Note:** `Put` operation requires `S3:PutBucketWebsite` permission. By default, only the bucket owner can configure the website attached to a bucket.

Syntax

```
PUT /BUCKET?website-configuration=HTTP/1.1
```

Example

```
PUT /testbucket?website-configuration=HTTP/1.1
```

Reference

- For more information about this API call, see S3 API.

S3 get bucket website

The get bucket website API retrieves the configuration of the website that is specified in the `website` subresource.

> **Note:** `Get` operation requires the `S3:GetBucketWebsite` permission. By default, only the bucket owner can read the bucket website configuration.

Syntax

```
GET /BUCKET?website-configuration=HTTP/1.1
```

Example

```
GET /testbucket?website-configuration=HTTP/1.1
```

Reference

- For more information about this API call, see S3 API.

S3 delete bucket website

The delete bucket website API removes the website configuration for a bucket.

Syntax

```
DELETE /BUCKET?website-configuration=HTTP/1.1
```

Example

```
DELETE /testbucket?website-configuration=HTTP/1.1
```

Reference

- For more information about this API call, see S3 API.

S3 put bucket replication

The put bucket replication API configures replication configuration for a bucket or replaces an existing one.

```
PUT /BUCKET_NAME?replication HTTP/1.1
```

Reference

- For more information about this API call, see S3 API.

S3 get bucket replication

The get bucket replication API returns the replication configuration of a bucket.

```
GET /BUCKET_NAME?replication HTTP/1.1
```

Reference

- For more information about this API call, see S3 API.

S3 delete bucket replication

The delete bucket replication API deletes the replication configuration from a bucket.

```
DELETE /BUCKET_NAME?website-configuration=HTTP/1.1
```

Reference

- For more information about this API call, see S3 API.

S3 delete a bucket

Deletes a bucket. You can reuse bucket names following a successful bucket removal.

Syntax

```
DELETE /_BUCKET_ HTTP/1.1
Host: cname.domain.com

Authorization: AWS _ACCESS_KEY_:_HASH_OF_HEADER_AND_SECRET_
```

HTTP Response

```
204
```

Status Code

```
No Content
```

Description

Bucket removed.

S3 bucket lifecycle

You can use a bucket lifecycle configuration to manage your objects so they are stored effectively throughout their lifetime. The S3 API in the Ceph Object Gateway supports a subset of the AWS bucket lifecycle actions:

`Expiration`

This action defines the lifespan of objects within a bucket. It takes the number of days the object lives or expiration date, at which point Ceph Object Gateway deletes the object. If the bucket doesn't enable versioning, Ceph Object Gateway deletes the object permanently. If the bucket enables versioning, Ceph Object Gateway creates a delete marker for the current version, and then delete the current version.

`NoncurrentVersionExpiration`

This action defines the lifespan of noncurrent object versions within a bucket. To use this feature, you must enable bucket versioning. It takes the number of days a noncurrent object lives, at which point Ceph Object Gateway deletes the noncurrent object.

`NewerNoncurrentVersions`

This action specifies how many noncurrent object versions to retain. You can specify up to 100 noncurrent versions to retain. If the specified number to retain is more than 100, additional noncurrent versions are deleted.

`AbortIncompleteMultipartUpload`

This action defines the number of days an incomplete multipart upload lives before it is cancelled.

`BlockPublicPolicy reject`

This action is for public access block. It calls PUT access point policy and PUT bucket policy that are made through the access point if the specified policy (for either the access point or the underlying bucket) allows public access. It provides settings for access points, buckets, and accounts to help you manage public access to Amazon S3 resources. By default, new buckets, access points, and objects do not allow public access. However, you can modify bucket policies, access point policies, or object permissions to allow public access. S3 Block Public Access settings override these policies and permissions so that you can limit public access to these resources.

The lifecycle configuration contains one or more rules that use the `<Rule>` element.

Example

```
<LifecycleConfiguration>
    <Rule>
      <Prefix/>
      <Status>Enabled</Status>
      <Expiration>
        <Days>10</Days>
      </Expiration>
    </Rule>
</LifecycleConfiguration>
```

A lifecycle rule can apply to all or a subset of objects in a bucket based on the `<Filter>` element that you specify in the lifecycle rule. You can specify a filter in several ways:

- Key prefixes
- Object tags
- Both key prefixes and one or more object tags

Key prefixes

You can apply a lifecycle rule to a subset of objects based on the key name prefix. For example, specifying `<keypre/>` applies to objects that begin with `keypre/`:

```
<LifecycleConfiguration>
    <Rule>
        <Status>Enabled</Status>
        <Filter>
           <Prefix>keypre/</Prefix>
        </Filter>
    </Rule>
</LifecycleConfiguration>
```

You can also apply different lifecycle rules to objects with different key prefixes:

```
<LifecycleConfiguration>
    <Rule>
        <Status>Enabled</Status>
        <Filter>
           <Prefix>keypre/</Prefix>
        </Filter>
    </Rule>
    <Rule>
        <Status>Enabled</Status>
        <Filter>
           <Prefix>mypre/</Prefix>
        </Filter>
    </Rule>
</LifecycleConfiguration>
```

Object tags

You can apply a lifecycle rule to only objects with a specific tag that uses the `<Key>` and `<Value>` elements:

```
<LifecycleConfiguration>
    <Rule>
        <Status>Enabled</Status>
        <Filter>
           <Tag>
              <Key>key</Key>
              <Value>value</Value>
           </Tag>
```

```
        </Filter>
    </Rule>
</LifecycleConfiguration>
```

Both prefixes and one or more tags

In a lifecycle rule, you can specify a filter based on both the key prefix and one or more tags. They must be wrapped in the `<And>` element. A filter can have only one prefix, and zero or more tags:

```
<LifecycleConfiguration>
    <Rule>
        <Status>Enabled</Status>
        <Filter>
          <And>
             <Prefix>key-prefix</Prefix>
             <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
             </Tag>
             <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
             </Tag>
              ...
          </And>
        </Filter>
    </Rule>
</LifecycleConfiguration>
```

Related information

S3 deletes a bucket lifecycle
S3 GET bucket lifecycle
S3 create or replace a bucket lifecycle

S3 GET bucket lifecycle

To get a bucket lifecycle, use `GET` and specify a destination bucket.

Syntax

```
GET /BUCKET?lifecycle HTTP/1.1
Host: cname.domain.com

Authorization: AWS _ACCESS_KEY_:_HASH_OF_HEADER_AND_SECRET_
```

Request Headers

For more information about request headers, see S3 common request headers.

Response

The response contains the bucket lifecycle and its elements.

S3 create or replace a bucket lifecycle

To create or replace a bucket lifecycle, use `PUT` and specify a destination bucket and a lifecycle configuration. The Ceph Object Gateway only supports a subset of the S3 lifecycle functionality.

Syntax

```
PUT /_BUCKET_?lifecycle HTTP/1.1
Host: cname.domain.com

Authorization: AWS _ACCESS_KEY_:_HASH_OF_HEADER_AND_SECRET_
<LifecycleConfiguration>
  <Rule>
    <Expiration>
      <Days>10</Days>
    </Expiration>
  </Rule>
    ...
```

```
  <Rule>
  </Rule>
</LifecycleConfiguration>
```

Request Headers

`content-md5`

Description

A base64 encoded MD-5 hash of the message

Valid Values

String No defaults or constraints.

Required

No

Reference

- For more information about Amazon S3 common request headers, see S3 common request headers.
- For more information about Amazon S3 bucket lifecycles, see S3 bucket lifecycles.

S3 delete a bucket lifecycle

To delete a bucket lifecycle, use `DELETE` and specify a destination bucket.

Syntax

```
DELETE /_BUCKET_?lifecycle HTTP/1.1
Host: cname.domain.com

Authorization: AWS _ACCESS_KEY_:_HASH_OF_HEADER_AND_SECRET_
```

Request Headers

The request does not contain any special elements.

Response

The response returns common response status.

Reference

- For more information about Amazon S3 common request headers, see S3 common request headers.
- For more information about Amazon S3 common response status codes, see S3 common response status codes.

S3 get bucket location

Retrieves the bucket's zone group. The user needs to be the bucket owner to call this. A bucket can be constrained to a zone group by providing `LocationConstraint` during a PUT request.

Add the `location` subresource to the bucket resource.

Syntax

```
GET /BUCKET?location HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

Response Entities

`LocationConstraint`

Description

The zone group where bucket resides, an empty string for default zone group.

Type

String

S3 get bucket versioning

Retrieves the versioning state of a bucket. The user needs to be the bucket owner to call this.

Add the `versioning` subresource to the bucket resource.

Syntax

```
GET /BUCKET?versioning HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

S3 put bucket versioning

This subresource set the versioning state of an existing bucket. The user needs to be the bucket owner to set the versioning state. If the versioning state has never been set on a bucket, then it has no versioning state. Doing a GET versioning request does not return a versioning state value.

Setting the bucket versioning state:

`Enabled`: Enables versioning for the objects in the bucket. All objects added to the bucket receive a unique version ID. `Suspended`: Disables versioning for the objects in the bucket. All objects added to the bucket receive the version ID null.

Syntax

```
PUT /BUCKET?versioning HTTP/1.1
```

Example

```
PUT /testbucket?versioning HTTP/1.1
```

Bucket Request Entities

`VersioningConfiguration`

Description

A container for the request.

Type

Container

`Status`

Description

Sets the versioning state of the bucket. Valid Values: Suspended/Enabled

Type

String

S3 get bucket access control lists

Retrieves the bucket access control list. The user needs to be the bucket owner or to have been granted `READ_ACP` permission on the bucket.

Add the `acl` subresource to the bucket request.

Syntax

```
GET /BUCKET?acl HTTP/1.1
Host: cname.domain.com
```

```
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

Response Entities

`AccessControlPolicy`

Description

A container for the response.

Type

Container

`AccessControlList`

Description

A container for the ACL information.

Type

Container

`Owner`

Description

A container for the bucket owner's `ID` and `DisplayName`.

Type

Container

`ID`

Description

The bucket owner's ID.

Type

String

`DisplayName`

Description

The bucket owner's display name.

Type

String

`Grant`

Description

A container for `Grantee` and `Permission`.

Type

Container

`Grantee`

Description

A container for the `DisplayName` and `ID` of the user receiving a grant of permission.

Type

Container

`Permission`

Description

The permission given to the `Grantee` bucket.

Type

String

S3 put bucket Access Control Lists

Sets an access control to an existing bucket. The user needs to be the bucket owner or to have been granted `WRITE_ACP` permission on the bucket.

Add the `acl` subresource to the bucket request.

Syntax

```
PUT /_BUCKET_?acl HTTP/1.1
```

Request Entities S3 list multipart uploads

`AccessControlList`

Description

A container for the ACL information.

Type

Container

`Owner`

Description

A container for the bucket owner's `ID` and `DisplayName`.

Type

Container

`ID`

Description

The bucket owner's ID.

Type

String

`DisplayName`

Description

The bucket owner's display name.

Type

String

`Grant`

Description

A container for `Grantee` and `Permission`.

Type

Container

`Grantee`

Description

A container for the `DisplayName` and `ID` of the user receiving a grant of permission.

Type

Container

`Permission`

Description

The permission given to the `Grantee` bucket.

Type

String

S3 get bucket cors

Retrieves the cors configuration information set for the bucket. The user needs to be the bucket owner or to have been granted `READ_ACP` permission on the bucket.

Add the `cors` subresource to the bucket request.

Syntax

```
GET /BUCKET?cors HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

S3 put bucket cors

Sets the cors configuration for the bucket. The user needs to be the bucket owner or to have been granted `READ_ACP` permission on the bucket.

Add the `cors` subresource to the bucket request.

Syntax

```
PUT /BUCKET?cors HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

S3 delete a bucket cors

Deletes the cors configuration information set for the bucket. The user needs to be the bucket owner or to have been granted `READ_ACP` permission on the bucket.

Add the `cors` subresource to the bucket request.

Syntax

```
DELETE /BUCKET?cors HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

S3 list bucket object versions

The *GET /BUCKET?versions* request returns a list of metadata about all the version of objects within a bucket.

The *GET /BUCKET?versions* request requires READ access to the bucket.

Add the versions subresource to the bucket request.

```
GET /BUCKET?versions HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

You can specify parameters for *GET /BUCKET?versions*, but none of them are required. “GET /BUCKET?versions parameters” on page 124 lists the optional parameters.

GET /BUCKET?versions parameters		
Parameter	**Description**	**Type**
`prefix`	Returns in-progress uploads whose keys contain the specified prefix.	String
`delimiter`	The delimiter between the prefix and the rest of the object name.	String
`key-marker`	The beginning marker for the list of uploads.	String
`max-keys`	The maximum number of in-progress uploads. The default is 1000.	Integer
`version-id-marker`	Specifies the object version to begin the list.	String

"GET /BUCKET?versions response entities" on page 124 lists the response entities.

GET /BUCKET?versions response entities		
Response Entity	**Description**	**Type**
`KeyMarker`	The key marker specified by the `key-marker` request parameter, if any.	String
`NextKeyMarker`	The key marker to use in a subsequent request if `IsTruncated` is `true`.	String
`NextUploadIdMarker`	The upload ID marker to use in a subsequent request if `IsTruncated` is `true`.	String
`IsTruncated`	If `true`, only a subset of the bucket's upload contents were returned.	Boolean
`Size`	The size of the uploaded part.	Integer
`DisplayName`	The owner's display name.	String
`ID`	The owner's ID.	String
`Owner`	A container for the `ID` and `DisplayName` of the user who owns the object.	Container
`StorageClass`	The method used to store the resulting object. `STANDARD` or `REDUCED_REDUNDANCY`.	String
`Version`	Container for the version information.	Container
`versionId`	Version ID of an object.	String
`versionIdMarker`	The last version of the key in a truncated response.	String

S3 head bucket

Calls HEAD on a bucket to determine if it exists and if the caller has access permissions. Returns `200 OK` if the bucket exists and the caller has permissions; `404 Not Found` if the bucket does not exist; and, `403 Forbidden` if the bucket exists but the caller does not have access permissions.

Syntax

```
HEAD /_BUCKET_ HTTP/1.1
Host: cname.domain.com
Date: date
Authorization: AWS _ACCESS_KEY_:_HASH_OF_HEADER_AND_SECRET_
```

S3 list multipart uploads

`GET /?uploads` returns a list of the current in-progress multipart uploads, that is, the application initiates a multipart upload, but the service hasn't completed all the uploads yet.

Syntax

```
GET /_BUCKET_?uploads HTTP/1.1
```

You can specify parameters for `GET /_BUCKET_?uploads`, but none of them are required.

Parameters

`prefix`

Description

Returns in-progress uploads whose keys contain the specified prefix.

Type

String

`delimiter`

Description

The delimiter between the prefix and the rest of the object name.

Type

String

`key-marker`

Description

The beginning marker for the list of uploads.

Type

String

`max-keys`

Description

The maximum number of in-progress uploads. The default is 1000.

Type

Integer

`max-uploads`

Description

The maximum number of multipart uploads. The range is from 1-1000. The default is 1000.

Type

Integer

`version-id-marker`

Description

Ignored if `key-marker` isn't specified. Specifies the `ID` of the first upload to list in lexicographical order at or following the `ID`.

Type

String

Response Entities

`ListMultipartUploadsResult`

Description

A container for the results.

Type

Container

`ListMultipartUploadsResult.Prefix`

Description

The prefix specified by the `prefix` request parameter, if any.

Type

String

`Bucket`

Description

The bucket that will receive the bucket contents.

Type

String

`KeyMarker`

Description

The key marker specified by the `key-marker` request parameter, if any.

Type

String

`UploadIdMarker`

Description

The marker specified by the `upload-id-marker` request parameter, if any.

Type

String

`NextKeyMarker`

Description

The key marker to use in a subsequent request if `IsTruncated` is `true`.

Type

String

`NextUploadIdMarker`

Description

The upload ID marker to use in a subsequent request if `IsTruncated` is `true`.

Type

String

`MaxUploads`

Description

The max uploads specified by the `max-uploads` request parameter.

Type

Integer

`Delimiter`

Description

If set, objects with the same prefix will appear in the `CommonPrefixes` list.

Type

String

`IsTruncated`

Description

If `true`, only a subset of the bucket's upload contents were returned.

Type

Boolean

`Upload`

Description

A container for `Key`, `UploadId`, `InitiatorOwner`, `StorageClass`, and `Initiated` elements.

Type

Container

`Key`

Description

The key of the object once the multipart upload is complete.

Type

String

`UploadId`

Description

The `ID` that identifies the multipart upload.

Type

String

`Initiator`

Description

Contains the `ID` and `DisplayName` of the user who initiated the upload.

Type

Container

`DisplayName`

Description

The initiator's display name.

Type

String

`ID`

Description

The initiator's ID.

Type

String

`Owner`

Description

A container for the `ID` and `DisplayName` of the user who owns the uploaded object.

Type

Container

`StorageClass`

Description

The method used to store the resulting object. `STANDARD` or `REDUCED_REDUNDANCY`

Type

String

`Initiated`

Description

The date and time the user initiated the upload.

Type

Date

`CommonPrefixes`

Description

If multiple objects contain the same prefix, they will appear in this list.

Type

Container

`CommonPrefixes.Prefix`

Description

The substring of the key after the prefix as defined by the `prefix` request parameter.

Type

String

S3 bucket policies

The Ceph Object Gateway supports a subset of the Amazon S3 policy language that is applied to buckets.

Creation and Removal

Ceph Object Gateway manages S3 Bucket policies through standard S3 operations rather than using the `radosgw-admin` CLI tool.

Administrators can use the `s3cmd` command to set or delete a policy.

Example

```
$ cat > examplepol
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::usfolks:user/fred"]},
    "Action": "s3:PutObjectAcl",
    "Resource": [
      "arn:aws:s3:::happybucket/*"
    ]
  }]
}

