# Red Hat Ceph Storage 4

## Developer Guide

Using the various application programming interfaces for Red Hat Ceph Storage

Last Updated: 2021-09-15

# Red Hat Ceph Storage 4 Developer Guide

Using the various application programming interfaces for Red Hat Ceph Storage

## Legal Notice



## Abstract

This document provides instructions for Using the various application programming interfaces for Red Hat Ceph Storage running on AMD64 and Intel 64 architectures. Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

# Table of Contents

CHAPTER 1. CEPH OBJECT GATEWAY ADMINISTRATIVE API ..... 5
1.1. PREREQUISITES 6
1.2. ADMINISTRATION OPERATIONS 6
1.3. ADMINISTRATION AUTHENTICATION REQUESTS 6
1.4. CREATING AN ADMINISTRATIVE USER 14
1.5. GET USER INFORMATION 16
1.6. CREATE A USER 17
1.7. MODIFY A USER 20
1.8. REMOVE A USER 22
1.9. CREATE A SUBUSER 22
1.10. MODIFY A SUBUSER 24
1.11. REMOVE A SUBUSER 25
1.12. ADD CAPABILITIES TO A USER 26
1.13. REMOVE CAPABILITIES FROM A USER 27
1.14. CREATE A KEY 28
1.15. REMOVE A KEY 30
1.16. BUCKET NOTIFICATIONS 30
1.16.1. Prerequisites 31
1.16.2. Creating a topic 31
1.16.3. Getting topic information 34
1.16.4. Listing topics 35
1.16.5. Deleting topics 36
1.16.6. Event record 37
1.16.7. Supported event types 38
1.16.8. Additional Resources 39
1.17. GET BUCKET INFORMATION 39
1.18. CHECK A BUCKET INDEX 40
1.19. REMOVE A BUCKET 41
1.20. LINK A BUCKET 42
1.21. UNLINK A BUCKET 43
1.22. GET A BUCKET OR OBJECT POLICY 44
1.23. REMOVE AN OBJECT 45
1.24. QUOTAS 46
1.25. GET A USER QUOTA 46
1.26. SET A USER QUOTA 46
1.27. GET A BUCKET QUOTA 46
1.28. SET A BUCKET QUOTA 48
1.29. GET USAGE INFORMATION 48
1.30. REMOVE USAGE INFORMATION 50
1.31. STANDARD ERROR RESPONSES 50

CHAPTER 2. CEPH OBJECT GATEWAY AND THE S3 API ..... 52
2.1. PREREQUISITES 52
2.2. S3 LIMITATIONS 52
2.3. ACCESSING THE CEPH OBJECT GATEWAY WITH THE S3 API 52
2.3.1. Prerequisites 52
2.3.2. S3 authentication 53
2.3.3. S3 server-side encryption 54
2.3.4. S3 access control lists 55
2.3.5. Preparing access to the Ceph Object Gateway using S3 56
2.3.6. Accessing the Ceph Object Gateway using Ruby AWS S3 57

2.3.7. Accessing the Ceph Object Gateway using Ruby AWS SDK 62
2.3.8. Accessing the Ceph Object Gateway using PHP 68
2.3.9. Secure Token Service 72
2.3.9.1. The Secure Token Service application programming interfaces 73
2.3.9.2. Configuring the Secure Token Service 76
2.3.9.3. Creating a user for an OpenID Connect provider 77
2.3.9.4. Obtaining a thumbprint of an OpenID Connect provider 78
2.3.9.5. Configuring and using STS Lite with Keystone (Technology Preview) 79
2.3.9.6. Working around the limitations of using STS Lite with Keystone (Technology Preview) 83
2.4. S3 BUCKET OPERATIONS 84
2.4.1. Prerequisites 85
2.4.2. S3 create bucket notifications 85
2.4.3. S3 get bucket notifications 89
2.4.4. S3 delete bucket notifications 91
2.4.5. Accessing bucket host names 92
2.4.6. S3 list buckets 93
2.4.7. S3 return a list of bucket objects 93
2.4.8. S3 create a new bucket 95
2.4.9. S3 delete a bucket 96
2.4.10. S3 bucket lifecycle 96
2.4.11. S3 GET bucket lifecycle 98
2.4.12. S3 create or replace a bucket lifecycle 99
2.4.13. S3 delete a bucket lifecycle 99
2.4.14. S3 get bucket location 100
2.4.15. S3 get bucket versioning 100
2.4.16. S3 put the bucket versioning 100
2.4.17. S3 get bucket access control lists 101
2.4.18. S3 put bucket Access Control Lists 102
2.4.19. S3 get bucket cors 102
2.4.20. S3 put bucket cors 103
2.4.21. S3 delete a bucket cors 103
2.4.22. S3 list bucket object versions 103
2.4.23. S3 head bucket 105
2.4.24. S3 list multipart uploads 105
2.4.25. S3 bucket policies 107
2.4.26. S3 get the request payment configuration on a bucket 110
2.4.27. S3 set the request payment configuration on a bucket 110
2.4.28. Multi-tenant bucket operations 111
2.4.29. Additional Resources 111
2.5. S3 OBJECT OPERATIONS 112
2.5.1. Prerequisites 112
2.5.2. S3 get an object from a bucket 113
2.5.3. S3 get information on an object 113
2.5.4. S3 add an object to a bucket 114
2.5.5. S3 delete an object 115
2.5.6. S3 delete multiple objects 115
2.5.7. S3 get an object's Access Control List (ACL) 115
2.5.8. S3 set an object's Access Control List (ACL) 116
2.5.9. S3 copy an object 117
2.5.10. S3 add an object to a bucket using HTML forms 118
2.5.11. S3 determine options for a request 118
2.5.12. S3 initiate a multipart upload 118
2.5.13. S3 add a part to a multipart upload 119

2.5.14. S3 list the parts of a multipart upload 120
2.5.15. S3 assemble the uploaded parts 121
2.5.16. S3 copy a multipart upload 122
2.5.17. S3 abort a multipart upload 123
2.5.18. S3 Hadoop interoperability 123
2.5.19. Additional Resources 123
2.6. ADDITIONAL RESOURCES 123

**CHAPTER 3. CEPH OBJECT GATEWAY AND THE SWIFT API 124**
3.1. PREREQUISITES 125
3.2. SWIFT API LIMITATIONS 125
3.3. CREATE A SWIFT USER 125
3.4. SWIFT AUTHENTICATING A USER 128
3.5. SWIFT CONTAINER OPERATIONS 128
3.5.1. Prerequisites 128
3.5.2. Swift container operations 129
3.5.3. Swift update a container's Access Control List (ACL) 129
3.5.4. Swift list containers 130
3.5.5. Swift list a container's objects 131
3.5.6. Swift create a container 132
3.5.7. Swift delete a container 133
3.5.8. Swift add or update the container metadata 134
3.6. SWIFT OBJECT OPERATIONS 134
3.6.1. Prerequisites 134
3.6.2. Swift object operations 134
3.6.3. Swift get an object 134
3.6.4. Swift create or update an object 135
3.6.5. Swift delete an object 136
3.6.6. Swift copy an object 136
3.6.7. Swift get object metadata 137
3.6.8. Swift add or update object metadata 137
3.7. SWIFT TEMPORARY URL OPERATIONS 138
3.7.1. Swift get temporary URL objects 138
3.7.2. Swift POST temporary URL keys 139
3.8. SWIFT MULTI-TENANCY CONTAINER OPERATIONS 139
3.9. ADDITIONAL RESOURCES 140

**APPENDIX A. S3 COMMON REQUEST HEADERS 141**

**APPENDIX B. S3 COMMON RESPONSE STATUS CODES 142**

**APPENDIX C. S3 UNSUPPORTED HEADER FIELDS 144**

**APPENDIX D. SWIFT REQUEST HEADERS 145**

**APPENDIX E. SWIFT RESPONSE HEADERS 146**

**APPENDIX F. EXAMPLES USING THE SECURE TOKEN SERVICE APIS 147**

# CHAPTER 1. CEPH OBJECT GATEWAY ADMINISTRATIVE API

As a developer, you can administer the Ceph Object Gateway by interacting with the RESTful application programing 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

Red Hat recommends using the command-line interface when configuring the Ceph Object Gateway.