$ s3cmd setpolicy examplepol s3://happybucket
$ s3cmd delpolicy s3://happybucket
```

Limitations

Ceph Object Gateway supports the following S3 actions only:

- `s3:AbortMultipartUpload`
- `s3:CreateBucket`
- `s3:DeleteBucketPolicy`
- `s3:DeleteBucket`
- `s3:DeleteBucketReplication`
- `s3:DeleteReplicationConfiguration`
- `s3:DeleteBucketWebsite`
- `s3:DeleteObject`
- `s3:DeleteObjectVersion`
- `s3:GetBucketAcl`
- `s3:GetBucketCORS`
- `s3:GetBucketLocation`
- `s3:GetBucketPolicy`
- `s3:GetBucketRequestPayment`
- `s3:GetBucketVersioning`
- `s3:GetBucketWebsite`
- `s3:GetBucketReplication`

  `s3:GetReplicationConfiguration`
- `s3:GetLifecycleConfiguration`
- `s3:GetObjectAcl`
- `s3:GetObject`
- `s3:GetObjectTorrent`
- `s3:GetObjectVersionAcl`
- `s3:GetObjectVersion`
- `s3:GetObjectVersionTorrent`
- `s3:ListAllMyBuckets`
- `s3:ListBucketMultiPartUploads`
- `s3:ListBucket`
- `s3:ListBucketVersions`
- `s3:ListMultipartUploadParts`
- `s3:PutBucketAcl`
- `s3:PutBucketCORS`
- `s3:PutBucketPolicy`
- `s3:PutBucketRequestPayment`
- `s3:PutBucketVersioning`
- `s3:PutBucketWebsite`
- `s3:PutBucketReplication`

  `s3:PutReplicationConfiguration`

- `s3:PutLifecycleConfiguration`
- `s3:PutObjectAcl`
- `s3:PutObject`
- `s3:PutObjectVersionAcl`

> **Note:** Ceph Object Gateway does not support setting policies on users, groups, or roles.

The Ceph Object Gateway uses the RGW 'tenant' identifier in place of the Amazon twelve-digit account ID. Administrators who want to use policies between Amazon Web Service (AWS) S3 and Ceph Object Gateway S3 must use the Amazon account ID as the tenant ID during user creation.

With AWS S3, all tenants share a single namespace. By contrast, Ceph Object Gateway gives every tenant its own namespace of buckets. Now, Ceph Object Gateway clients who try to access a bucket that belongs to another tenant must address it as `tenant:bucket` in the S3 request.

In the AWS, a bucket policy can grant access to another account, and that account owner can then grant access to individual users with user permissions. Since Ceph Object Gateway does not yet support user, role, and group permissions, account owners need to grant access directly to individual users.

> **Important:** Granting an entire account access to a bucket grants access to ALL users in that account.

Bucket policies do **NOT** support string interpolation.

Ceph Object Gateway supports the following condition keys:

- `aws:CurrentTime`
- `aws:EpochTime`
- `aws:PrincipalType`
- `aws:Referer`
- `aws:SecureTransport`
- `aws:SourceIp`
- `aws:UserAgent`
- `aws:username`

Ceph Object Gateway **ONLY** supports the following condition keys for the `ListBucket` action:

- `s3:prefix`
- `s3:delimiter`
- `s3:max-keys`

Impact on Swift

Ceph Object Gateway provides no functions to set bucket policies under the Swift API. However, bucket policies that are set with the S3 API govern Swift and S3 operations.

Ceph Object Gateway matches Swift credentials against principals that are specified in a policy.

S3 enable bucket logging

Enables bucket logging for a bucket. Use this operation to configure the target log bucket, prefix, and logging type (Standard or Journal).

Syntax

```
PUT /_BUCKET_?logging HTTP/1.1
```

Request entities

Parameters are XML-encoded in the body of the request, in the following format:

```
<BucketLoggingStatus xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <LoggingEnabled>
    <TargetBucket>string</TargetBucket>
    <TargetGrants>
      <Grant>
        <Grantee>
          <DisplayName>string</DisplayName>
          <EmailAddress>string</EmailAddress>
          <ID>string</ID>
          <xsi:type>string</xsi:type>
          <URI>string</URI>
        </Grantee>
        <Permission>string</Permission>
      </Grant>
    </TargetGrants>
    <TargetObjectKeyFormat>
      <PartitionedPrefix>
        <PartitionDateSource>DeliveryTime|EventTime</PartitionDateSource>
      </PartitionedPrefix>
      <SimplePrefix/>
    </TargetObjectKeyFormat>
    <TargetPrefix>string</TargetPrefix>
    <LoggingType>Standard|Journal</LoggingType>
    <ObjectRollTime>integer</ObjectRollTime>
    <Filter>
      <S3Key>
        <FilterRule>
          <Name>suffix/prefix/regex</Name>
          <Value></Value>
        </FilterRule>
      </S3Key>
    </Filter>
  </LoggingEnabled>
</BucketLoggingStatus>
```

Request Parameters

Following table lists the request parameters:

Name	Type	Description	Required
BucketLoggingStatus	Container	Enables or disables logging configuration for the bucket.	Yes
LoggingEnabled	Container	Contains the logging configuration details for the bucket.	Yes
TargetBucket	String	Specifies the log bucket where access logs are stored. The log bucket cannot have its own logging enabled.	Yes
TargetGrants	Container	Not supported. The log bucket owner is automatically the owner of the log objects.	No
TargetObjectKeyFormat	Container	Defines the format of the log object key. Can include `PartitionedPrefix` or `SimplePrefix`.	No
PartitionedPrefix	Container	Indicates a partitioned log object key format. The `PartitionDateSource` field is fixed as `DeliveryTime`.	No
SimplePrefix	Container	Indicates a simple (default) log object key format.	No
TargetPrefix	String	Prefix used for log object names. Useful to distinguish logs from multiple source buckets.	No

Name	Type	Description	Required
LoggingType	String	Logging mode. Valid values: `Standard` (default) or `Journal`. Journal mode logs operations before completion.	No
ObjectRollTime	Integer	Time in seconds after which a new log object is created. Default: 3600 (1 hour).	No
Filter	Container	Optional filter for selecting log entries based on object key or metadata.	No

Response entities

The response is XML encoded in the body of the request, only if a configuration change triggers flushing of the current logging object. In this case it will return the name of the flushed logging object in following format:

```
<PostBucketLoggingOutput xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <FlushedLoggingObject>string</FlushedLoggingObject>
</PostBucketLoggingOutput>
```

HTTP Response

The following HTTP response might be returned:

HTTP Response

`400`

Status Code

`MalformedXML`

Description

The XML is not well-formed.

HTTP Response

`400`

Status Code

`InvalidArgument`

Description

Missing mandatory value or invalid value.

HTTP Response

`404`

Status Code

`NoSuchBucket`

Description

The bucket does not exist.

S3 disable bucket logging

Disables bucket logging for a bucket. Use this operation to remove the logging configuration from a source bucket.

Syntax

```
PUT /{bucket}?logging HTTP/1.1
```

Request entities

Parameters are XML encoded in the body of the request, in the following format:

```
<BucketLoggingStatus xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</BucketLoggingStatus>
```

HTTP Response

404

Status Code

NoSuchBucket

Description

The bucket does not exist.

S3 get bucket logging configuration

Retrieves the current bucket logging configuration, including the target bucket, prefix, and logging type.

Syntax

```
GET /{bucket}?logging HTTP/1.1
```

Response entities

The response header contains `Last-Modified` date/time of the logging configuration. Logging configuration is XML encoded in the body of the response, in the following format:

```
<BucketLoggingStatus xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <LoggingEnabled>
    <TargetBucket>string</TargetBucket>
    <TargetGrants>
      <Grant>
        <Grantee>
          <DisplayName>string</DisplayName>
          <EmailAddress>string</EmailAddress>
          <ID>string</ID>
          <xsi:type>string</xsi:type>
          <URI>string</URI>
        </Grantee>
        <Permission>string</Permission>
      </Grant>
    </TargetGrants>
    <TargetObjectKeyFormat>
      <PartitionedPrefix>
        <PartitionDateSource>DeliveryTime|EventTime</PartitionDateSource>
      </PartitionedPrefix>
      <SimplePrefix>
      </SimplePrefix>
    </TargetObjectKeyFormat>
    <TargetPrefix>string</TargetPrefix>
    <LoggingType>Standard|Journal</LoggingType>
    <ObjectRollTime>integer</ObjectRollTime>
    <Filter>
      <S3Key>
        <FilterRule>
          <Name>suffix/prefix/regex</Name>
          <Value></Value>
        </FilterRule>
      </S3Key>
    </Filter>
  </LoggingEnabled>
</BucketLoggingStatus>
```

HTTP Response

404

Status Code

NoSuchBucket

Description

The bucket does not exist.

Using `radosgw-admin` to view bucket logging configuration

To view the same logging configuration that the S3 API returns, use the **`radosgw-admin`** command.

Following is an example for viewing logging configuration on the source bucket

```
[root@ceph14 ~]# radosgw-admin bucket logging info --bucket src-bkt1
{
    "bucketLoggingStatus": {
        "loggingEnabled": {
            "targetBucket": "dest-bkt1",
            "targetPrefix": "src-bkt1-logs-",
            "objectRollTime": 75,
            "loggingType": "Journal",
            "recordsBatchSize": 0,
            "targetObjectKeyFormat": {
                "simplePrefix": {}
            }
        }
    }
}
[root@ceph14 ~]## View logging configuration on the target (destination) bucket
[root@ceph14 ~]# radosgw-admin bucket logging info --bucket dest-bkt1
[
    {
        "name": "src-bkt1",
        "marker": "870dab95-5bb1-42b1-b50a-1a6b4a7c5e77.218451.1",
        "bucket_id": "870dab95-5bb1-42b1-b50a-1a6b4a7c5e77.218451.1",
        "tenant": "",
        "explicit_placement": {
            "data_pool": "",
            "data_extra_pool": "",
            "index_pool": ""
        }
    }
]
[root@ceph14 ~]#
```

Following is an example for view logging configuration on the target (destination) bucket.

```
[root@ceph14 ~]# radosgw-admin bucket logging info --bucket dest-bkt1
[
    {
        "name": "src-bkt1",
        "marker": "870dab95-5bb1-42b1-b50a-1a6b4a7c5e77.218451.1",
        "bucket_id": "870dab95-5bb1-42b1-b50a-1a6b4a7c5e77.218451.1",
        "tenant": "",
        "explicit_placement": {
            "data_pool": "",
            "data_extra_pool": "",
            "index_pool": ""
        }
    }
]
[root@ceph14 ~]#
```

S3 flush bucket logging

Flushes pending logging objects for a bucket. Use this operation to immediately commit log data to the target log bucket. If no log records exist to flush, then the flush operation flushes an empty log object to the target bucket.

Syntax

```
POST /{bucket}?logging HTTP/1.1
```

Response entities

The response body is XML encoded in the following format. It returns the name of the object that was flushed:

```
<PostBucketLoggingOutput xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <FlushedLoggingObject>string</FlushedLoggingObject>
</PostBucketLoggingOutput>
```

HTTP Response

404

Status Code

`NoSuchBucket`

Description

The bucket does not exist.

Using **`radosgw-admin`**
Use the **`radosgw-admin`** command to immediately flush and commit pending logging objects for the bucket.

The following example shows how to flush pending logging objects by using the **`radosgw-admin`** command.

```
# Flush pending logging objects for the source bucket
[root@ceph14 ~]# radosgw-admin bucket logging flush --bucket src-bkt1
flushed pending logging object 'src-bkt1-logs-2025-11-18-20-43-19-0000000011AEI402' to
target bucket 'dest-bkt1'
[root@ceph14 ~]#
```

S3 get the request payment configuration on a bucket

Uses the `requestPayment` subresource to return the request payment configuration of a bucket. The user needs to be the bucket owner or to have been granted `READ_ACP` permission on the bucket.

Add the `requestPayment` subresource to the bucket request.

Syntax

```
GET /BUCKET?requestPayment HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

S3 set the request payment configuration on a bucket

Uses the `requestPayment` subresource to set the request payment configuration of a bucket. By default, the bucket owner pays for downloads from the bucket. This configuration parameter enables the bucket owner to specify that the person requesting the download will be charged for the request and the data download from the bucket.

Add the `requestPayment` subresource to the bucket request.

Syntax

```
PUT /BUCKET?requestPayment HTTP/1.1
Host: cname.domain.com
```

Request Entities

`Payer`

Description

Specifies who pays for the download and request fees.

Type

Enum

`RequestPaymentConfiguration`

Description

A container for `Payer`.

Type

Container

Multi-tenant bucket operations

When a client application accesses buckets, it always operates with the credentials of a particular user. Consequently, every bucket operation has an implicit tenant in its context if no tenant is specified explicitly.

Thus multi-tenancy is completely backward compatible with previous releases, as long as the referred buckets and referring user belong to the same tenant.

Extensions employed to specify an explicit tenant differ according to the protocol and authentication system used.

In the following example, a colon character separates tenant and bucket. Thus a sample URL would be:

```
https://rgw.domain.com/tenant:bucket
```

By contrast, a simple Python example separates the tenant and bucket in the bucket method itself:

Example

```
from boto.s3.connection import S3Connection, OrdinaryCallingFormat
  c = S3Connection(
    aws_access_key_id="TESTER",
    aws_secret_access_key="test123",
    host="rgw.domain.com",
    calling_format = OrdinaryCallingFormat()
  )
  bucket = c.get_bucket("tenant:bucket")
```

Note: It's not possible to use S3-style subdomains using multi-tenancy, since host names cannot contain colons or any other separators that are not already valid in bucket names. Using a period creates an ambiguous syntax. Therefore, the `bucket-in-URL-path` format has to be used with multi-tenancy.

Reference

- For more information, see Multi-tenant namespace.

S3 Block Public Access

You can use the S3 Block Public Access feature to set buckets and users to help you manage public access to Red Hat Ceph Storage object storage S3 resources.

Using this feature, bucket policies, access point policies, and object permissions can be overridden to allow public access. By default, new buckets, access points, and objects do not allow public access.

The S3 API in the Ceph Object Gateway supports a subset of the AWS public access settings:

`BlockPublicPolicy`

This defines the setting to allow users to manage access point and bucket policies. This setting does not allow the users to publicly share the bucket or the objects it contains. Existing access point and bucket policies are not affected by enabling this setting. Setting this option to `TRUE` causes the Ceph Object Gateway:

- To reject calls to PUT Bucket policy.
- To reject calls to PUT access point policy for all of the bucket's same-account access points.

Important: Apply this setting at the user level so that users cannot alter a specific bucket's block public access setting.

Note: The `TRUE` setting only works if the specified policy allows public access.

`RestrictPublicBuckets`

This defines the setting to restrict access to a bucket or access point with public policy.
The restriction applies to only authorized users within the bucket owner's account within the bucket owner's account and access point owner's account. This blocks cross-account access to the access point or bucket, except for the cases specified, while still allowing users within the account to manage the access points or buckets.

Enabling this setting does not affect existing access point or bucket policies. It only defines that Ceph Object Gateway blocks public and cross-account access derived from any public access point or bucket policy, including non-public delegation to specific accounts.

> **Note:** Access control lists (ACLs) are not currently supported by Red Hat Ceph Storage.

`BlockPublicAcls`

When this setting is set to `TRUE`, requests that include public access control lists (ACLs) are blocked. For example, operations such as `PutBucketAcl`, `PutObjectAcl`, and `PutObject` fail if the request specifies a public ACL.
This setting prevents any new ACLs from being created or modified if they would grant public access. It does not affect existing public ACLs, so objects that are already public via an ACL remain public and accessible via `GetObject` requests.

> **Note:** This setting affects only requests that include public ACLs and does not remove or alter existing ACLs on buckets or objects.

`IgnorePublicAcls`

When this setting is set to `TRUE`, all public access control lists (ACLs) applied to buckets and objects are ignored when evaluating access permissions.
This setting is the one that addresses existing public ACLs. When this setting is set to `TRUE`, all public access control lists (ACLs) applied to buckets and objects are ignored while evaluating access permissions.

Existing ACLs remain unchanged and new public ACLs can still be set. This setting prevents public access granted through ACLs by ignoring them.

> **Note:** This setting affects how access permissions are evaluated but does not modify or remove existing ACLs.

Bucket policies are assumed to be public unless defined otherwise. To block public access a bucket policy must give access only to fixed values for one or more of the following:

> **Note:** A fixed value does not contain a wildcard (`*`) or an AWS Identity and Access Management Policy Variable.

- An AWS principal, user, role, or service principal
- A set of Classless Inter-Domain Routings (CIDRs), using `aws:SourceIp`
- `aws:SourceArn`
- `aws:SourceVpc`
- `aws:SourceVpce`
- `aws:SourceOwner`
- `aws:SourceAccount`
- `s3:x-amz-server-side-encryption-aws-kms-key-id`
- `aws:userid` outside the pattern `ARIOLEID:*`
- `s3:DataAccessPointArn`

> **Note:** When used in a bucket policy, this value can contain a wildcard for the access point name without rendering the policy public, as long as the account ID is fixed.

- `s3:DataAccessPointPointAccount`

Example of a policy that is considered public

```
{
        "Principal": "*",
        "Resource": "*",
        "Action": "s3:PutObject",
        "Effect": "Allow",
        "Condition": { "StringLike": {"aws:SourceVpc": "vpc-*"}}
    }
```

To make a policy non-public, include any of the condition keys with a fixed value.

Example of a policy that is considered non-public

```
{
        "Principal": "*",
        "Resource": "*",
        "Action": "s3:PutObject",
        "Effect": "Allow",
        "Condition": {"StringEquals": {"aws:SourceVpc": "vpc-91237329"}}
    }
```

For more information, see the following:

- “S3 GET PublicAccessBlock” on page 188
- “S3 PUT PublicAccessBlock” on page 188
- “S3 delete PublicAccessBlock” on page 189

Related information

Blocking public access to your Amazon S3 storage

Set object ownership when creating a bucket

Set the S3 object ownership mode for a bucket during creation by including the `x-amz-object-ownership` request header in Ceph Object Gateway (RGW).

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- You have credentials with permission to create buckets and configure bucket ownership controls.
- You know the Ceph Object Gateway (RGW) endpoint (for example, `https://<rgw-endpoint>`).

You can specify the ownership mode when creating a bucket by using the `x-amz-object-ownership` request header. For details about ownership modes and their behavior, see S3 object ownership.

To change ownership behavior later, update the bucket’s ownership controls. See Updating bucket ownership controls.

- Create the bucket and include the `x-amz-object-ownership` header in the request to set the ownership mode.
  - Use the following PUT request template to enforce bucket-owner ownership and disable ACLs:

```
PUT /analytics-bucket HTTP/1.1
Host: <rgw-endpoint>
x-amz-object-ownership: BucketOwnerEnforced
Date: <date>
Authorization: <authorization-header>
```

- You can also set other supported ownership modes by changing the `x-amz-object-ownership` header value:

```
x-amz-object-ownership: BucketOwnerPreferred
x-amz-object-ownership: ObjectWriter
```

  Setting `ObjectWriter` explicitly produces the same behavior as omitting the header.

- To use the default ObjectWriter behavior, omit the x-amz-object-ownership request header.

```
PUT /my-default-bucket HTTP/1.1
Host: <rgw-endpoint>
Date: <date>
Authorization: <authorization-header>
```

Result

The bucket is created with the specified ownership mode. All new uploads follow the selected ownership and ACL behavior.

AFTER COMPLETING THE TASK

Verify that the bucket was created and validate ownership behavior.