The administrative API provides the following functionality:

- Authentication Requests
- **User Account Management**
  - Administrative User
  - Getting User Information
  - Creating
  - Modifying
  - Removing
  - Creating Subuser
  - Modifying Subuser
  - Removing Subuser
- **User Capabilities Management**
  - Adding
  - Removing
- **Key Management**
  - Creating
  - Removing
- **Bucket Management**
  - Getting Bucket Information
  - Checking Index
  - Removing
  - Linking
  - Unlinking

    - Policy
- **Object Management**
    - Removing
    - Policy
- **Quota Management**
    - Getting User
    - Setting User
    - Getting Bucket
    - Setting Bucket
- **Getting Usage Information**
- **Removing Usage Information**
- **Standard Error Responses**

## 1.1. PREREQUISITES

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

## 1.2. 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
```

## 1.3. 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.

**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://www.eu.apache.org/dist/httpcomponents/httpclient/binary/. 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 execute 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 upper case 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 examplary 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();
 }
 }
}
```

### Additional Resources

- See the *S3 Authentication section* in the *Red Hat Ceph Storage Developer Guide* for additional details.
- 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.

## 1.4. CREATING AN ADMINISTRATIVE 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.

## 1.5. GET USER INFORMATION

Get the user's information.

### Capabilities

```
users=read
```

### Syntax

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

**Table 1.1. Request Parameters**

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **uid** | The user for which the information is requested. | String | **foo_user** | Yes |

**Table 1.2. Response Entities**

| Name | Description | Type | Parent |
|---|---|---|---|
| **user** | A container for the user data information. | Container | N/A |
| **user_id** | The user ID. | String | **user** |
| **display_name** | Display name for the user. | String | **user** |
| **suspended** | True if the user is suspended. | Boolean | **user** |
| **max_buckets** | The maximum number of buckets to be owned by the user. | Integer | **user** |
| **subusers** | Subusers associated with this user account. | Container | **user** |
| **keys** | S3 keys associated with this user account. | Container | **user** |
| **swift_keys** | Swift keys associated with this user account. | Container | **user** |
| **caps** | User capabilities. | Container | **user** |

If successful, the response contains the user information.

### Special Error Responses

None.

## 1.6. CREATE A USER

Create a new user. By Default, a S3 key pair will be created automatically and returned in the response. If only one of **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
```

**Table 1.3. Request Parameters**

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **uid** | The user ID to be created. | String | **foo_user** | Yes |
| **display-name** | The display name of the user to be created. | String | **foo user** | Yes |
| **email** | The email address associated with the user. | String | **foo@bar.com** | No |
| **key-type** | Key type to be generated, options are: swift, s3 (default). | String | **s3** [**s3**] | No |
| **access-key** | Specify access key. | String | **ABCD0EF12GHIJ2K34LMN** | No |
| **secret-key** | Specify secret key. | String | **0AbCDEFg1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8** | No |
| **user-caps** | User capabilities. | String | **usage=read, write; users=read** | No |
| **generate-key** | Generate a new key pair and add to the existing keyring. | Boolean | True [True] | No |
| **max-buckets** | Specify the maximum number of buckets the user can own. | Integer | 500 [1000] | No |
| **suspended** | Specify whether the user should be suspended. | Boolean | False [False] | No |

**Table 1.4. Response Entities**

| Name | Description | Type | Parent |
|---|---|---|---|
| **user** | A container for the user data information. | Container | N/A |

| Name | Description | Type | Parent |
|---|---|---|---|
| **user_id** | The user ID. | String | **user** |
| **display_name** | Display name for the user. | String | **user** |
| **suspended** | True if the user is suspended. | Boolean | **user** |
| **max_buckets** | The maximum number of buckets to be owned by the user. | Integer | **user** |
| **subusers** | Subusers associated with this user account. | Container | **user** |
| **keys** | S3 keys associated with this user account. | Container | **user** |
| **swift_keys** | Swift keys associated with this user account. | Container | **user** |
| **caps** | User capabilities. | Container | **user** |

If successful, the response contains the user information.

Table 1.5. Special Error Responses

| Name | Description | Code |
|---|---|---|
| **UserExists** | Attempt to create existing user. | 409 Conflict |
| **InvalidAccessKey** | Invalid access key specified. | 400 Bad Request |
| **InvalidKeyType** | Invalid key type specified. | 400 Bad Request |
| **InvalidSecretKey** | Invalid secret key specified. | 400 Bad Request |
| **InvalidKeyType** | Invalid key type specified. | 400 Bad Request |
| **KeyExists** | Provided access key exists and belongs to another user. | 409 Conflict |
| **EmailExists** | Provided email address exists. | 409 Conflict |
| **InvalidCap** | Attempt to grant invalid admin capability. | 400 Bad Request |

### Additional Resources

- See the *Red Hat Ceph Storage Developer Guide* for creating subusers.

## 1.7. MODIFY A USER

Modify an existing user.

### Capabilities

```
`users=write`
```

### Syntax

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

**Table 1.6. Request Parameters**

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **uid** | The user ID to be modified. | String | **foo_user** | Yes |
| **display-name** | The display name of the user to be modified. | String | **foo user** | No |
| **email** | The email address to be associated with the user. | String | **foo@bar.com** | No |
| **generate-key** | Generate a new key pair and add to the existing keyring. | Boolean | True [False] | No |
| **access-key** | Specify access key. | String | **ABCD0EF12GHIJ2K34LMN** | No |
| **secret-key** | Specify secret key. | String | **0AbCDEFg1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8** | No |
| **key-type** | Key type to be generated, options are: swift, s3 (default). | String | **s3** | No |
| **user-caps** | User capabilities. | String | **usage=read, write; users=read** | No |
| **max-buckets** | Specify the maximum number of buckets the user can own. | Integer | 500 [1000] | No |
| **suspended** | Specify whether the user should be suspended. | Boolean | False [False] | No |

Table 1.7. Response Entities

| Name | Description | Type | Parent |
|---|---|---|---|
| **user** | A container for the user data information. | Container | N/A |
| **user_id** | The user ID. | String | **user** |
| **display_name** | Display name for the user. | String | **user** |
| **suspended** | True if the user is suspended. | Boolean | **user** |
| **max_buckets** | The maximum number of buckets to be owned by the user. | Integer | **user** |
| **subusers** | Subusers associated with this user account. | Container | **user** |
| **keys** | S3 keys associated with this user account. | Container | **user** |
| **swift_keys** | Swift keys associated with this user account. | Container | **user** |
| **caps** | User capabilities. | Container | **user** |

If successful, the response contains the user information.

Table 1.8. Special Error Responses

| Name | Description | Code |
|---|---|---|
| **InvalidAccessKey** | Invalid access key specified. | 400 Bad Request |
| **InvalidKeyType** | Invalid key type specified. | 400 Bad Request |
| **InvalidSecretKey** | Invalid secret key specified. | 400 Bad Request |
| **KeyExists** | Provided access key exists and belongs to another user. | 409 Conflict |
| **EmailExists** | Provided email address exists. | 409 Conflict |
| **InvalidCap** | Attempt to grant invalid admin capability. | 400 Bad Request |

Additional Resources

- See the *Red Hat Ceph Storage Developer Guide* for modifying subusers.

## 1.8. REMOVE A USER

Remove an existing user.

### Capabilities

```
`users=write`
```

### Syntax

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

**Table 1.9. Request Parameters**

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **uid** | The user ID to be removed. | String | **foo_user** | Yes. |
| **purge-data** | When specified the buckets and objects belonging to the user will also be removed. | Boolean | True | No |

### Response Entities

None.

### Special Error Responses

None.

### Additional Resources

- See *Red Hat Ceph Storage Developer Guide* for removing subusers.

## 1.9. CREATE A SUBUSER

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 will be automatically generated.

### Capabilities