- Confirm bucket creation (for example, by sending a `HEAD` request).
- Upload a test object and verify that ownership and ACL behavior match the selected mode.
- In BucketOwnerEnforced mode, upload requests that specify ACLs other than `bucket-owner-full-control` fail, and `PutObjectAcl` returns an error.

Updating bucket ownership controls

Update the S3 object ownership mode on an existing bucket by using the bucket ownership controls operations in Ceph Object Gateway (RGW).

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- You have credentials with permission to configure ownership controls on the bucket.
- You know the RGW endpoint (for example, `https://<rgw-endpoint>`).
- The bucket already exists.

Bucket ownership controls define how object ownership is assigned within an existing bucket. You can modify the ownership mode by using the ownership controls APIs.

For details about each ownership mode, see S3 object ownership.

1. Apply the new ownership mode by sending a `PUT` request with an ownership controls configuration.
   Use the following format in the request body:

```
PUT /my-bucket?ownershipControls HTTP/1.1
Host: <rgw-endpoint>
Content-Type: application/xml
Authorization: <authorization-header>
Content-Length: <length>

<OwnershipControls>
  <Rule>
    <ObjectOwnership>BucketOwnerPreferred</ObjectOwnership>
  </Rule>
</OwnershipControls>
```

   Replace `BucketOwnerPreferred` with the ownership mode that you want to set.

2. Verify the ownership mode applied to the bucket.
   Send a `GET` request to retrieve the current ownership controls

```
GET /my-bucket?ownershipControls HTTP/1.1
Host: <rgw-endpoint>
Authorization: <authorization-header>
```

The response contains the current `ObjectOwnership` value.

3. Optional: Remove ownership controls from the bucket.
Return to the default behavior (`ObjectWriter`), by deleting the ownership configuration

```
DELETE /my-bucket?ownershipControls HTTP/1.1
Host: <rgw-endpoint>
Authorization: <authorization-header>
```

Removing the configuration does not change ownership of existing objects. It only affects new uploads.

Result

The configured ownership mode is applied to all new objects uploaded to the bucket. Existing objects retain their current ownership settings.

AFTER COMPLETING THE TASK

Upload a test object to confirm that ownership and ACL behavior follow the updated mode.

For more information about how object ownership affects uploads and ACL operations, see S3 object operations.

Related information

Set object ownership when creating a bucket

S3 object operations

Perform object operations with the Amazon S3 application programming interface (API) through the Ceph Object Gateway.

To use these APIs a running Red Hat Ceph Storage cluster and RESTful client are required.

"S3 functional operations and support status for objects" on page 140 describes the Amazon S3 functional operations for objects, along with the function's support status.

S3 functional operations and support status for objects

Feature	Status
Get Object	Supported
Head object	Supported
Put Object Lock	Supported
Get Object Lock	Supported
Put Object Legal Hold	Supported
Get Object Legal Hold	Supported
Put Object Retention	Supported
Get Object Retention	Supported
Put Object Tagging	Supported
Get Object Tagging	Supported
Delete Object Tagging	Supported
Put Object	Supported
Delete Object	Supported
Delete Multiple Objects	Supported
Get Object ACLs	Supported

Feature	Status
Put Object ACLs	Supported
Copy Object	Supported
Post Object	Supported
Options Object	Supported
Initiate Multipart Upload	Supported
Add a Part to a Multipart Upload	Supported
List Parts of a Multipart Upload	Supported
Assemble Multipart Upload	Supported
Copy Multipart Upload	Supported
Abort Multipart Upload	Supported
Multi-Tenancy	Supported

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A RESTful client.

S3 get an object from a bucket

Use the S3 **GET** command to retrieve an object from a bucket.

Retrieve an object from a bucket.

```
GET /BUCKET/OBJECT HTTP/1.1
```

Add the `versionId` subresource to retrieve a particular version of the object.

```
GET /BUCKET/OBJECT?versionId=VERSION_ID HTTP/1.1
```

Request headers

`partNumber`

Description

Part number of the object being read. This enables a ranged `GET` request for the specified part. Using this request is useful for downloading just a part of an object.

Valid values

A positive integer between 1 and 10,000.

Required

No

`range`

Description

The range of the object to retrieve.

> **Note:** Multiple ranges of data per GET request are not supported.

Valid values

Range:bytes=*beginbyte-endbyte*

Required

No

if-modified-since

Description

Gets only if modified since the timestamp.

Valid values

Timestamp

Required

No

if-unmodified-since

Description

Gets only if not modified since the timestamp.

Valid values

Timestamp

Required

No

if-match

Description

Gets only if object ETag does not match ETag.

Valid values

Entity Tag

Required

No

if-none-match

Description

Gets only if object ETag matches ETag.

Valid values

Entity Tag

Required

No

Example syntax with request headers:

```
GET /BUCKET/OBJECT?partNumber=PARTNUMBER&versionId=VersionId HTTP/1.1
Host: Bucket.s3.amazonaws.com
If-Match: IfMatch
If-Modified-Since: IfModifiedSince
If-None-Match: IfNoneMatch
If-Unmodified-Since: IfUnmodifiedSince
Range: Range
```

Response headers

GET response headers	
Response header	**Description**
`Content-Range`	Data range. Only returned if the range header field was specified in the request.
`x-amz-version-id`	Returns the version ID or null.

Response header	Description
`x-rgw-replicated-from`	Returns the source zone and any intermediate zones involved in an object's replication path within a Ceph multi-zone environment. This header is included in *GetObject* and *HeadObject* responses.
`x-rgw-replicated-at`	Returns a timestamp indicating when the object was replicated to its current location. You can calculate the duration for replication to complete by using this header with *Last-Modified* header.

Note: As of now, `x-rgw-replicated-from` and `x-rgw-replicated-at` are supported by client tools like s3cmd or curl verify at the replicated zone. These tools can be used in addition to *radosgw-admin* command for verification. With *radosgw-admin* object stat we have a known issue [BZ-2312552] of missing header key `x-rgw-replicated-from`.

S3 get object attributes

Use the S3 `GetObjectAttributes` API to retrieve the metadata of an object without returning the object's data.

`GetObjectAttributes` API combines the functionality of `HeadObject` and `ListParts` . It provides all the information returned by these two calls in a single request, streamlining the process and reducing the number of API calls needed.

```
GET /BUCKET/OBJECT?attributes&versionId=VersionId
```

For example,

```
GET /testbucket/testobject?attributes&versionId=testversionid
Host: Bucket.s3.amazonaws.com
x-amz-max-parts: MaxParts
x-amz-part-number-marker: PartNumberMarker
x-amz-server-side-encryption-customer-algorithm: SSECustomerAlgorithm
x-amz-server-side-encryption-customer-key: SSECustomerKey
x-amz-server-side-encryption-customer-key-MD5: SSECustomerKeyMD5
x-amz-request-payer: RequestPayer
x-amz-expected-bucket-owner: ExpectedBucketOwner
x-amz-object-attributes: ObjectAttributes
```

The **`versionId`** subresource retrieves a particular version of the object.

Request entities

For example,

```
GET /testbucket?attributes&versionId=VersionId HTTP/1.1
Host: Bucket.s3.amazonaws.com
x-amz-max-parts: MaxParts
x-amz-part-number-marker: PartNumberMarker
x-amz-server-side-encryption-customer-algorithm: SSECustomerAlgorithm
x-amz-server-side-encryption-customer-key: SSECustomerKey
x-amz-server-side-encryption-customer-key-MD5: SSECustomerKeyMD5
x-amz-request-payer: RequestPayer
x-amz-expected-bucket-owner: ExpectedBucketOwner
x-amz-object-attributes: ObjectAttributes
```

GET request headers

GET response headers			
Request header	**Description**	**Type / Valid values**	**Required?**
`Bucket`	The name of the bucket that contains the object.	String	Yes
`Key`	The object key.	String	Yes

Request header	Description	Type / Valid values	Required?
versionID	The version ID used to reference a specific version of the object.	String	No
x-amz-max-parts	Sets the maximum number of parts to return.	String	No
x-amz-object-attributes	Specifies the fields at the root level that you want returned in the response. Fields that you do not specify are not returned.	ETag | Checksum | ObjectParts | StorageClass | ObjectSize	Yes
x-amz-part-number-marker	Specifies the part after which listing should begin. Only parts with higher part numbers will be listed.	String	No

Response entities

For example,

```
HTTP/1.1 200
x-amz-delete-marker: DeleteMarker
Last-Modified: LastModified
x-amz-version-id: VersionId
x-amz-request-charged: RequestCharged
<?xml version="1.0" encoding="UTF-8"?>
<GetObjectAttributesOutput>
   <ETag>string</ETag>
   <Checksum>
      <ChecksumCRC32>string</ChecksumCRC32>
      <ChecksumCRC32C>string</ChecksumCRC32C>
      <ChecksumSHA1>string</ChecksumSHA1>
      <ChecksumSHA256>string</ChecksumSHA256>
   </Checksum>
   <ObjectParts>
      <IsTruncated>boolean</IsTruncated>
      <MaxParts>integer</MaxParts>
      <NextPartNumberMarker>integer</NextPartNumberMarker>
      <PartNumberMarker>integer</PartNumberMarker>
      <Part>
         <ChecksumCRC32>string</ChecksumCRC32>
         <ChecksumCRC32C>string</ChecksumCRC32C>
         <ChecksumSHA1>string</ChecksumSHA1>
         <ChecksumSHA256>string</ChecksumSHA256>
         <PartNumber>integer</PartNumber>
         <Size>long</Size>
      </Part>
      ...
      <PartsCount>integer</PartsCount>
   </ObjectParts>
   <StorageClass>string</StorageClass>
   <ObjectSize>long</ObjectSize>
</GetObjectAttributesOutput>
```

GET response headers

GET response headers	
Response header	**Description**
last modified	The creation date of the object
x-amz-delete-marker	Specifies whether the object retrieved was (true) or was not (false) a delete marker. If false, this response header does not appear in the response
x-amz-request-charged	If present, indicates that the requester was successfully charged for the request
x-amz-version-id	The version ID of the object.

Response header	Description
`GetObjectAttributesOutput`	Root level tag for the `GetObjectAttributesOutput` parameters.
`Checksum`	The checksum or digest of the object.
`ChecksumCRC32 (string)`	The base64-encoded, 32-bit CRC-32 checksum of the object. This will only be present if it was uploaded with the object. When you use an API operation on an object that was uploaded using multipart uploads, this value may not be a direct checksum value of the full object. Instead, it's a calculation based on the checksum values of each individual part.
`ChecksumCRC32C (string)`	The base64-encoded, 32-bit CRC-32C checksum of the object. This will only be present if it was uploaded with the object. When you use an API operation on an object that was uploaded using multipart uploads, this value may not be a direct checksum value of the full object. Instead, it's a calculation based on the checksum values of each individual part.
`ChecksumSHA1 (string)`	The base64-encoded, 160-bit SHA-1 digest of the object. This will only be present if it was uploaded with the object. When you use the API operation on an object that was uploaded using multipart uploads, this value may not be a direct checksum value of the full object. Instead, it's a calculation based on the checksum values of each individual part.
`ChecksumSHA256 (string)`	The base64-encoded, 256-bit SHA-256 digest of the object. This will only be present if it was uploaded with the object. When you use an API operation on an object that was uploaded using multipart uploads, this value may not be a direct checksum value of the full object. Instead, it's a calculation based on the checksum values of each individual part.
`Etag`	An ETag is an opaque identifier assigned by a web server to a specific version of a resource found at a URL.
`ObjectParts`	A collection of parts associated with a multi-part upload.
`Objectparts (structure)`	A collection of parts associated with a multipart upload.
`TotalPartsCount (integer)`	The total number of parts.
`PartNumberMarker (integer)`	The marker for the current part.
`NextPartNumberMarker (integer)`	When a list is truncated, this element specifies the last part in the list, as well as the value to use for the **`PartNumberMarker`** request parameter in a subsequent request.
`MaxParts (integer)`	The maximum number of parts allowed in the response.
`IsTruncated (boolean)`	Indicates whether the returned list of parts is truncated. A value of true indicates that the list was truncated. A list can be truncated if the number of parts exceeds the limit returned in the MaxParts element.

Response header	Description
`Parts (list)`	A container for elements related to a particular part. A response can contain zero or more Parts elements. **Note:** – General purpose buckets - For **`GetObjectAttributes`** , if an additional checksum (including x-amz-checksum-crc32 , x-amz-checksum-crc32c , x-amz-checksum-sha1 , or x-amz-checksum-sha256) isn't applied to the object specified in the request, the response doesn't return Part. – Directory buckets - For **`GetObjectAttributes`** , no matter whether a additional checksum is applied to the object specified in the request, the response returns Part .
`PartNumber (integer)`	The part number identifying the part. This value is a positive integer between 1 and 10,000.
`Size (long)`	The size of the uploaded part in bytes.
`ObjectSize`	The size of the object in bytes.
`StorageClass`	Provides the storage class information of the object. Amazon S3 returns this header for all objects except for S3 Standard storage class objects.

Retrieve sync replication Headers of object

Returns details about an object. This request returns the same header information as the Get Object request, but includes only the metadata, excluding the object data payload.

Use the following command to retrieve the current version of the object:

```
HEAD /BUCKET/OBJECT HTTP/1.1
```

Add the `versionId` subresource to the command to retrieve information for a particular version.

```
HEAD /BUCKET/OBJECT?versionId=VERSION_ID HTTP/1.1
```

Request headers			
Request header	**Description**	**Valid values**	**Required**
`range`	The range of the object to retrieve.	Range: bytes=beginbyte-endbyte	No
`if-modified-since`	Gets only if modified since the timestamp.	Timestamp	No
`if-match`	Gets only if object ETag matches ETag.	Entity tag	No
`if-none-match`	Gets only if object ETag matches ETag.	Entity tag	No

Response headers

HEAD response headers	
Response header	**Description**
`x-amz-version-id`	Returns the version ID or null.
`x-rgw-replicated-from`	Returns the source zone and any intermediate zones involved in an object's replication path within a Ceph multi-zone environment. This header is included in *GetObject* and *HeadObject* responses.
`x-rgw-replicated-at`	Returns a timestamp indicating when the object was replicated to its current location. You can calculate the duration for replication to complete by using this header with *Last-Modified* header.

Note: As of now, `x-rgw-replicated-from` and `x-rgw-replicated-at` are supported by client tools like s3cmd or curl verify at the replicated zone. These tools can be used in addition to *radosgw-admin* command for verification. With *radosgw-admin* object stat we have a known issue [BZ-2312552] of missing header key `x-rgw-replicated-from`.

S3 put object lock

The put object lock API places a lock configuration on the selected bucket. With object lock, you can store objects using a Write-Once-Read-Many (WORM) model. Object lock ensures an object is not deleted or overwritten, for a fixed amount of time or indefinitely. The rule specified in the object lock configuration is applied by default to every new object placed in the selected bucket.

For more information about the WORM model, see Ceph Object Gateway.

Note: Enable the object lock when creating a bucket otherwise, the operation fails.

Syntax

```
PUT /BUCKET?object-lock HTTP/1.1
```

Example

```
PUT /testbucket?object-lock HTTP/1.1
```

Request Entities

`ObjectLockConfiguration`

Description

A container for the request.

Type

Container

Required

Yes

`ObjectLockEnabled`

Description

Indicates whether this bucket has an object lock configuration enabled.

Type

String

Required

Yes

`Rule`

Description

The object lock rule in place for the specified bucket.

Type

Container

Required

No

`DefaultRetention`

Description

The default retention period applied to new objects placed in the specified bucket.

Type

Container

Required

No

`Mode`

Description

The default object lock retention mode. Valid values: GOVERNANCE/COMPLIANCE.

Type

Container

Required

Yes

`Days`

Description

The number of days specified for the default retention period.

Type

Integer

Required

No

`Years`

Description

The number of years specified for the default retention period.

Type

Integer

Required

No

HTTP Response

`400`

Status Code

`MalformedXML`

Description

The XML is not well-formed.

`409`

Status Code

`InvalidBucketState`

Description

The bucket object lock is not enabled.

Reference

– For more information about this API call, see S3 API.

S3 get object lock

The get object lock API retrieves the lock configuration for a bucket.

Syntax

```
GET /BUCKET?object-lock HTTP/1.1
```

Example

```
GET /testbucket?object-lock HTTP/1.1
```

Response Entities

`ObjectLockConfiguration`

Description

A container for the request.

Type

Container

Required

Yes

`ObjectLockEnabled`

Description

Indicates whether this bucket has an object lock configuration enabled.

Type

String

Required

Yes

`Rule`

Description

The object lock rule is in place for the specified bucket.

Type

Container

Required

No

`DefaultRetention`

Description

The default retention period applied to new objects placed in the specified bucket.

Type

Container

Required

No

`Mode`

Description

The default object lock retention mode. Valid values: GOVERNANCE/COMPLIANCE.

Type

Container

Required

Yes

`Days`

Description

The number of days specified for the default retention period.

Type

Integer

Required

No

`Years`

Description

The number of years specified for the default retention period.

Type

Integer

Required

No

Reference

– For more information about this API call, see S3 API.

S3 put object legal hold

The put object legal hold API applies a legal hold configuration to the selected object. With a legal hold in place, you cannot overwrite or delete an object version. A legal hold does not have an associated retention period and remains in place until you explicitly remove it.

Syntax

```
PUT /BUCKET/OBJECT?legal-hold&versionId= HTTP/1.1
```

Example

```
PUT /testbucket/testobject?legal-hold&versionId= HTTP/1.1
```

The `versionId` subresource retrieves a particular version of the object.

Request Entities

`LegalHold`

Description

A container for the request.

Type

Container

Required

Yes

`Status`

Description

Indicates whether the specified object has a legal hold in place. Valid values: ON/OFF

Type

String

Required

Yes

Reference

– For more information about this API call, see S3 API.

S3 get object legal hold

The get object legal hold API retrieves an object's current legal hold status.

Syntax

```
GET /BUCKET/OBJECT?legal-hold&versionId= HTTP/1.1
```

Example

```
GET /testbucket/testobject?legal-hold&versionId= HTTP/1.1
```

The `versionId` subresource retrieves a particular version of the object.

Response Entities

`LegalHold`

Description

A container for the request.

Type

Container

Required

Yes

`Status`

Description

Indicates whether the specified object has a legal hold in place. Valid values: ON/OFF

Type

String

Required

Yes

Reference

- For more information about this API call, see S3 API.

S3 put object retention

The put object retention API places an object retention configuration on an object. A retention period protects an object version for a fixed amount of time. There are two modes: GOVERNANCE and COMPLIANCE. These two retention modes apply different levels of protection to your objects.