```
`users=write`
```

### Syntax

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

Table 1.10. Request Parameters

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **uid** | The user ID under which a subuser is to be created. | String | **foo_user** | Yes |
| **subuser** | Specify the subuser ID to be created. | String | **sub_foo** | Yes (or **gen-subuser**) |
| **gen-subuser** | Specify the subuser ID to be created. | String | **sub_foo** | Yes (or **subuser**) |
| **secret-key** | Specify secret key. | String | **0AbCDEFg1h2i34JklM5nop6QrSTUVWxyzaBC7D8** | No |
| **key-type** | Key type to be generated, options are: swift (default), s3. | String | **swift** [**swift**] | No |
| **access** | Set access permissions for sub-user, should be one of **read, write, readwrite, full**. | String | **read** | No |
| **generate-secret** | Generate the secret key. | Boolean | True [False] | No |

Table 1.11. Response Entities

| Name | Description | Type | Parent |
|---|---|---|---|
| **subusers** | Subusers associated with the user account. | Container | N/A |
| **id** | Subuser ID. | String | **subusers** |
| **permissions** | Subuser access to user account. | String | **subusers** |

If successful, the response contains the subuser information.

Table 1.12. Special Error Responses

| Name | Description | Code |
|---|---|---|
| **SubuserExists** | Specified subuser exists. | 409 Conflict |
| **InvalidKeyType** | Invalid key type specified. | 400 Bad Request |
| **InvalidSecretKey** | Invalid secret key specified. | 400 Bad Request |
| **InvalidAccess** | Invalid subuser access specified. | 400 Bad Request |

## 1.10. MODIFY A SUBUSER

Modify an existing subuser.

### Capabilities

```
`users=write`
```

### Syntax

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

Table 1.13. Request Parameters

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **uid** | The user ID under which the subuser is to be modified. | String | **foo_user** | Yes |
| **subuser** | The subuser ID to be modified. | String | **sub_foo** | Yes |
| **generate-secret** | Generate a new secret key for the subuser, replacing the existing key. | Boolean | True [False] | No |
| **secret** | Specify secret key. | String | **0AbCDEFg1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8** | No |
| **key-type** | Key type to be generated, options are: swift (default), s3. | String | **swift** [**swift**] | No |

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **access** | Set access permissions for sub-user, should be one of **read, write, readwrite, full**. | String | **read** | No |

Table 1.14. Response Entities

| Name | Description | Type | Parent |
|---|---|---|---|
| **subusers** | Subusers associated with the user account. | Container | N/A |
| **id** | Subuser ID. | String | **subusers** |
| **permissions** | Subuser access to user account. | String | **subusers** |

If successful, the response contains the subuser information.

Table 1.15. Special Error Responses

| Name | Description | Code |
|---|---|---|
| **InvalidKeyType** | Invalid key type specified. | 400 Bad Request |
| **InvalidSecretKey** | Invalid secret key specified. | 400 Bad Request |
| **InvalidAccess** | Invalid subuser access specified. | 400 Bad Request |

## 1.11. REMOVE A SUBUSER

Remove an existing subuser.

### Capabilities

```
`users=write`
```

### Syntax

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

**Table 1.16. Request Parameters**

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **uid** | The user ID under which the subuser is to be removed. | String | **foo_user** | Yes |
| **subuser** | The subuser ID to be removed. | String | **sub_foo** | Yes |
| **purge-keys** | Remove keys belonging to the subuser. | Boolean | True [True] | No |

### Response Entities

None.

### Special Error Responses

None.

## 1.12. ADD CAPABILITIES TO A USER

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
```

**Table 1.17. Request Parameters**

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **uid** | The user ID to add an administrative capability to. | String | **foo_user** | Yes |
| **user-caps** | The administrative capability to add to the user. | String | **usage=read, write** | Yes |

**Table 1.18. Response Entities**

| Name | Description | Type | Parent |
|---|---|---|---|
| **user** | A container for the user data information. | Container | N/A |
| **user_id** | The user ID. | String | **user** |
| **caps** | User capabilities. | Container | **user** |

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

**Table 1.19. Special Error Responses**

| Name | Description | Code |
|---|---|---|
| **InvalidCap** | Attempt to grant invalid admin capability. | 400 Bad Request |