> **Note:** During this period, your object is Write-Once-Read-Many (WORM-protected) and cannot be overwritten or deleted. For more information about the WORM model, see Ceph Object Gateway.

Syntax

```
PUT /BUCKET/OBJECT?retention&versionId= HTTP/1.1
```

Example

```
PUT /testbucket/testobject?retention&versionId= HTTP/1.1
```

The `versionId` sub-resource retrieves a particular version of the object.

Request Entities

`Retention`

Description

A container for the request.

Type

Container

Required

Yes

`Mode`

Description

Retention mode for the specified object.

Valid values: GOVERNANCE, COMPLIANCE.

Type

String

Required

Yes

`RetainUntilDate`

Description

Retention date.

Format

2020-01-05T00:00:00.000Z

Type

Timestamp

Required

Yes

Reference

– For more information about this API call, see S3 API.

S3 get object retention

The get object retention API retrieves an object retention configuration on an object.

Syntax

```
GET /BUCKET/OBJECT?retention&versionId= HTTP/1.1
```

Example

```
GET /testbucket/testobject?retention&versionId= HTTP/1.1
```

The `versionId` subresource retrieves a particular version of the object.

Response Entities

`Retention`

Description

A container for the request.

Type

Container

Required

Yes

`Mode`

Description

Retention mode for the specified object. Valid values: GOVERNANCE/COMPLIANCE

Type

String

Required

Yes

`RetainUntilDate`

Description

Retention date. Format: 2020-01-05T00:00:00.000Z

Type

Timestamp

Required

Yes

Reference

– For more information about this API call, see S3 API.

S3 put object tagging

The put object tagging API associates tags with an object. A tag is a key-value pair. To put tags of any other version, use the `versionId` query parameter. You must have permission to perform the `s3:PutObjectTagging` action. By default, the bucket owner has this permission and can grant this permission to others.

Syntax

```
PUT /BUCKET/OBJECT?tagging&versionId= HTTP/1.1
```

Example

```
PUT /testbucket/testobject?tagging&versionId= HTTP/1.1
```

Request Entities

`Tagging`

Description

A container for the request.

Type

Container

Required

Yes

`TagSet`

Description

A collection of a set of tags.

Type

String

Required

Yes

Reference

– For more information about this API call, see S3 API.

S3 get object tagging

The get object tagging API returns the tag of an object. By default, the `GET` operation returns information on the current version of an object.

> **Note:** For a versioned bucket, you can have multiple versions of an object in your bucket. To retrieve tags of any other version, add the `versionId` query parameter in the request.

Syntax

```
GET /BUCKET/OBJECT?tagging&versionId= HTTP/1.1
```

Example

```
GET /testbucket/testobject?tagging&versionId= HTTP/1.1
```

Reference

– For more information about this API call, see S3 API.

S3 delete object tagging

The delete object tagging API removes the entire tag set from the specified object. You must have permission to perform the `s3:DeleteObjectTagging` action, to use this operation.

Note: To delete tags of a specific object version, add the `versionId` query parameter in the request.

Syntax

```
DELETE /_BUCKET_/_OBJECT_?tagging&versionId= HTTP/1.1
```

Example

```
DELETE /testbucket/testobject?tagging&versionId= HTTP/1.1
```

Reference

– For more information about this API call, see S3 API.

S3 add an object to a bucket

Adds an object to a bucket. You must have write permissions on the bucket to perform this operation.

Syntax

```
PUT /_BUCKET_/_OBJECT_ HTTP/1.1
```

Request Headers

`content-md5`

Description

A base64 encoded MD-5 hash of the message.

Valid Values

A string. No defaults or constraints.

Required

No

`content-type`

Description

A standard MIME type.

Valid Values

Any MIME type. Default: `binary/octet-stream`.

Required

No

`x-amz-meta-<...>*`

Description

User metadata. Stored with the object.

Valid Values

A string up to 8kb. No defaults.

Required

No

`x-amz-acl`

Description

A canned ACL.

Valid Values

`private`, `public-read`, `public-read-write`, `authenticated-read`

Required

No

`if-match`

Description

Uploads the object only if the current object's ETag matches the specified ETag value. If the ETag does not match, the request fails with `HTTP 412 (Precondition Failed)`. If multiple requests attempt to modify the same object concurrently, the server can return `HTTP 409 (Conflict)`.

Valid Values

`Entity Tag (ETag)`

Required

No

if-none-match

Uploads the object only if the specified ETag does not match the current object's ETag. If the ETag matches, the request fails with `HTTP 412 (Precondition Failed)`.

Valid Values

`Entity Tag (ETag)`

Required

No

Response Headers

`x-amz-version-id`

Description

Returns the version ID or null.

> **Note:** Conditional headers are not supported for multipart upload initialization requests.

S3 delete an object

Removes an object. Requires WRITE permission set on the containing bucket. If bucket versioning is enabled, the delete operation creates a delete marker. To permanently delete a specific version, you must specify the `versionId` subresource.

Syntax

Delete the latest version of an object:

```
DELETE /BUCKET/OBJECT HTTP/1.1
```

Delete a specific version of an object:

Syntax

```
DELETE /BUCKET/OBJECT?versionId=VERSION_ID HTTP/1.1
```

Conditional delete

You can perform a conditional delete so that the object is deleted only if certain metadata conditions are met. This helps prevent accidental deletion if the object has changed since it was last read.

Red Hat Ceph Storage supports conditional delete operations uniformly for both general purpose buckets and directory buckets. This behavior applies to all supported conditional delete options in Ceph.

The following table lists the supported conditional delete options

Option	Description	Supported bucket type
if-match <ETag>	Deletes the object only if the specified ETag matches the current ETag of the object. Returns `412 Precondition Failed` if it does not match.	General purpose and directory buckets.
if-match-last-modified-time <timestamp>	Deletes the object only if its LastModifiedTime matches the specified timestamp. Returns `412 Precondition Failed` if it does not match.	Directory buckets only.
if-match-size <bytes>	Deletes the object only if the object size matches the specified size in bytes. Returns `412 Precondition Failed` if it does not match.	Directory buckets only.

Note: You can use these conditional parameters individually or in combination. If the condition fails, the server returns HTTP status code `412 Precondition Failed`.

Examples

Example 1: Delete an object unconditionally.

```
aws s3api delete-object \
  --bucket my-example-bucket \
  --key logs/2025/system.log
```

Example 2: Delete a specific version of an object.

```
aws s3api delete-object \
  --bucket my-versioned-bucket \
  --key data/file.txt \
  --version-id 3HL4kqCxf3vjVBH40NrjfkdXh6xYzl7l
```

Example 3: Conditional delete using ETag.

```
aws s3api delete-object \
  --bucket my-dir-bucket \
  --key reports/january.csv \
  --if-match "b1946ac92492d2347c6235b4d2611184"
```

Example 4: Conditional delete using size and last modified time (directory bucket only).

```
aws s3api delete-object \
  --bucket amzn-s3-demo-bucket-usw2-az1-x-s3 \
  --key analytics/summary.json \
  --if-match-last-modified-time 2025-09-20T10:25:00Z \
  --if-match-size 2048
```

S3 delete multiple objects

Deletes multiple objects from a bucket in a single request. Each object entry can include optional metadata fields to perform conditional deletes.

Syntax

Conditional delete

When deleting multiple objects, you can include optional metadata for each object to ensure that deletion occurs only if conditions are met.

ETag

Deletes the object only if the current ETag matches the specified value.

LastModifiedTime

Deletes the object only if the last modified time matches the specified timestamp.

Size

Deletes the object only if the size matches the specified size in bytes.

> **Note:** If any condition fails, that object entry is skipped, and the response includes the error for that object. Other objects in the batch are still processed.

Examples

Example 1: Delete multiple objects. Deletes three objects in one request.

```
aws s3api delete-objects \
  --bucket my-batch-bucket \
  --delete '{
    "Objects": [
      {"Key": "images/photo1.jpg"},
      {"Key": "images/photo2.jpg"},
      {"Key": "images/photo3.jpg"}
    ],
    "Quiet": false
  }'
```

Example 2: Conditional delete of multiple objects using ETag, LastModifiedTime, and Size Deletes each object only if the specified metadata matches.

```
aws s3api delete-objects \
  --bucket my-dir-bucket-usw2-az1-x-s3 \
  --delete '{
    "Objects": [
      {
        "Key": "logs/2025/january.log",
        "ETag": "b1946ac92492d2347c6235b4d2611184",
        "LastModifiedTime": "2025-09-22T14:12:00Z",
        "Size": 4096
      },
      {
        "Key": "logs/2025/february.log",
        "ETag": "5d41402abc4b2a76b9719d911017c592"
      }
    ],
    "Quiet": false
  }'
```

S3 get an object's Access Control List (ACL)

Returns the ACL for the current version of the object:

Syntax

```
GET /BUCKET/OBJECT?acl HTTP/1.1
```

Add the `versionId` subresource to retrieve the ACL for a particular version:

Syntax

```
GET /BUCKET/OBJECT?versionId=VERSION_ID&acl HTTP/1.1
```

Response Headers

`x-amz-version-id`

Description

Returns the version ID or null.

Response Entities

`AccessControlPolicy`

Description

A container for the response.

Type

Container

`AccessControlList`

Description

A container for the ACL information.

Type

Container

`Owner`

Description

A container for the bucket owner's `ID` and `DisplayName`.

Type

Container

`ID`

Description

The bucket owner's ID.

Type

String

`DisplayName`

Description

The bucket owner's display name.

Type

String

`Grant`

Description

A container for `Grantee` and `Permission`.

Type

Container

`Grantee`

Description

A container for the `DisplayName` and `ID` of the user receiving a grant of permission.

Type

Container

`Permission`

Description

The permission given to the `Grantee` bucket.

Type

String

S3 set an object's Access Control List (ACL)

Sets an object ACL for the current version of the object.

Syntax

```
PUT /BUCKET/OBJECT?acl
```

Request Entities

`AccessControlPolicy`

Description

A container for the response.

Type

Container

`AccessControlList`

Description

A container for the ACL information.

Type

Container

`Owner`

Description

A container for the bucket owner's `ID` and `DisplayName`.

Type

Container

`ID`

Description

The bucket owner's ID.

Type

String

`DisplayName`

Description

The bucket owner's display name.

Type

String

`Grant`

Description

A container for `Grantee` and `Permission`.

Type

Container

`Grantee`

Description

A container for the `DisplayName` and `ID` of the user receiving a grant of permission.

Type

Container

`Permission`

Description

The permission given to the `Grantee` bucket.

Type

String

S3 copy an object

To copy an object, use `PUT` and specify a destination bucket and the object name.

Syntax

```
PUT /_DEST_BUCKET_/_DEST_OBJECT_ HTTP/1.1
x-amz-copy-source: _SOURCE_BUCKET_/_SOURCE_OBJECT_
```

Request Headers

`x-amz-copy-source`

Description

The source bucket name + object name.

Valid Values

`BUCKET_/_OBJECT`

Required

Yes

`x-amz-acl`

Description

A canned ACL.

Valid Values

`private`, `public-read`, `public-read-write`, `authenticated-read`

Required

No

`x-amz-copy-if-modified-since`

Description

Copies only if modified since the timestamp.

Valid Values

Timestamp

Required

No

`x-amz-copy-if-unmodified-since`

Description

Copies only if unmodified since the timestamp.

Valid Values

Timestamp

Required

No

`x-amz-copy-if-match`

Description

Copies only if object ETag matches ETag.

Valid Values

Entity Tag

Required

No

`x-amz-copy-if-none-match`

Description

Copies only if object ETag matches ETag.

Valid Values

Entity Tag

Required

No

Response Entities

`CopyObjectResult`

Description

A container for the response elements.

Type

Container

`LastModified`

Description

The last modified date of the source object.

Type

Date

`Etag`

Description

The ETag of the new object.

Type

String

S3 add an object to a bucket using HTML forms

Adds an object to a bucket using HTML forms. You must have write permissions on the bucket to perform this operation.

Syntax

```
POST _BUCKET_/_OBJECT_ HTTP/1.1
```

S3 determine options for a request

A preflight request to determine if an actual request can be sent with the specific origin, HTTP method, and headers.

Syntax

```
OPTIONS /_OBJECT_ HTTP/1.1
```

S3 initiate a multipart upload

Initiates a multi-part upload process. Returns a `UploadId`, which you can specify when adding additional parts, listing parts, and completing or abandoning a multi-part upload.

Syntax

```
POST /BUCKET/OBJECT?uploads
```

Request Headers

`content-md5`

Description

A base64 encoded MD-5 hash of the message.

Valid Values

A string. No defaults or constraints.

Required

No

`content-type`

Description

A standard MIME type.

Valid Values

Any MIME type. Default: `binary/octet-stream`

Required

No

`x-amz-meta-<...>`

Description

User metadata. Stored with the object.

Valid Values

A string up to 8kb. No defaults.

Required

No

`x-amz-acl`

Description

A canned ACL.

Valid Values

`private`, `public-read`, `public-read-write`, `authenticated-read`

Required

No

Response Entities

`InitiatedMultipartUploadsResult`

Description

A container for the results.

Type

Container

`Bucket`

Description

The bucket that will receive the object contents.

Type

String

`Key`

Description

The key specified by the `key` request parameter, if any.

Type

String

`UploadId`

Description

The ID specified by the `upload-id` request parameter identifying the multipart upload, if any.

Type

String

S3 add a part to a multipart upload

Adds a part to a multi-part upload.

Specify the `uploadId` subresource and the upload ID to add a part to a multi-part upload:

Syntax

```
PUT /BUCKET/OBJECT?partNumber=&uploadId=UPLOAD_ID HTTP/1.1
```

The following HTTP response might be returned:

HTTP Response

`404`

Status Code

`NoSuchUpload`

Description

Specified upload-id does not match any initiated upload on this object.

S3 list the parts of a multipart upload

Specify the `uploadId` subresource and the upload ID to list the parts of a multi-part upload:

Syntax

```
GET /BUCKET/OBJECT?uploadId=UPLOAD_ID HTTP/1.1
```

Response Entities

`InitiatedMultipartUploadsResult`

Description

A container for the results.

Type

Container

`Bucket`

Description

The bucket that will receive the object contents.

Type

String

`Key`

Description

The key specified by the `key` request parameter, if any.

Type

String

`UploadId`

Description

The ID specified by the `upload-id` request parameter identifying the multipart upload, if any.

Type

String

`Initiator`

Description

Contains the `ID` and `DisplayName` of the user who initiated the upload.

Type

Container

`ID`

Description

The initiator's ID.

Type

String

`DisplayName`

Description

The initiator's display name.

Type

String

`Owner`

Description

A container for the `ID` and `DisplayName` of the user who owns the uploaded object.

Type

Container

`StorageClass`

Description

The method used to store the resulting object. `STANDARD` or `REDUCED_REDUNDANCY`

Type

String

`PartNumberMarker`

Description

The part marker to use in a subsequent request if `IsTruncated` is `true`. Precedes the list.

Type

String

`NextPartNumberMarker`

Description

The next part marker to use in a subsequent request if `IsTruncated` is `true`. The end of the list.

Type

String

`IsTruncated`

Description

If `true`, only a subset of the object's upload contents were returned.

Type

Boolean

`Part`

Description

A container for `Key`, `Part`, `InitiatorOwner`, `StorageClass`, and `Initiated` elements.

Type

Container

`PartNumber`

Description

A container for `Key`, `Part`, `InitiatorOwner`, `StorageClass`, and `Initiated` elements.

Type

Integer

`ETag`

Description

The part's entity tag.

Type

String

`Size`

Description

The size of the uploaded part.

Type

Integer

S3 assemble the uploaded parts

Assembles uploaded parts and creates a new object, thereby completing a multipart upload.

Specify the `uploadId` subresource and the upload ID to complete a multi-part upload:

Syntax

```
POST /BUCKET/OBJECT?uploadId=UPLOAD_ID HTTP/1.1
```

Request Entities

`CompleteMultipartUpload`

Description

A container consisting of one or more parts.

Type

Container

Required

Yes

`Part`

Description

A container for the `PartNumber` and `ETag`.

Type

Container

Required

Yes

`PartNumber`

Description

The identifier of the part.

Type

Integer

Required

Yes

`ETag`

Description

The part's entity tag.

Type

String

Required

Yes

Response Entities

`CompleteMultipartUploadResult`

Description

A container for the response.

Type

Container

`Location`

Description

The resource identifier (path) of the new object.

Type

URI

`bucket`

Description

The name of the bucket that contains the new object.

Type

String

`Key`

Description

The object's key.

Type

String

`ETag`

Description

The entity tag of the new object.

Type

String

S3 copy a multipart upload

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

Specify the `uploadId` subresource and the upload ID to perform a multi-part upload copy:

Syntax

```
PUT /_BUCKET_/_OBJECT_?partNumber=PartNumber&uploadId=_UPLOADID HTTP/1.1
Host: cname.domain.com

Authorization: AWS _ACCESS_KEY_:_HASH_OF_HEADER_AND_SECRET_
```

Request Headers

`x-amz-copy-source`

Description

The source bucket name and object name.

Valid Values

BUCKET/*OBJECT*

Required

Yes

`x-amz-copy-source-range`

Description

The range of bytes to copy from the source object.

Valid Values

Range: `bytes=first-last`, where the first and last are the zero-based byte offsets to copy. For example,`bytes=0-9` indicates that you want to copy the first ten bytes of the source.

Required

No

Response Entities

`CopyPartResult`

Description

A container for all response elements.

Type

Container

`ETag`

Description

Returns the ETag of the new part.

Type

String

`LastModified`

Description

Returns the date the part was last modified.

Type

String

Reference

– For more information about this feature, see the *Amazon S3 site*.

S3 abort a multipart upload

Aborts a multipart upload.

Specify the `uploadId` subresource and the upload ID to abort a multi-part upload:

Syntax

```
DELETE /BUCKET/OBJECT?uploadId=UPLOAD_ID HTTP/1.1
```

S3 Hadoop interoperability

For data analytics applications that require Hadoop Distributed File System (HDFS) access, the Ceph Object Gateway can be accessed using the Apache S3A connector for Hadoop. The S3A connector is an open-source tool that presents S3 compatible object storage as an HDFS file system with HDFS file system read and write semantics to the applications while data is stored in the Ceph Object Gateway.

Ceph Object Gateway is fully compatible with the S3A connector that ships with Hadoop 2.7.3.

S3 select operations

As a developer, you can use the S3 select API for high-level analytic applications like Spark-SQL to improve latency and throughput.

There are three S3 select workflow - CSV, Apache Parquet (Parquet), and JSON that provide S3 select operations with CSV, Parquet, and JSON objects:

– A CSV file stores tabular data in plain text format. Each line of the file is a data record.

– Parquet is an open-source, column-oriented data file format designed for efficient data storage and retrieval. It provides highly efficient data compression and encoding schemes with enhanced performance to handle complex data in bulk. It can be used for single, multi, and archive site. Parquet enables the S3 select-engine to skip columns and chunks, thereby reducing IOPS dramatically (contrary to CSV and JSON format).

– JSON uses SQL statements to scan and extract information from JSON documents. It can be nested in various ways, such as within objects or arrays. These objects and arrays can be further nested within each other without any limitations.

– JSON is a format structure. The S3 select engine enables the use of SQL statements on top of the JSON format input data using the JSON reader, enabling the scanning of highly nested and complex JSON formatted data.

For example, a CSV, Parquet, or JSON S3 object with several gigabytes of data allows the user to extract a single column which is filtered by another column using the following query:

```
select customerid from s3Object where age>30 and age<65;
```

Currently, the S3 object must retrieve data from the Ceph OSD through the Ceph Object Gateway before filtering and extracting data. There is improved performance when the object is large and the query is more specific.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A RESTful client.
- A S3 user created with user access.

S3 select content from an object

The select object content API filters the content of an object through the structured query language (SQL).

Requests

In the request, you must specify the data serialization format as, comma-separated values (CSV) of the object to retrieve the specified content. Amazon Web Services (AWS) command-line interface (CLI) select object content uses the CSV format to parse object data into records and returns only the records specified in the query.

For more information and an example of the description of what should reside in the inventory object, see **Metadata collected by Inventory**, within the AWS Systems Manager User Guide.

> **Note:** You must specify the data serialization format for the response. You must have `s3:GetObject` permission for this operation.

```
POST /BUCKET/KEY?select&select-type=2 HTTP/1.1\r\n
```

For example,

```
POST /testbucket/sample1csv?select&select-type=2 HTTP/1.1\r\n
```

Request entities			
Request	**Description**	**Type**	**Required**
`Bucket`	The bucket to select object content from.	String	Yes
`Key`	The object key. Required minimum length of 1.	String	Yes
`SelectObjectContentRequest`	Root level tag for the select object content request parameters.	String	Yes
`Expression`	The expression that is used to query the object.	String	Yes
`ExpressionType`	The type of the provided expression for example SQL. Valid value: `SQL`	String	Yes
`InputSerialization`	Describes the format of the data in the object that is being queried.	String	Yes
`OutputSerialization`	Format of data returned in comma separator and new-line.	String	Yes

Responses

If the action is successful, the service sends back `HTTP 200` response. Data is returned in XML format by the service:

Response entities			
Response	**Description**	**Type**	**Required**
`Payload`	Root level tag for the payload parameters.	String	Yes
`Records`	The records event.	Base64-encoded binary data object	No
`Stats`	The stats event.	Long	No

Ceph Object Gateway supports the following response type:

```
{:event-type,records} {:content-type,application/octet-stream} {:message-type,event}
```

CSV requests

```
aws --endpoint-URL http://localhost:80 s3api select-object-content
 --bucket BUCKET NAME
 --expression-type SQL
 --input-serialization
 {"CSV": {"FieldDelimiter": "," , "QuoteCharacter": "\"" , "RecordDelimiter" : "\n" ,
"QuoteEscapeCharacter" : "\\" , "FileHeaderInfo": "USE" }, "CompressionType": "NONE"}
 --output-serialization {"CSV": {}}
 --key OBJECT_NAME.csv
 --expression "select count(0) from s3object where int(_1)<10;" output.csv
```

Note: The object names in `--key OBJECT_NAME` do not require `.csv`, `.json`, or `.parquet`.

For example,

```
aws --endpoint-url http://localhost:80 s3api select-object-content
 --bucket testbucket
 --expression-type 'SQL'
 --input-serialization
 '{"CSV": {"FieldDelimiter": "," , "QuoteCharacter": "\"" , "RecordDelimiter" : "\n" ,
"QuoteEscapeCharacter" : "\\" , "FileHeaderInfo": "USE" }, "CompressionType": "NONE"}'
 --output-serialization '{"CSV": {}}'
 --key testobject.csv
 --expression "select count(0) from s3object where int(_1)<10;" output.csv
```

Parquet requests

Note: The only supported output serialization is `.csv`.

```
aws --endpoint-url http://localhost:80 s3api select-object-content
 --bucket BUCKET NAME
 --expression-type 'SQL'
 --input-serialization
 '{"Parquet": {}, {"CompressionType": "NONE"}'
 --output-serialization '{"CSV": {}}}'
 --key OBJECT_NAME.parquet
 --expression "select count(0) from s3object where int(_1)<10;" output.csv
```

For example,

```
aws --endpoint-url http://localhost:80 s3api select-object-content
 --bucket testbucket
 --expression-type 'SQL'
 --input-serialization
 '{"Parquet":{}, "CompressionType": "NONE"}'
 --output-serialization '{"CSV": {}}'
 --key testobject.parquet
 --expression "select count(0) from s3object where int(_1)<10;" output.csv
```

JSON requests

```
aws --endpoint-url http://localhost:80 s3api select-object-content
 --bucket BUCKET_NAME
 --key OBJECT_NAME.json
 --expression-type 'SQL'
 --input-serialization
 '{"JSON": {"Type": "DOCUMENT"}, "CompressionType": "NONE"}'
 --output-serialization '{"CSV": {}}'
 --expression "select count(*) from S3Object[*];" /dev/stdout
```

For example,

```
aws s3api --endpoint-url http://localhost:80
select-object-content --bucket test
--key testobject.json
--expression-type 'SQL'
--input-serialization
'{"JSON": {"Type": "DOCUMENT"}, "CompressionType": "NONE"}'
--output-serialization '{"CSV": {}}'
--expression "select count(*) from S3Object[*];" /dev/stdout
```

BOTO3 requests

```
import pprint
import boto3
from botocore.exceptions import ClientError

def run_s3select(bucket,key,query,column_delim=",",row_delim="\n",quot_char='"',esc_char='\
\',csv_header_info="NONE"):

   s3 = boto3.client('s3',
       endpoint_url=endpoint,
       aws_access_key_id=access_key,
       region_name=region_name,
       aws_secret_access_key=secret_key)

   result = ""
   try:
       r = s3.select_object_content(
       Bucket=bucket,
       Key=key,
       ExpressionType='SQL',
       InputSerialization = {"CSV": {"RecordDelimiter" : row_delim, "FieldDelimiter" :
column_delim,"QuoteEscapeCharacter": esc_char, "QuoteCharacter": quot_char,
"FileHeaderInfo": csv_header_info}, "CompressionType": "NONE"},
       OutputSerialization = {"CSV": {}},
       Expression=query,
       RequestProgress = {"Enabled": progress})

   except ClientError as c:
       result += str(c)
       return result

   for event in r['Payload']:
            if 'Records' in event:
                result = ""
                records = event['Records']['Payload'].decode('utf-8')
                result += records
            if 'Progress' in event:
                print("progress")
                pprint.pprint(event['Progress'],width=1)
            if 'Stats' in event:
                print("Stats")
                pprint.pprint(event['Stats'],width=1)
            if 'End' in event:
                print("End")
                pprint.pprint(event['End'],width=1)
```

```
   return result



 run_s3select(
 "my_bucket",
 "my_csv_object",
 "select int(_1) as a1, int(_2) as a2 , (a1+a2) as a3 from s3object where a3>100 and
a3<300;")
```

Supported features

Currently, only part of the AWS s3 select command is supported. For a full list of supported features, see "AWS s3 select command supported features" on page 173.

AWS s3 select command supported features

Features	Details	Description	Example
Arithmetic operators	`^ * % / + - ( )`	N/A	`select (int(_1)+int(_2))*int(_9) from s3object;`
Arithmetic operators	`% modulo`	N/A	`select count(*) from s3object where cast(_1 as int)%2 = 0;`
Arithmetic operators	`^ power-of`	N/A	`select cast(2^10 as int) from s3object;`
Compare operators	`> < >= <= == !=`	N/A	`select _1,_2 from s3object where (int(_1)+int(_3))>int(_5);`
logical operator	`AND OR NOT`	N/A	`select count(*) from s3object where not (int(1)>123 and int(_5)<200);`
logical operator	`is null`	Returns true/false for null indication in expression	N/A
logical operator and NULL	`is not null`	Returns true/false for null indication in expression	N/A
logical operator and NULL	unknown state	Review null-handle and observe the results of logical operations with NULL. The query returns `0`.	`select count(*) from s3object where null and (3>2);`
Arithmetic operator with NULL	unknown state	Review null-handle and observe the results of binary operations with NULL. The query returns `0`.	`select count(*) from s3object where (null+1) and (3>2);`
Compare with NULL	unknown state	Review null-handle and observe results of compare operations with NULL. The query returns `0`.	`select count(*) from s3object where (null*1.5) != 3;`
missing column	unknown state	N/A	`select count(*) from s3object where _1 is null;`

Features	Details	Description	Example
projection column	Similar to if or then or else	N/A	`select case when (1+1==(2+1)*3) then 'case_1' when ((4*3)==(12)) then 'case_2' else 'case_else' end, age*2 from s3object;`
projection column	Similar to switch/case default	N/A	`select case cast(_1 as int) + 1 when 2 then “a” when 3 then “b” else “c” end from s3object;`
logical operator	N/A	`coalesce` returns first non-null argument	`select coalesce(nullif(5,5), nullif(1,1.0),age+12) from s3object;`
logical operator	N/A	`nullif` returns null in case both arguments are equal, or else the first one,`nullif(1,1)=NULL nullif(null,1)=NULL nullif(2,1)=2`	`select nullif(cast(_1 as int),cast(_2 as int)) from s3object;`
logical operator	N/A	`{expression} in ( .. {expression} ..)`	`select count(*) from s3object where 'ben' in (trim(_5),substring(_1,char_length(_1)-3,3),last_name);`
logical operator	N/A	`{expression} between {expression} and {expression}`	`select _1 from s3object where cast(_1 as int) between 800 and 900;; select count(*) from stdin where substring(_3,char_length(_3),1) between “x” and trim(_1) and substring(_3,char_length(_3)-1,1) = “:”;`
logical operator	N/A	`{expression} like {match-pattern}`	`select count(*) from s3object where first_name like '%de_'; select count(*) from s3object where _1 like "%a[r-s];`
casting operator	N/A	N/A	`select cast(123 as int)%2 from s3object;`
casting operator	N/A	N/A	`select cast(123.456 as float)%2 from s3object;`
casting operator	N/A	N/A	`select cast('ABC0-9' as string),cast(substr('ab12cd',3,2) as int)*4 from s3object;`

Features	Details	Description	Example
casting operator	N/A	N/A	`select cast(substring('publish on 2007-01-01',12,10) as timestamp) from s3object;`
non AWS casting operator	N/A	N/A	`select int(_1),int( 1.2 + 3.4) from s3object;`
non AWS casting operator	N/A	N/A	`select float(1.2) from s3object;`
non AWS casting operator	N/A	N/A	`select to_timestamp('1999-10-10T12:23:44Z') from s3object;`
Aggregation Function	`sun`	N/A	`select sum(int(_1)) from s3object;`
Aggregation Function	`avg`	N/A	`select avg(cast(_1 as float) + cast(_2 as int)) from s3object;`
Aggregation Function	`min`	N/A	`select avg(cast(_1 a float) + cast(_2 as int)) from s3object;`
Aggregation Function	`max`	N/A	`select max(float(_1)),min(int(_5)) from s3object;`
Aggregation Function	`count`	N/A	`select count(*) from s3object where (int(1)+int(_3))>int(_5);`
Timestamp Functions	`extract`	N/A	`select count(*) from s3object where extract(year from to_timestamp(_2)) > 1950 and extract(year from to_timestamp(_1)) < 1960;`
Timestamp Functions	`dateadd`	N/A	`select count(0) from s3object where date_diff(year,to_timestamp(_1),date_add(day,366,to_timestamp(_1))) = 1;`
Timestamp Functions	`datediff`	N/A	`select count(0) from s3object where date_diff(month,to_timestamp(_1),to_timestamp(_2)) = 2;`
Timestamp Functions	`utcnow`	N/A	`select count(0) from s3object where date_diff(hour,utcnow(),date_add(day,1,utcnow())) = 24`

Features	Details	Description	Example
Timestamp Functions	`to_string`	N/A	`select to_string( to_timestamp("2009-09-17T17:56:06.234567Z"), "yyyyMMdd-H:m:s") from s3object;`
String Functions	`substring`	N/A	`select count(0) from s3object where int(substring(_1,1,4))>1950 and int(substring(_1,1,4))<1960;`
String Functions	`substring`	substring with from negative number is valid considered as first	`select substring("123456789" from -4) from s3object;`
String Functions	`substring`	substring with from zero for out-of-bound number is valid just as (first,last)	`select substring("123456789" from 0 for 100) from s3object;`
String Functions	`trim`	N/A	`select trim(' foobar ') from s3object;`
String Functions	`trim`	N/A	`select trim(trailing from ' foobar ') from s3object;`
String Functions	`trim`	N/A	`select trim(leading from ' foobar ') from s3object;`
String Functions	`trim`	N/A	`select trim(both '12' from '1112211foobar22211122') from s3object;`
String Functions	`lower or upper`	N/A	`select lower('ABcD12#$e') from s3object;`
String Functions	`char_length, character_length`	N/A	`select count(*) from s3object where char_length(_3)=3;`
Complex queries	N/A	N/A	`select sum(cast(_1 as int)),max(cast(_3 as int)), substring('abcdefghijklm', (2-1)*3+sum(cast(_1 as int))/sum(cast(_1 as int))+1, (count() + count(0))/count(0)) from s3object;`
alias support	N/A	N/A	`select int(_1) as a1, int(_2) as a2 , (a1+a2) as a3 from s3object where a3>100 and a3<300;`

Related information

Amazon's S3 Select Object Content API

S3 supported select functions

Get to know the different S3 select supported functions.

S3 select supports the following functions:

- “Timestamp” on page 177
- “Aggregation” on page 179
- “String” on page 179
- “NULL” on page 180

For more information about S3 select functions, see **SelectObjectContent** within the Amazon Simple Storage Service (S3) API Reference section of AWS Documentation.

Timestamp

`to_timestamp` (string)

Description

Converts string to timestamp basic type. In the string format, any missing 'time' value is populated with zero; for missing month and day value, 1 is the default value. 'Timezone' is in format `+/-HH:mm` or `Z` , where the letter 'Z' indicates Coordinated Universal Time (UTC). Value of timezone can range between - 12:00 and +14:00.

Supported

Currently it can convert the following string formats into timestamp:

- `YYYY-MM-DDTHH:mm:ss.SSSSSS+/-HH:mm`
- `YYYY-MM-DDTHH:mm:ss.SSSSSSZ`
- `YYYY-MM-DDTHH:mm:ss+/-HH:mm`
- `YYYY-MM-DDTHH:mm:ssZ`
- `YYYY-MM-DDTHH:mm+/-HH:mm`
- `YYYY-MM-DDTHH:mmZ`
- `YYYY-MM-DDT`
- `YYYYT`

`to_string` (timestamp, format_pattern)

Description

Returns a string representation of the input timestamp in the given input string format.

Parameters

`to_string` parameters		
Format	**Example**	**Description**
yy	69	2-digit year
y	1969	4-digit year
yyyy	1969	Zero-padded 4-digit year
M	1	Month of the year
MM	01	Zero-padded month of the year
MMM	Jan	Abbreviated month of the year name
MMMM	January	Full month of the year name

Format	Example	Description
MMMMM	J	Month of the year first letter. **Note:** Not valid for use with the `to_timestamp` function.
d	2	Day of the month (1-31)
dd	02	Zero-padded day of the month (01-31)
a	AM	AM or PM of day
h	3	Hour of day (1-12)
hh	03	Zero-padded hour of the day (01-12)
H	3	Hour of the day (0-24)
HH	03	Zero-padded hour of the day (00-23)
m	4	Minute of the hour (0-59)
mm	04	Zero-padded minute of the hour (00-59)
s	5	Second of the minute (0-59)
ss	05	Zero padded second of the minute (00-59)
S	1	Fraction of the second (precision 0.1, range: 0.0-0.9)
SS	12	Fraction of the second (precision 0.01, range: 0.0-0.99)
SSS	123	Fraction of the second (precision: 0.01, range: 0.0-0.999).
SSSS	1234	Fraction of the second (precision: 0.001, range: 0.0-0.9999).
SSSSSS	123456	Fraction of the second (maximum precision: 1 nanosecond, range: 0.0-0.999999)
n	60000000	Nano of second
X	+07 or Z	Offset in hours, or “Z” if the offset is 0
XX or XXXX	+0700 or Z	Offset in hours and minutes or “Z” if the offset is 0
XXX or XXXXX	+07:00 or Z	Offset in hours and minutes or “Z” if the offset is 0
x	7	Offset in hours
xx or xxxx	700	Offset in hours and minutes
xxx or xxxxx	+07:00	Offset in hours and minutes

`extract` (date-part from timestamp)

Description

Returns integer according to date-part extract from input timestamp.

Supported

year, month, week, day, hour, minute, second, timezone_hour, timezone_minute.

`date_add` (date-part ,integer,timestamp)

Description

Returns timestamp, a calculation based on the results of input timestamp and date-part.

Supported

year, month, day, hour, minute, second.

`date_diff` (date-part,timestamp,timestamp)

Description

Return an integer, a calculated result of the difference between two timestamps according to date-part.

Supported

year, month, day, hour, minute, second.

`utcnow()`

Description

Return timestamp of current time.

Aggregation

`count()`

Description

Returns integers based on the number of rows that match a condition if there is one.

`sum(expression)`

Description

Returns a summary of expression on each row that matches a condition if there is one.

`avg(expression)`

Description

Returns an average expression on each row that matches a condition if there is one.

`max` (expression)

Description

Returns the maximal result for all expressions that match a condition if there is one.

`min` (expression)

Description

Returns the minimal result for all expressions that match a condition if there is one.

String

`substring` (string,from,for)

Description

Returns a string extract from the input string according to from, for inputs.

`Char_length`

Description

Returns a number of characters in string. `Character_length` also does the same.

trim([[leading | trailing | both remove_chars] from] string)

Description

Trims leading/trailing (or both) characters from the target string. The default value is a blank character.

Upper\lower

Description

Converts characters into uppercase or lowercase.

NULL

The NULL value indicates a missing or unknown that is NULL and that cannot produce a value on any arithmetic operations.

"The NULL use case" on page 180 depicts various NULL use cases.

The NULL use case

A is NULL	Result (NULL=UNKNOWN)
Not A	NULL
A or False	NULL
A or True	True
A or A	NULL
A and False	False
A and True	NULL
A and A	NULL

S3 alias programming construct

Alias programming construct is an essential part of the s3 select language because it enables better programming with objects that contain many columns or complex queries.

When you parse a statement with alias construct, it replaces the alias with a reference to the right projection column. On querying, the reference is evaluated like any other expression. Alias maintains result-cache. If an alias is used more than one time, the same expression is not evaluated. The same result is returned because the result from the cache is used. Currently, Red Hat supports the column alias.

There is a risk that self or cyclic reference might occur causing stack-overflow (endless-loop), for that concern upon evaluating an alias, it is validated for cyclic reference.

With each new row the cache is invalidated as the results may then differ.

Example

```
select int(_1) as a1, int(_2) as a2 , (a1+a2) as a3 from s3object where a3>100 and a3<300;
```

S3 parsing explained

The S3 select engine has parsers for CSV, Parquet, and JSON file formats. The parsers separate the commands into more processable components and are then attached to tags that define each component.

S3 CSV parsing

Use this information to get to know about how CSV parses S3-objects.

You can define the CSV definitions with input serialization uses these default values:

- Use {\n} for row-delimiter.
- Use {"} for quote.
- Use {\} for escape characters.

The `csv-header-info` is parsed upon `USE` appearing in the AWS-CLI; this is the first row in the input object containing the schema. Currently, output serialization and compression-type is not supported. The S3 select engine has a CSV parser which parses S3-objects:

- Each row ends with a row-delimiter.
- The field-separator separates the adjacent columns.
- The successive field separator defines the `NULL` column.
- The quote-character overrides the field-separator; that is, the field separator is any character between the quotes.
- The escape character disables any special character except the row delimiter.

The following are examples of CSV parsing rules:

CSV parsing		
Feature	**Description**	**Input (Tokens)**
`NULL`	Successive field delimiter	`,,1,,2, ==> {null}{null}{1}{null}{2}{null}`
`QUOTE`	The quote character overrides the field delimiter.	`11,22,"a,b,c,d",last ==> {11}{22}{"a,b,c,d"}{last}`
`Escape`	The escape character overrides the meta-character.	`11,22,str=\"abcd\"\,str2=\"123\",last ==> {11}{22}{str="abcd",str2="123"}{last}`
`row delimiter`	There is no closed quote; row delimiter is the closing line.	`11,22,a="str,44,55,66 ==> {11}{22}{a="str,44,55,66}`
`csv header info`	FileHeaderInfo tag	USE value means each token on the first line is the column-name; IGNORE value means to skip the first line.

Reference

- See Amazon's S3 Select Object Content API for more details.

S3 Parquet parsing

Apache Parquet is an open-source, columnar data file format designed for efficient data storage and retrieval. Use this information to get to know about how Apache Parquet parses S3-objects.

The S3 select engine's Parquet parser parses S3-objects as follows:

```
4-byte magic number "PAR1"
<Column 1 Chunk 1 + Column Metadata>
<Column 2 Chunk 1 + Column Metadata>
...
<Column N Chunk 1 + Column Metadata>
<Column 1 Chunk 2 + Column Metadata>
<Column 2 Chunk 2 + Column Metadata>
...
<Column N Chunk 2 + Column Metadata>
...
<Column 1 Chunk M + Column Metadata>
<Column 2 Chunk M + Column Metadata>
...
<Column N Chunk M + Column Metadata>
File Metadata
4-byte length in bytes of file metadata
4-byte magic number "PAR1"
```

- In the above example, there are N columns in this table, split into M row groups. The file metadata contains the locations of all the column metadata start locations.
- Metadata is written after the data to allow for single pass writing.

- All the column chunks can be found in the file metadata which should later be read sequentially.
- The format is explicitly designed to separate the metadata from the data. This allows splitting columns into multiple files, as well as having a single metadata file reference multiple parquet files.

S3 JSON parsing

Use this information to get to know about how JSON parses S3-objects.

The values in a JSON document can be nested within objects or arrays without limitations. When querying a specific value in a JSON document in the S3 select engine, the location of the value is specified via a path in the `SELECT` statement.

The generic structure of a JSON document does not have a row and column structure like CSV and Parquet. Instead, it is the SQL statement itself that defines the rows and columns when querying a JSON document.

The S3 select engine's JSON parser parses S3-objects as follows:

- The `FROM` clause in the `SELECT` statement defines the row boundaries
- A row in a JSON document is similar to how the row delimiter is used to define rows for CSV objects, and how row groups are used to define rows for Parquet objects
- Consider the following example:

```
{
"firstName": "Joe",
"lastName": "Jackson",
"gender": "male",
"age": "twenty",
"address":
  {
   "streetAddress": "101",
   "city": "San Diego",
   "state": "CA"
  },
"firstName": "Joe_2",
"lastName": "Jackson_2",
"gender": "male",
"age": 21,
"address":
 {
  "streetAddress": "101",
  "city": "San Diego",
  "state": "CA"
 },
"phoneNumbers":
 [
  { "type": "home1", "number": "734928_1","addr": 11 },
  { "type": "home2", "number": "734928_2","addr": 22 }
 ],
"key_after_array": "XXX",
"description" :
 {
  "main_desc" : "value_1",
  "second_desc" : "value_2"
 }
}

 # the from-clause define a single row.
 # _1 points to root object level.
 # _1.age appears twice in Documnet-row, the last value is used for the operation.
 query = "select
_1.firstname,_1.key_after_array,_1.age+4,_1.description.main_desc,_1.description.second
_desc from s3object[*];";
 expected_result = Joe_2,XXX,25,value_1,value_2


 # the from-clause points the phonenumbers array (it defines the _1)
 # each element in phoneNumbers array define a row.
 # in this case each element is an object contains 3 keys/values.
 # the query "can not access" values outside phonenumbers array, the query can access
only values appears on _1.phonenumbers path.
 query = "select cast(substring(_1.number,1,6) as int) *10 from
s3object[*].phonenumbers where _1.type='home2';";
 expected_result = 7349280
```

- The statement instructs the reader to search for the path *aa.bb.cc* and defines the row boundaries based on the occurrence of this path.
- A row begins when the reader encounters the path, and it ends when the reader exits the innermost part of the path, which in this case is the object *cc*.

S3 SQL limit operator

The SQL limit operator is used to limit the number of rows processed by the query.

Upon reaching the limit set by the user, the Ceph Object Gateway stops fetching additional chunks.

The following are examples of SQL limit operator for CSV, Parquet, and JSON.

Example of SQL limit operator for CSV

```
[cephuser@host ~]$ aws s3api --endpoint-url http://extensa027.ceph.rh.com:80 select-object-
content  --bucket bkt1 --key placement_data1 --expression-type 'SQL' --input-serialization
'{"CSV": {"FileHeaderInfo": "USE"}, "CompressionType": "NONE"}' --output-serialization
'{"CSV": {}}' --expression "select * from s3object limit 10;"

/dev/stdout
1,M,67.00,Others,91.00,Others,Commerce,58.00,Sci&Tech,No,55,Mkt&HR,58.8,Placed,270000
2,M,79.33,Central,78.33,Others,Science,77.48,Sci&Tech,Yes,86.5,Mkt&Fin,66.28,Placed,200000
3,M,65.00,Central,68.00,Central,Arts,64.00,Comm&Mgmt,No,75,Mkt&Fin,57.8,Placed,250000
4,M,56.00,Central,52.00,Central,Science,52.00,Sci&Tech,No,66,Mkt&HR,59.43,Not Placed,
5,M,85.80,Central,73.60,Central,Commerce,73.30,Comm&Mgmt,No,96.8,Mkt&Fin,55.5,Placed,425000
6,M,55.00,Others,49.80,Others,Science,67.25,Sci&Tech,Yes,55,Mkt&Fin,51.58,Not Placed,
7,F,46.00,Others,49.20,Others,Commerce,79.00,Comm&Mgmt,No,74.28,Mkt&Fin,53.29,Not Placed,
8,M,82.00,Central,64.00,Central,Science,66.00,Sci&Tech,Yes,67,Mkt&Fin,62.14,Placed,252000
9,M,73.00,Central,79.00,Central,Commerce,72.00,Comm&Mgmt,No,91.34,Mkt&Fin,61.29,Placed,2310
00
10,M,58.00,Central,70.00,Central,Commerce,61.00,Comm&Mgmt,No,54,Mkt&Fin,52.21,Not Placed,
```

Example of SQL limit operator for Parquet

```
[cephuser@extensa022 ~]$ aws s3api --endpoint-url http://extensa027.ceph.rh.com:80 select-
object-content  --bucket parquetbkt2 --key smallfile --expression-type 'SQL' --input-
serialization '{"Parquet": {}, "CompressionType": "NONE"}' --output-serialization '{"CSV":
{}}'  --expression "select * from s3object limit 5;"

/dev/stdout
4819,90340852.424672872,exile,0
4638,95603700.561133489,contradiction,1
1667,99440023.596563846,slam,2
7113,33213151.245887276,publisher,3
2872,43556628.954562038,demonstrate,4
```

Example of SQL limit operator for JSON

```
[cephuser@extensa027 ~]$ aws s3api --endpoint-url http://extensa027.ceph.rh.com:80 select-
object-content  --bucket bkt1 --key books_data.json --expression-type 'SQL' --input-
serialization '{"JSON": {"Type": "DOCUMENT"}, "CompressionType": "NONE"}' --output-
serialization '{"CSV": {}}' --expression "select * from s3object[*];"

/dev/stdout
id. : 1091
authors.name. : Makoto Satoh
authors.org. : Shinshu University
authors.id. : 2312688602
authors.name. : Ryo Muramatsu
authors.org. : Shinshu University
authors.id. : 2482909946
title. : Preliminary Design of a Network Protocol Learning Tool Based on the Comprehension
of High School Students: Design by an Empirical Study Using a Simple Mind Map
year. : 2013
#=== 0 ===#
id. : 1388
authors.name. : Mizue Kayama
authors.org. : Shinshu University
authors.id. : 2128134587
authors.name. : Pranava K. Jha
authors.id. : 2718958994
authors.name. : Kazunori Itoh
authors.org. : Shinshu University
authors.id. : 2101782692
title. : Further Results on Independence in Direct-Product Graphs.
year. : 2000
#=== 1 ===#
id. : 5781
authors.name. : Jovan Dj. Golic
authors.id. : 1237859792
authors.name. : Guglielmo Morgari
authors.id. : 220887178
title. : Vectorial fast correlation attacks.
year. : 2004
#=== 2 ===#
id. : 6522
authors.name. : Güzin Ulutas
authors.org. : Karadeniz Technical Univ.
authors.id. : 2022192081
authors.name. : Mustafa Ulutas
authors.org. : Karadeniz Technical Univ.
authors.id. : 2023460672
authors.name. : Vasif V. Nabiyev
authors.org. : Karadeniz Technical Univ.
authors.id. : 2174205032
title. : Improved Secret Image Sharing Method By Encoding Shared Values With Authentication
Bits
year. : 2011
#=== 3 ===#
```

Integrating Ceph Object Gateway with Trino

Integrate the Ceph Object Gateway with Trino, an important utility that enables the user to run SQL queries 9x faster on S3 objects.

PREREQUISITE

- A running cluster with Ceph Object Gateway installed.
- Docker or Podman installed.
- Buckets created.
- Objects are uploaded.

You can integrate Ceph Object Gateway with Trino for S3 select operations.

The following are some benefits of using Trino:

- Trino is a complete SQL engine.
- Pushes down S3 select requests wherein the Trino engine identifies part of the SQL statement that is cost effective to run on the server-side.

- Uses the optimization rules of Ceph/S3select to enhance performance.
- Leverages scalability and divides the original object into multiple equal parts, performs S3 select requests, and merges the request.

Important: If the `s3select` syntax does not work while querying through trino, use the SQL syntax.

1. Deploy Trino and hive.
   a. Clone the repository.

```
[cephuser@host01 ~]$ git clone https://github.com/ceph/s3select.git
[cephuser@host01 ~]$ cd s3select
```

   b. Modify the `hms_trino.yaml` file with S3 endpoint, access key, and secret key.

```
[cephuser@host01 s3select]$ cat container/trino/hms_trino.yaml
version: '3'
services:
  hms:
    image: galsl/hms:dev
    container_name: hms
    environment:
      # S3_ENDPOINT the CEPH/RGW end-point-url
      - S3_ENDPOINT=http://rgw_ip:port
      - S3_ACCESS_KEY=abc
      - S3_SECRET_KEY=abc
    # the container starts with booting the hive metastore
    command: sh -c '. ~/.bashrc; start_hive_metastore'
    ports:
      - 9083:9083
    networks:
      - trino_hms

  trino:
    image: trinodb/trino:405
    container_name: trino
    volumes:
      # the trino directory contains the necessary configuration
      - ./trino:/etc/trino
    ports:
      - 8080:8080
    networks:
      - trino_hms

networks:
  trino_hm
```

   c. Modify the `hive.properties` file with S3 endpoint, access key, and secret key.

```
[cephuser@host01 s3select]$ cat container/trino/trino/catalog/hive.properties
connector.name=hive
hive.metastore.uri=thrift://hms:9083

#hive.metastore.warehouse.dir=s3a://hive/

hive.allow-drop-table=true
hive.allow-rename-table=true
hive.allow-add-column=true
hive.allow-drop-column=true
hive.allow-rename-column=true

hive.non-managed-table-writes-enabled=true
hive.s3select-pushdown.enabled=true
hive.s3.aws-access-key=abc
hive.s3.aws-secret-key=abc

# should modify per s3-endpoint-url
hive.s3.endpoint=http://rgw_ip:port
#hive.s3.max-connections=1
#hive.s3select-pushdown.max-connections=1

hive.s3.connect-timeout=100s
hive.s3.socket-timeout=100s
```

```
hive.max-splits-per-second=10000
hive.max-split-size=128MB
```

2. Start a Trino container to integrate Ceph Object Gateway.

```
[cephuser@host01 s3select]$ sudo docker compose -f ./container/trino/hms_trino.yaml up
-d
```

3. Verify integration.

```
[cephuser@host01 s3select]$ sudo docker exec -it trino /bin/bash
trino@66f753905e82:/$ trino
trino> create schema hive.csvbkt1schema;
trino> create table hive.csvbkt1schema.polariondatacsv(c1 varchar,c2 varchar, c3
varchar, c4 varchar, c5 varchar, c6 varchar, c7 varchar, c8 varchar, c9 varchar) WITH
( external_location = 's3a://csvbkt1/',format = 'CSV');
trino> select * from hive.csvbkt1schema.polariondatacsv;
```

> **Note:** The external location must point to the bucket name or a directory, and not the end file.

S3 object ownership

S3 object ownership is a bucket-level setting in the Ceph Object Gateway (RGW) that defines who owns uploaded objects and whether Access Control Lists (ACLs) are evaluated for access decisions. Use object ownership to simplify access control, improve consistency in multi-tenant deployments, and reduce reliance on ACLs.

Ceph Object Gateway supports three S3-compatible ownership modes. Each ownership mode determines how object ownership is assigned at upload time and how ACLs are evaluated.

Object ownership can be configured at bucket creation or updated on existing buckets

- Set object ownership during bucket creation by using the `x-amz-object-ownership` header. For more information, see “Set object ownership when creating a bucket” on page 138.
- Use bucket ownership control operations to set, get, or delete ownership configuration. For more information, see “Updating bucket ownership controls” on page 139.

ObjectWriter

- This is the default behavior in Ceph Object Gateway.
- The uploader owns the object.
- ACLs are enabled.
- Access evaluation uses policies and ACLs.

BucketOwnerEnforced

- The bucket owner owns all objects.
- ACLs are disabled; ACL operations such as `PutObjectAcl` fail.
- Upload requests that specify ACLs other than `bucket-owner-full-control` fail.
- Access evaluation uses policies only.
- When you update an existing bucket to `BucketOwnerEnforced` by using `PutBucketOwnershipControls`, the bucket ACL must be private. If the bucket ACL is not private, the request fails with the error `InvalidBucketAclWithObjectOwnership` (HTTP 400). This requirement does not apply when you set `BucketOwnerEnforced` during bucket creation by using the `x-amz-object-ownership` header.

BucketOwnerPreferred

- Uploader owns the object by default.
- The bucket owner becomes the owner if the upload includes the bucket-owner-full-control canned ACL.

- ACLs remain enabled.
- Access evaluation uses policies and ACLs.

Using object ownership

You can use object ownership to simplify access control and ensure consistent ownership behavior across different workloads.

- **Policy-only access (optional):** Disable ACLs and rely solely on bucket or IAM-style policies.
- **Consistent multi-tenant behavior:** Ensure the bucket owner retains object ownership when multiple users upload to the same bucket.
- **Compatibility:** Match modern S3 application expectations for bucket-owner-controlled access.

Mode behavior summary

Mode	Ownership	ACL Behavior
ObjectWriter	Uploader	ACLs enabled; evaluated with policies
BucketOwnerEnforced	Bucket owner for all objects	ACLs disabled; ACL operations fail; access evaluated by policies only
BucketOwnerPreferred	Uploader by default; bucket owner when request includes `bucket-owner-full-control`	ACLs enabled; evaluated with policies

Effects on object operations

Object ownership controls how uploads and ACL-related APIs behave:

Uploads and ACL-related API behaviors in BucketOwnerEnforced operations

- `PutObjectAcl` and `PutBucketAcl` return an error.
- Upload requests that specify ACLs other than `bucket-owner-full-control` fail.
- Access decisions use only policies.

Uploads and ACL-related API behaviors in BucketOwnerPreferred and ObjectWriter operations

- ACLs are enabled and evaluated.
- Object ownership follows the mode-specific rules described above.

S3 Control operations

S3 Control operations enable account-level management of object storage behavior in Red Hat Ceph Storage, allowing administrators to apply configuration settings across all buckets owned by a tenant.

The S3 Control API provides account-level control over certain object storage features in the Ceph Object Gateway. Unlike standard S3 bucket operations, S3 Control operations apply to the entire tenant (account) rather than to individual buckets.

Account-level settings are useful for enforcing consistent security and access policies across all existing and future buckets without requiring administrators to configure each bucket individually.

Account-level versus bucket-level operations

Red Hat Ceph Storage supports both bucket-level S3 operations and account-level S3 Control operations.

- **Bucket-level S3 operations** apply to individual buckets and must be configured separately for each bucket.
- **Account-level S3 Control operations** apply across all buckets owned by a tenant and provide centralized control over supported features.

Account-level S3 Control settings do not represent bucket operations and are documented separately to reflect their broader scope.

Public Access Block at the account level

Red Hat Ceph Storage supports the S3 Control Public Access Block feature, which allows administrators to block public access across all buckets in a tenant.

The following S3 Control operations are available for managing account-level Public Access Block settings:

- "S3 GET PublicAccessBlock" on page 188
- "S3 PUT PublicAccessBlock" on page 188
- "S3 delete PublicAccessBlock" on page 189

S3 GET `PublicAccessBlock`

To get the S3 Block Public Access feature configured, use `GET` and specify a destination AWS account.

For information about the S3 Public Access Block feature used with Red Hat Ceph Storage, see "S3 Block Public Access" on page 136.

```
GET /v20180820/configuration/publicAccessBlock HTTP/1.1
Host: cname.domain.com
x-amz-account-id: ACCOUNTID
```

Request Headers

For more information about request headers, see S3 common request headers.

Response

The response is an HTTP 200 response and is returned in XML format.

The response includes `PublicAccessBlockConfiguration`, which contains the account-level Public Access Block settings.

PublicAccessBlockConfiguration		
Element	**Type**	**Description**
`BlockPublicAcls`	Boolean	Indicates whether requests that include public ACLs are blocked.
`IgnorePublicAcls`	Boolean	Indicates whether public ACLs are ignored when evaluating access permissions.
`BlockPublicPolicy`	Boolean	Indicates whether public bucket policies are blocked.
`RestrictPublicBuckets`	Boolean	Indicates whether access to buckets with public policies is restricted to authorized users only.

Related information

S3 delete PublicAccessBlock
S3 PUT PublicAccessBlock

S3 PUT `PublicAccessBlock`

Create or modify the account-level `PublicAccessBlock` configuration.

To use this operation, you must have the `s3:PutBucketPublicAccessBlock` permission.

> **Important:** If the `PublicAccessBlock` configuration is different between the bucket and the account, Amazon S3 uses the most restrictive combination of the bucket-level and account-level settings.

Note:
For information about the S3 Public Access Block feature used with Red Hat Ceph Storage, see "S3 Block Public Access" on page 136.

```
PUT /v20180820/configuration/publicAccessBlock HTTP/1.1
Host: cname.domain.com
x-amz-account-id: ACCOUNTID
<?xml version="1.0" encoding="UTF-8"?>
<PublicAccessBlockConfiguration xmlns="http://awss3control.amazonaws.com/doc/2018-08-20">
   <BlockPublicAcls>boolean</BlockPublicAcls>
   <IgnorePublicAcls>boolean</IgnorePublicAcls>
   <BlockPublicPolicy>boolean</BlockPublicPolicy>
   <RestrictPublicBuckets>boolean</RestrictPublicBuckets>
</PublicAccessBlockConfiguration>
```

Request Elements

The request includes `PublicAccessBlockConfiguration`, which specifies the account-level Public Access Block settings.

PublicAccessBlockConfiguration		
Element	**Type**	**Description**
`BlockPublicAcls`	Boolean	Specifies whether requests that include public ACLs are blocked.
`IgnorePublicAcls`	Boolean	Specifies whether public ACLs are ignored when evaluating access permissions.
`BlockPublicPolicy`	Boolean	Specifies whether public bucket policies are blocked.
`RestrictPublicBuckets`	Boolean	Specifies whether access to buckets with public policies is restricted to authorized users only.

Request Headers

For more information about request headers, see S3 common request headers.

Response

The response is an HTTP 200 response and is returned with an empty HTTP body.

Related information

S3 delete PublicAccessBlock
S3 GET PublicAccessBlock

S3 delete `PublicAccessBlock`

Use this to delete the `PublicAccessBlock` configuration for an account.

For information about the S3 Public Access Block feature used with Red Hat Ceph Storage, see "S3 Block Public Access" on page 136.

```
DELETE /v20180820/configuration/publicAccessBlock HTTP/1.1
Host: s3-control.amazonaws.com
x-amz-account-id: AccountId
```

Request Headers

For information about common request headers, see "S3 common request headers" on page 206.

Response

The response is an HTTP 200 response and is returned with an empty HTTP body.

Related information

S3 GET PublicAccessBlock
S3 PUT PublicAccessBlock

Ceph Object Gateway and the Swift API

Use a RESTful application programming interface (API) that is compatible with the Swift API data access model.

Manage the buckets and objects stored in Red Hat Ceph Storage cluster through the Ceph Object Gateway.

To use these APIs a running Red Hat Ceph Storage cluster and RESTful client are required.

"Support status for Swift functional features" on page 190 describes the support status for current Swift functional features.

Support status for Swift functional features

Feature	Status	Notes
Authentication	Supported	None
Get Account Metadata	Supported	No custom metadata
Swift ACLs	Supported	Supports a subset of Swift ACLs
List Containers	Supported	None
List Container's Objects	Supported	None
Create Container	Supported	None
Delete Container	Supported	None
Get Container Metadata	Supported	None
Add/Update Container Metadata	Supported	None
Delete Container Metadata	Supported	None
Get Object	Supported	None
Create/Update an Object	Supported	None
Create Large Object	Supported	None
Delete Object	Supported	None
Copy Object	Supported	None
Get Object Metadata	Supported	None
Add/Update Object Metadata	Supported	None
Temp URL Operations	Supported	None
CORS	Not Supported	None
Expiring Objects	Supported	None
Object Versioning	Not Supported	None
Static Website	Not Supported	None

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- A RESTful client.

Swift API limitations

Important: The following limitations should be used with caution. There are implications related to your hardware selections, so you should always discuss these requirements with your Red Hat account team.

- Maximum object size when using Swift API: 5GB
- Maximum metadata size when using Swift API: There is no defined limit on the total size of user metadata that can be applied to an object, but a single HTTP request is imited to 16,000 bytes.

Create a Swift user

Learn how to create a Swift user.

To test the Swift interface, create a Swift subuser. Creating a Swift user is a two-step process. The first step is to create the user. The second step is to create the secret key.

Note: In a multi-site deployment, always create a user on a host in the master zone of the master zone group.

Prerequisites

- Installation of the Ceph Object Gateway.
- Root-level access to the Ceph Object Gateway node.

1. Create the Swift user:

   Syntax

   ```
   radosgw-admin subuser create --uid=NAME --subuser=NAME:swift --access=full
   ```

   Replace `NAME` with the Swift user name, for example:

   Example

   ```
   [root@host01 ~]# radosgw-admin subuser create --uid=testuser --subuser=testuser:swift
   --access=full
   {
       "user_id": "testuser",
       "display_name": "First User",
       "email": "",
       "suspended": 0,
       "max_buckets": 1000,
       "auid": 0,
       "subusers": [
           {
               "id": "testuser:swift",
               "permissions": "full-control"
           }
       ],
       "keys": [
           {
               "user": "testuser",
               "access_key": "O8JDE41XMI74O185EHKD",
               "secret_key": "i4Au2yxG5wtr1JK01mI8kjJPM93HNAoVWOSTdJd6"
           }
       ],
       "swift_keys": [
           {
               "user": "testuser:swift",
               "secret_key": "13TLtdEW7bCqgttQgPzxFxziu0AgabtOc6vM8DLA"
           }
       ],
       "caps": [],
       "op_mask": "read, write, delete",
       "default_placement": "",
       "placement_tags": [],
   ```

```
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    },
    "temp_url_keys": [],
    "type": "rgw"
}
```

2. Create the secret key:

   Syntax

   ```
   radosgw-admin key create --subuser=NAME:swift --key-type=swift --gen-secret
   ```

   Replace `NAME` with the Swift user name, for example:

   Example

   ```
   [root@host01 ~]# radosgw-admin key create --subuser=testuser:swift --key-type=swift --
   gen-secret
   {
       "user_id": "testuser",
       "display_name": "First User",
       "email": "",
       "suspended": 0,
       "max_buckets": 1000,
       "auid": 0,
       "subusers": [
           {
               "id": "testuser:swift",
               "permissions": "full-control"
           }
       ],
       "keys": [
           {
               "user": "testuser",
               "access_key": "O8JDE41XMI74O185EHKD",
               "secret_key": "i4Au2yxG5wtr1JK01mI8kjJPM93HNAoVWOSTdJd6"
           }
       ],
       "swift_keys": [
           {
               "user": "testuser:swift",
               "secret_key": "a4ioT4jEP653CDcdU8p4OuhruwABBRZmyNUbnSSt"
           }
       ],
       "caps": [],
       "op_mask": "read, write, delete",
       "default_placement": "",
       "placement_tags": [],
       "bucket_quota": {
           "enabled": false,
           "check_on_raw": false,
           "max_size": -1,
           "max_size_kb": 0,
           "max_objects": -1
       },
       "user_quota": {
           "enabled": false,
           "check_on_raw": false,
           "max_size": -1,
           "max_size_kb": 0,
           "max_objects": -1
       },
       "temp_url_keys": [],
       "type": "rgw"
   }
   ```

Swift authenticating a user

To authenticate a user, make a request containing an `X-Auth-User` and a `X-Auth-Key` in the header.

Syntax

```
GET /auth HTTP/1.1
Host: swift.example.com
X-Auth-User: johndoe
X-Auth-Key: R7UUOLFDI2ZI9PRCQ53K
```

Example Response

```
HTTP/1.1 204 No Content
Date: Mon, 16 Jul 2012 11:05:33 GMT
Server: swift
X-Storage-Url: https://swift.example.com
X-Storage-Token: UOlCCC8TahFKlWuv9DB09TWHF0nDjpPElha0kAa
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
```

Note: You can retrieve data about Ceph's Swift-compatible service by executing `GET` requests using the `X-Storage-Url` value during authentication.

Reference

For more information, see:

- "Swift request headers" on page 211
- "Swift response headers" on page 211

Swift container operations

As a developer, you can perform container operations with the Swift application programming interface (API) through the Ceph Object Gateway. You can list, create, update, and delete containers. You can also add or update the container's metadata.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A RESTful client.

Swift container operations

A container is a mechanism for storing data objects. An account can have many containers, but container names must be unique. This API enables a client to create a container, set access controls and metadata, retrieve a container's contents, and delete a container. Since this API makes requests related to information in a particular user's account, all requests in this API must be authenticated unless a container's access control is deliberately made publicly accessible, that is, allows anonymous requests.

Note: The Amazon S3 API uses the term *bucket* to describe a data container. When you hear someone refer to a *bucket* within the Swift API, the term *bucket* might be construed as the equivalent of the term *container.*

One facet of object storage is that it does not support hierarchical paths or directories. Instead, it supports one level consisting of one or more containers, where each container might have objects. The RADOS Gateway's Swift-compatible API supports the notion of *pseudo-hierarchical containers*, which is a means of using object naming to emulate a container, or directory hierarchy without actually implementing one in the storage system. You can name objects with pseudo-hierarchical names, for example, photos/buildings/empire-state.jpg, but container names cannot contain a forward slash (/) character.

Important: When uploading large objects to versioned Swift containers, use the `--leave-segments` option with the `python-swiftclient` utility. Not using `--leave-segments` overwrites the manifest file. Consequently, an existing object is overwritten, which leads to data loss.

Swift update a container's Access Control List (ACL)

When a user creates a container, the user has read and write access to the container by default. To allow other users to read a container's contents or write to a container, you must specifically enable the user. You can also specify `*` in the `X-Container-Read` or `X-Container-Write` settings, which effectively enables all users to either read from or write to the container. Setting `*` makes the container public. That is it enables anonymous users to either read from or write to the container.

Syntax

```
POST /_AP_VERSION_/_ACCOUNT_/_TENANT_:_CONTAINER_ HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: _AUTH_TOKEN_
X-Container-Read: *
X-Container-Write: _UID1_, _UID2_, _UID3_
```

`X-Container-Read`

Description

The user IDs with read permissions for the container.

Type

Comma-separated string values of user IDs.

Required

No

`X-Container-Write`

Description

The user IDs with write permissions for the container.

Type

Comma-separated string values of user IDs.

Required

No

Swift list containers

A `GET` request that specifies the API version and the account will return a list of containers for a particular user account. Since the request returns a particular user's containers, the request requires an authentication token. The request cannot be made anonymously.

Syntax

```
GET /API_VERSION/ACCOUNT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

Request Parameters

`limit`

Description

Limits the number of results to the specified value.

Type

Integer

Valid Values

N/A

Required

Yes

`format`

Description

Limits the number of results to the specified value.

Type

Integer

Valid Values

`json` or `xml`

Required

No

`marker`

Description

Returns a list of results greater than the marker value.

Type

String

Valid Values

N/A

Required

No

The response contains a list of containers, or returns with an HTTP `204` response code.

Response Entities

`account`

Description

A list for account information.

Type

Container

`container`

Description

The list of containers.

Type

Container

`name`

Description

The name of a container.

Type

String

`bytes`

Description

The size of the container.

Type

Integer

Swift list a container's objects

To list the objects within a container, make a `GET` request with the API version, account, and the name of the container. You can specify query parameters to filter the full list, or leave out the parameters to return a list of the first 10,000 object names stored in the container.

Syntax

```
GET /_AP_VERSION_/_TENANT_:_CONTAINER_ HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: _AUTH_TOKEN_
```

Request Parameters

`format`

Description

Limits the number of results to the specified value.

Type

Integer

Valid Values

`json` or `xml`

Required

No

`prefix`

Description

Limits the result set to objects beginning with the specified prefix.

Type

String

Valid Values

N/A

Required

No

`marker`

Description

Returns a list of results greater than the marker value.

Type

String

Valid Values

N/A

Required

No

`limit`

Description

Limits the number of results to the specified value.

Type

Integer

Valid Values

0 - 10,000

Required

No

`delimiter`

Description

The delimiter between the prefix and the rest of the object name.

Type

String

Valid Values

N/A

Required

No

`path`

Description

The pseudo-hierarchical path of the objects.

Type

String

Valid Values

N/A

Required

No

Response Entities

`container`

Description

The container.

Type

Container

`object`

Description

An object within the container.

Type

Container

`name`

Description

The name of an object within the container.

Type

String

`hash`

Description

A hash code of the object's contents.

Type

String

`last_modified`

Description

The last time the object's contents were modified.

Type

Date

`content_type`

Description

The type of content within the object.

Type

String

Swift create a container

To create a new container, make a `PUT` request with the API version, account, and the name of the new container. The container name must be unique, must not contain a forward-slash (/) character, and should be less than 256 bytes. You can include access control headers and metadata headers in the request. You can also include a storage policy identifying a key for a set of placement pools. For example, run `radosgw-admin zone get` to see a list of available keys under `placement_pools`. A storage policy enables you to specify a special set of pools for the container, for example, SSD-based storage. The operation is idempotent. If you make a request to create a container that already exists, it will return with a HTTP 202 return code, but will not create another container.

Syntax

```
PUT /_AP_VERSION_/_ACCOUNT_/_TENANT_:_CONTAINER_ HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: _AUTH_TOKEN_
X-Container-Read: _COMMA_SEPARATED_UIDS_
X-Container-Write: _COMMA_SEPARATED_UIDS_
X-Container-Meta-_KEY_:VALUE
X-Storage-Policy: _PLACEMENT_POOLS_KEY_
```

Headers

`X-Container-Read`

Description

The user IDs with read permissions for the container.

Type

Comma-separated string values of user IDs.

Required

No

`X-Container-Write`

Description

The user IDs with write permissions for the container.

Type

Comma-separated string values of user IDs.

Required

No

`X-Container-Meta-_KEY`

Description

A user-defined metadata key that takes an arbitrary string value.

Type

String

Required

No

`X-Storage-Policy`

Description

The key that identifies the storage policy under `placement_pools` for the Ceph Object Gateway. Run `radosgw-admin zone get` for available keys.

Type

String

Required

No

If a container with the same name already exists, and the user is the container owner then the operation will succeed. Otherwise, the operation will fail.

HTTP Response

`409`

Status Code

`BucketAlreadyExists`

Description

The container already exists under a different user's ownership.

Swift delete a container

To delete a container, make a `DELETE` request with the API version, account, and the name of the container. The container must be empty. If you'd like to check if the container is empty, run a `HEAD` request against the container. Once you've successfully removed the container, you'll be able to reuse the container name.

Syntax

```
DELETE /_AP_VERSION_/_ACCOUNT_/_TENANT_:_CONTAINER_ HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: _AUTH_TOKEN_
```

HTTP Response

`204`

Status Code

`NoContent`

Description

The container was removed.

Swift add or update the container metadata

To add metadata to a container, make a `POST` request with the API version, account, and container name. You must have write permissions on the container to add or update metadata.

Syntax

```
POST /_AP_VERSION_/_ACCOUNT_/_TENANT_:_CONTAINER_ HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: _AUTH_TOKEN_
X-Container-Meta-Color: red
X-Container-Meta-Taste: salty
```

Request Headers

`X-Container-Meta-_KEY`

Description

A user-defined metadata key that takes an arbitrary string value.

Type

String

Required

No

Swift object operations

As a developer, you can perform object operations with the Swift application programming interface (API) through the Ceph Object Gateway. You can list, create, update, and delete objects. You can also add or update the object's metadata.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A RESTful client.

Swift object operations

An object is a container for storing data and metadata. A container might have many objects, but the object names must be unique. This API enables a client to create an object, set access controls and metadata, retrieve an object’s data and metadata, and delete an object. Since this API makes requests related to information in a particular user’s account, all requests in this API must be authenticated. Unless the container or object’s access control is deliberately made publicly accessible, that is, allows anonymous requests.

Swift get an object

To retrieve an object, make a `GET` request with the API version, account, container, and object name. You must have read permissions on the container to retrieve an object within it.

Syntax

```
GET /_AP_VERSION_/_ACCOUNT_/_TENANT_:_CONTAINER_/_OBJECT_ HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: _AUTH_TOKEN_
```

Request Headers

`range`

Description

To retrieve a subset of an object’s contents, you can specify a byte range.

Type

Date

Required

No

`If-Modified-Since`

Description

Only copies if modified since the date and time of the source object's `last_modified` attribute.

Type

Date

Required

No

`If-Unmodified-Since`

Description

Only copies if not modified since the date and time of the source object's `last_modified` attribute.

Type

Date

Required

No

`Copy-If-Match`

Description

Copies only if the ETag in the request matches the source object's ETag.

Type

ETag

Required

No

`Copy-If-None-Match`

Description

Copies only if the `ETag` in the request does not match the source object's ETag.

Type

ETag

Required

No

Response Headers

`Content-Range`

Description

The range of the subset of object contents. Returned only if the range header field was specified in the request.

Swift create or update an object

To create a new object, make a `PUT` request with the API version, account, container name, and the name of the new object. You must have write permission on the container to create or update an object. The object name must be unique within the container. The `PUT` request is not idempotent, so if you do not use a unique name, the request will update the object. However, you can use pseudo-hierarchical syntax in the object name to distinguish it from another object of the same name if it is under a different pseudo-hierarchical directory. You can include access control headers and metadata headers in the request.

Syntax

```
PUT /_AP_VERSION_/_ACCOUNT_/_TENANT_:_CONTAINER_ HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: _AUTH_TOKEN_
```

Request Headers

`ETag`

Description

An MD5 hash of the object's contents. Recommended.

Type

String

Valid Values

N/A

Required

No

`Content-Type`

Description

An MD5 hash of the object's contents.

Type

String

Valid Values

N/A

Required

No

`Transfer-Encoding`

Description

Indicates whether the object is part of a larger aggregate object.

Type

String

Valid Values

`chunked`

Required

No

Swift delete an object

To delete an object, make a `DELETE` request with the API version, account, container, and object name. You must have write permissions on the container to delete an object within it. Once you've successfully deleted the object, you will be able to reuse the object name.

Syntax

```
DELETE /_API_VERSION_/_ACCOUNT_/_TENANT_:_CONTAINER_/_OBJECT_ HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: _AUTH_TOKEN_
```

Swift copy an object

Copying an object allows you to make a server-side copy of an object, so that you do not have to download it and upload it under another container. To copy the contents of one object to another object, you can make either a `PUT` request or a `COPY` request with the API version, account, and the container name.

For a `PUT` request, use the destination container and object name in the request, and the source container and object in the request header.

For a `Copy` request, use the source container and object in the request, and the destination container and object in the request header. You must have write permission on the container to copy an object. The destination object name must be unique within the container. The request is not idempotent, so if you do not use a unique name, the request will update the destination object. You can use pseudo-hierarchical syntax in the object name to distinguish the destination object from the source object of the same name if it is under a different pseudo-hierarchical directory. You can include access control headers and metadata headers in the request.

Syntax

```
PUT /_AP_VERSION_/_ACCOUNT_/_TENANT_:_CONTAINER_ HTTP/1.1
X-Copy-From: _TENANT_:_SOURCE_CONTAINER_/_SOURCE_OBJECT_
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: _AUTH_TOKEN_
```

or alternatively:

Syntax

```
COPY /_AP_VERSION_/_ACCOUNT_/_TENANT_:_SOURCE_CONTAINER_/_SOURCE_OBJECT_ HTTP/1.1
Destination: _TENANT_:_DEST_CONTAINER_/_DEST_OBJECT_
```

Request Headers

`X-Copy-From`

Description

Used with a `PUT` request to define the source container/object path.

Type

String

Required

Yes, if using `PUT`.

`Destination`

Description

Used with a `COPY` request to define the destination container/object path.

Type

String

Required

Yes, if using `COPY`.

`If-Modified-Since`

Description

Only copies if modified since the date and time of the source object's `last_modified` attribute.

Type

Date

Required

No

`If-Unmodified-Since`

Description

Only copies if not modified since the date and time of the source object's `last_modified` attribute.

Type

Date

Required

No

`Copy-If-Match`

Description

Copies only if the ETag in the request matches the source object's ETag.

Type

ETag

Required

No

`Copy-If-None-Match`

Description

Copies only if the `ETag` in the request does not match the source object's ETag.

Type

ETag

Required

No

Swift get object metadata

To retrieve an object's metadata, make a `HEAD` request with the API version, account, container, and object name. You must have read permissions on the container to retrieve metadata from an object within the container. This request returns the same header information as the request for the object itself, but it does not return the object's data.

Syntax

```
HEAD /_AP_VERSION_/_ACCOUNT_/_TENANT_:_CONTAINER_/_OBJECT_ HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: _AUTH_TOKEN_
```

Swift add or update object metadata

To add metadata to an object, make a `POST` request with the API version, account, container, and object name. You must have write permissions on the parent container to add or update metadata.

Syntax

```
POST /_AP_VERSION_/_ACCOUNT_/_TENANT_:_CONTAINER_/_OBJECT_ HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: _AUTH_TOKEN_
```

Request Headers

`X-Object-Meta-_KEY`

Description

A user-defined meta data key that takes an arbitrary string value.

Type

String

Required

No

Swift temporary URL operations

To allow temporary access, temp url functionality is supported by swift endpoint of `radosgw`. For example GET requests, to objects without the need to share credentials.

For this functionality, initially the value of `X-Account-Meta-Temp-URL-Key` and optionally `X-Account-Meta-Temp-URL-Key-2` should be set. The Temp URL functionality relies on a HMAC-SHA1 signature against these secret keys.

Swift get temporary URL objects

Temporary URL uses a cryptographic HMAC-SHA1 signature, which includes the following elements:

- The value of the Request method, `GET` for instance.
- The expiry time, in the format of seconds since the epoch, that is, Unix time.
- The request path that starts from `v1` onwards.

The items are normalized with newlines that are appended between them. An HMAC is generated that uses the SHA-1 hashing algorithm against one of the Temp URL Keys posted earlier.

The following example is a sample python script to demonstrate this.

Example

```
import hmac
from hashlib import sha1
from time import time

method = 'GET'
host = 'https://objectstore.example.com'
duration_in_seconds = 300  # Duration for which the url is valid
expires = int(time() + duration_in_seconds)
path = '/v1/your-bucket/your-object'
key = 'secret'
hmac_body = '%s\n%s\n%s' % (method, expires, path)
hmac_body = hmac.new(key, hmac_body, sha1).hexdigest()
sig = hmac.new(key, hmac_body, sha1).hexdigest()
rest_uri = "{host}{path}?temp_url_sig={sig}&temp_url_expires={expires}".format(
     host=host, path=path, sig=sig, expires=expires)
print rest_uri
```

Example Output

```
https://objectstore.example.com/v1/your-bucket/your-object?
temp_url_sig=ff4657876227fc6025f04fcf1e82818266d022c6&temp_url_expires=1423200992
```

Swift POST temporary URL keys

A `POST` request to the swift account with the required Key will set the secret temp URL key for the account against which temporary URL access can be provided to accounts. Up to two keys are supported, and signatures are checked against both the keys, if present, so that keys can be rotated without invalidating the temporary URLs.

Syntax

```
POST /API_VERSION/ACCOUNT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

Request Headers

`X-Account-Meta-Temp-URL-Key`

Description

A user-defined key that takes an arbitrary string value.

Type

String

Required

Yes

`X-Account-Meta-Temp-URL-Key-2`

Description

A user-defined key that takes an arbitrary string value.

Type

String

Required

No

Swift multi-tenancy container operations

When a client application accesses containers, it always operates with credentials of a particular user. In a Red Hat Ceph Storage cluster, every user belongs to a tenant. Consequently, every container operation has an implicit tenant in its context if no tenant is specified explicitly. Thus multi-tenancy is completely backward compatible with previous releases, as long as the referred containers and referring user belong to the same tenant.

Extensions employed to specify an explicit tenant differ according to the protocol and authentication system used.

A colon character separates tenant and container, thus a sample URL would be:

Example

```
https://rgw.domain.com/tenant:container
```

By contrast, in a `create_container()` method, simply separate the tenant and container in the container method itself:

Example

```
create_container("tenant:container")
```

Ceph RESTful API specifications

As a storage administrator, you can access the various Ceph sub-systems through the Ceph RESTful API endpoints. This is a reference section for the available Ceph RESTful API methods.

Before you begin, make sure that you have the following prerequisites in place:

- An understanding of how to use a RESTful API.
- A healthy running Red Hat Ceph Storage cluster.
- The Ceph Manager `dashboard` module is enabled.

S3 common request headers

The following table lists the valid common request headers and their descriptions.

Request Headers	
Request Header	**Description**
`CONTENT_LENGTH`	Length of the request body.
`DATE`	Request time and date (in UTC).
`HOST`	The name of the host server.
`AUTHORIZATION`	Authorization token.

S3 common response status codes

The following table lists the valid common HTTP response status and its corresponding code.

Response Status	
HTTP Status	**Response Code**
100	Continue
200	Success
201	Created
202	Accepted
204	NoContent
206	Partial content
304	NotModified
400	InvalidArgument
400	InvalidDigest
400	BadDigest
400	InvalidBucketName
400	InvalidObjectName
400	UnresolvableGrantByEmailAddress
400	InvalidPart
400	InvalidPartOrder
400	RequestTimeout
400	EntityTooLarge
403	AccessDenied
403	UserSuspended
403	RequestTimeTooSkewed
404	NoSuchKey
404	NoSuchBucket
404	NoSuchUpload
405	MethodNotAllowed
408	RequestTimeout
409	BucketAlreadyExists
409	BucketNotEmpty
411	MissingContentLength
412	PreconditionFailed
416	InvalidRange

HTTP Status	Response Code
422	UnprocessableEntity
500	InternalError

S3 supported and unsupported verbs

Use this information to know the latest supported and unsupported S3 verbs.

- Supported verbs: lists the S3 API verbs that are fully supported by Red Hat Ceph Storage Object Gateway and are available for use in customer applications and integrations
- Unsupported verbs: lists the S3 API verbs that are not currently supported by Red Hat Ceph Storage Object Gateway and cannot be used.

Supported verbs	
Action	**API**
AbortMultipartUpload	S3
CompleteMultipartUpload	S3
CopyObject	S3
CreateBucket	S3
CreateMultipartUpload	S3
DeleteBucket	S3
DeleteBucketCORS	S3
DeleteBucketEncryption	S3
DeleteBucketLifecycle	S3
DeleteBucketPolicy	S3
DeleteBucketOwnershipControls	S3
DeleteBucketReplication	S3
DeleteBucketTagging	S3
DeleteBucketWebsite	S3
DeleteObject	S3
DeleteObjects	S3
DeleteObjectTagging	S3
DeletePublicAccessBlock	S3
GetBucketAcl	S3
GetBucketCORS	S3
GetBucketEncryption	S3
GetBucketLocation	S3
GetBucketLogging	S3
GetBucketNotificationConfiguration	S3
GetBucketPolicy	S3
GetBucketPolicyStatus	S3
GetBucketOwnershipControls	S3
GetBucketReplication	S3
GetBucketRequestPayment	S3
GetBucketTagging	S3

Action	API
GetBucketVersioning	S3
GetBucketWebsite	S3
GetObject	S3
GetObjectAcl	S3
GetObjectAttributes	S3
GetObjectLegalHold	S3
GetObjectLockConfiguration	S3
GetObjectRetention	S3
GetObjectTagging	S3
GetObjectTorrent	S3
GetPublicAccessBlock	S3
HeadBucket	S3
HeadObject	S3
ListBuckets	S3
ListMultipartUploads	S3
ListObjects	S3
ListObjectsV2	S3
ListObjectVersions	S3
ListParts	S3
PutBucketAcl	S3
PutBucketCORS	S3
PutBucketEncryption	S3
PutBucketLifecycle	S3
PutBucketLifecycleConfiguration	S3
PutBucketLogging	S3
PutBucketNotificationConfiguration	S3
PutBucketPolicy	S3
PutBucketOwnershipControls	S3
PutBucketReplication	S3
PutBucketRequestPayment	S3
PutBucketTagging	S3
PutBucketVersioning	S3
PutBucketWebsite	S3
PutObject	S3
PutObjectAcl	S3
PutObjectLegalHold	S3
PutObjectLockConfiguration	S3
PutObjectRetention	S3
PutObjectTagging	S3
PutPublicAccessBlock	S3
RestoreObject	S3
SelectObjectContent	S3

Action	API
UploadPart	S3
UploadPartCopy	S3
DeletePublicAccessBlock	S3Control
GetPublicAccessBlock	S3Control
PutPublicAccessBlock	S3Control
AssumeRole	STS
AssumeRoleWithWebIdentity	STS
GetSessionToken	STS
AddClientIdToOpenIDConnectProvider	IAM
AttachGroupPolicy	IAM
AttachRolePolicy	IAM
AttachUserPolicy	IAM
CreateOpenIDConnectProvider	IAM
CreateRole	IAM
DeleteOpenIDConnectProvider	IAM
DeleteRole	IAM
GetAccountSummary	IAM
GetOpenIDConnectProvider	IAM
GetRole	IAM
ListOpenIDConnectProviders	IAM
ListRoles	IAM
UpdateOpenIDConnectProviderThumbprint	IAM
DeleteBucketNotification	S3 extension
CreateTopic	SNS
GetTopicAttributes	SNS
DeleteTopic	SNS
ListTopics	SNS
GetCallerIdentity	STS

Unsupported verbs

Action	API
DeleteBucketInventoryConfiguration	S3
DeleteIntelligentTieringConfiguration	S3
GetBucketInventoryConfiguration	S3
GetIntelligentTieringConfiguration	S3
ListBucketIntelligentTieringConfigurations	S3
ListBucketInventoryConfigurations	S3
PutBucketIntelligentTieringConfiguration	S3
PutBucketInventoryConfiguration	S3
AssumeRoleWithSAML	STS
DecodeAuthorizationMessage	STS
GetAccessKeyInfo	STS

Action	API
GetFederationToken	STS
CreatePolicy	IAM
CreateSAMLProvider	IAM
DeletePolicy	IAM
DeleteSAMLProviders	IAM
GetPolicy	IAM
GetSAMLProviders	IAM
ListPolicies	IAM
ListSAMLProviders	IAM

S3 unsupported header fields

Use this information to get to know S3 unsupported header fields.

Unsupported header fields	
Name	**Type**
x-amz-security-token	Request
Server	Response
x-amz-delete-marker	Response
x-amz-id-2	Response
x-amz-request-id	Response
x-amz-version-id	Response

Swift request headers

Request headers			
Name	**Description**	**Type**	**Required**
X-Auth-User	The key Ceph Object Gateway username to authenticate.	String	Yes
X-Auth-Key	The key associated to a Ceph Object Gateway username.	String	Yes

Swift response headers

The response from the server should include an X-Auth-Token value. The response might also contain a X-Storage-Url that provides the API_VERSION_/_ACCOUNT prefix that is specified in other requests throughout the API documentation.

Response Headers		
Name	**Description**	**Type**
X-Storage-Token	The authorization token for the X-Auth-User specified in the request.	String

Name	Description	Type
X-Storage-Url	The URL and _API_VERSION_/_ACCOUNT_ path for the user.	String

Examples using the Secure Token Service APIs

These examples are using Python's boto3 module to interface with the Ceph Object Gateway's implementation of the Secure Token Service (STS). In these examples, TESTER2 assumes a role created by TESTER1, as to access S3 resources owned by TESTER1 based on the permission policy attached to the role.

The *AssumeRole* example creates a role, assigns a policy to the role, then assumes a role to get temporary credentials and access to S3 resources using those temporary credentials.

The *AssumeRoleWithWebIdentity* example authenticates users using an external application with Keycloak, an OpenID Connect identity provider, assumes a role to get temporary credentials and access S3 resources according to the permission policy of the role.

AssumeRole Example

```
import boto3

iam_client = boto3.client(iam,
aws_access_key_id=ACCESS_KEY_OF_TESTER1,
aws_secret_access_key=SECRET_KEY_OF_TESTER1,
endpoint_url=<IAM URL>,
region_name='
)

policy_document = "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"AWS\":[\"arn:aws:iam:::user/TESTER1\"]},\"Action\":
[\"sts:AssumeRole\"]}]}"

role_response = iam_client.create_role(
AssumeRolePolicyDocument=policy_document,
Path=/,
RoleName='S3Access,
)

role_policy = "{\"Version\":\"2012-10-17\",\"Statement\":{\"Effect\":\"Allow\",\"Action\":
\"s3:*\",\"Resource\":\"arn:aws:s3:::*\"}}"

response = iam_client.putROLEpolicy(
RoleName=S3Access,
PolicyName=Policy1,
PolicyDocument=role_policy
)

sts_client = boto3.client(sts,
aws_access_key_id=ACCESS_KEY_OF_TESTER2,
aws_secret_access_key=SECRET_KEY_OF_TESTER2,
endpoint_url=<STS URL>,
region_name=',
)

response = sts_client.assume_role(
RoleArn=role_response['Role][Arn],
RoleSessionName=Bob,
DurationSeconds=3600
)

s3client = boto3.client(s3,
aws_access_key_id = response[Credentials][AccessKeyId],
aws_secret_access_key = response[Credentials][SecretAccessKey],
aws_session_token = response[Credentials][SessionToken],
endpoint_url=<S3 URL>,
region_name=',)

bucket_name = 'my-bucket
s3bucket = s3client.create_bucket(Bucket=bucket_name)
resp = s3client.list_buckets()
```

AssumeRoleWithWebIdentity Example

```
import boto3

iam_client = boto3.client('iam',
aws_access_key_id=ACCESS_KEY_OF_TESTER1,
aws_secret_access_key=SECRET_KEY_OF_TESTER1,
endpoint_url=<IAM URL>,
region_name=''
)

oidc_response = iam_client.create_openIDconnect_provider(
    Url=<URL of the OpenID Connect Provider>,
    ClientIDList=[
        <Client id registered with the IDP>
    ],
    ThumbprintList=[
        <IDP THUMBPRINT>
 ]
)

policy_document = "{\"Version\":\"2012-10-17\",\"Statement\":\[\{\"Effect\":\"Allow\",
\"Principal\":\{\"Federated\":\[\"arn:aws:iam:::oidc-provider/localhost:8080/auth/realms/
demo\"\]\},\"Action\":\[\"sts:AssumeRoleWithWebIdentity\"\],\"Condition\":\
{\"StringEquals\":\{\"localhost:8080/auth/realms/demo:app_id\":\"customer-portal\"\}\}\}\]
\}"
role_response = iam_client.create_role(
AssumeRolePolicyDocument=policy_document,
Path='/',
RoleName='S3Access',
)

role_policy = "{\"Version\":\"2012-10-17\",\"Statement\":{\"Effect\":\"Allow\",\"Action\":
\"s3:*\",\"Resource\":\"arn:aws:s3:::*\"}}"

response = iam_client.put_role_policy(
    RoleName='S3Access',
    PolicyName='Policy1',
    PolicyDocument=role_policy
)

sts_client = boto3.client('sts',
aws_access_key_id=ACCESS_KEY_OF_TESTER2,
aws_secret_access_key=SECRET_KEY_OF_TESTER2,
endpoint_url=<STS URL>,
region_name='',
)

response = sts_client.assume_role_with_web_identity(
RoleArn=role_response['Role']['Arn'],
RoleSessionName='Bob',
DurationSeconds=3600,
WebIdentityToken=<Web Token>
)

s3client = boto3.client(s3,
aws_access_key_id = response[Credentials][AccessKeyId],
aws_secret_access_key = response[Credentials][SecretAccessKey],
aws_session_token = response[Credentials][SessionToken],
endpoint_url=<S3 URL>,
region_name=',)

bucket_name = 'my-bucket
s3bucket = s3client.create_bucket(Bucket=bucket_name)
resp = s3client.list_buckets()
```

Reference

For more information about using Python's boto module, see Test S3 access.