## 1.13. REMOVE CAPABILITIES FROM A USER

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
```

**Table 1.20. Request Parameters**

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **uid** | The user ID to remove an administrative capability from. | String | **foo_user** | Yes |
| **user-caps** | The administrative capabilities to remove from the user. | String | **usage=read, write** | Yes |

**Table 1.21. Response Entities**

| Name | Description | Type | Parent |
|---|---|---|---|
| **user** | A container for the user data information. | Container | N/A |

| Name | Description | Type | Parent |
|---|---|---|---|
| **user_id** | The user ID. | String | **user** |
| **caps** | User capabilities. | Container | **user** |

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

**Table 1.22. Special Error Responses**

| Name | Description | Code |
|---|---|---|
| **InvalidCap** | Attempt to remove an invalid admin capability. | 400 Bad Request |
| **NoSuchCap** | User does not possess specified capability. | 404 Not Found |

## 1.14. CREATE A KEY

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
```

**Table 1.23. Request Parameters**

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **uid** | The user ID to receive the new key. | String | **foo_user** | Yes |

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **subuser** | The subuser ID to receive the new key. | String | **sub_foo** | No |
| **key-type** | Key type to be generated, options are: swift, s3 (default). | String | **s3** [**s3**] | No |
| **access-key** | Specify the access key. | String | **AB01C2D3EF45G6H7IJ8K** | No |
| **secret-key** | Specify the secret key. | String | **0ab/CdeFGhij1klmnopqRSTUv1WxyZabcDEFgHij** | No |
| **generate-key** | Generate a new key pair and add to the existing keyring. | Boolean | True [**True**] | No |

Table 1.24. Response Entities

| Name | Description | Type | Parent |
|---|---|---|---|
| **keys** | Keys of type created associated with this user account. | Container | N/A |
| **user** | The user account associated with the key. | String | **keys** |
| **access-key** | The access key. | String | **keys** |
| **secret-key** | The secret key | String | **keys** |

Table 1.25. Special Error Responses

| Name | Description | Code |
|---|---|---|
| **InvalidAccessKey** | Invalid access key specified. | 400 Bad Request |
| **InvalidKeyType** | Invalid key type specified. | 400 Bad Request |
| **InvalidSecretKey** | Invalid secret key specified. | 400 Bad Request |

| Name | Description | Code |
|---|---|---|
| **InvalidKeyType** | Invalid key type specified. | 400 Bad Request |
| **KeyExists** | Provided access key exists and belongs to another user. | 409 Conflict |

## 1.15. REMOVE A KEY

Remove an existing key.

### Capabilities

```
`users=write`
```

### Syntax

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

**Table 1.26. Request Parameters**

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **access-key** | The S3 access key belonging to the S3 key pair to remove. | String | **AB01C2D3EF45G6H7IJ8K** | Yes |
| **uid** | The user to remove the key from. | String | **foo_user** | No |
| **subuser** | The subuser to remove the key from. | String | **sub_foo** | No |
| **key-type** | Key type to be removed, options are: swift, s3. NOTE: Required to remove swift key. | String | **swift** | No |

### Special Error Responses

None.

### Response Entities

None.

## 1.16. BUCKET NOTIFICATIONS

As a storage administrator, you can use these APIs to provide configuration and control interfaces for the bucket notification mechanism. The API topics are named objects that contain the definition of a

specific endpoint. Bucket notifications associate topics with a specific bucket. The *S3 bucket operations* section gives more details on bucket notifications.

NOTE

In all topic actions, the parameters are URL encoded, and sent in the message body using **application/x-www-form-urlencoded** content type.

NOTE

Any bucket notification already associated with the topic needs to be re-created for the topic update to take effect.

### 1.16.1. Prerequisites

- Create bucket notifications on the Ceph Object Gateway.

### 1.16.2. Creating a topic

You can create topics 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.

**Procedure**

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]
```

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.
- HTTP endpoint:
  - **URL**: http[s]://*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 can only be provided with 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 acknowledgement is required, as messages may persist in the broker before being delivered into their final destination. Three acknowledgement 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 does 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.

NOTE

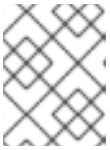
The **topic-name** is used for the AMQP topic.

- Kafka endpoint:
  - **URL**: kafka://[*USER*: *PASSWORD* @] *FQDN*[: *PORT*].
  - If **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]. If not,topic creation request will be rejected.
  - User and password may only be provided together with **use-ssl**, if not, connection to the broker would fail.
  - **port defaults to**: 9092.
  - **kafka-ack-level**: no end to end acknowledgement required, as messages may persist in the broker before being delivered into their final destination. Two acknowledgement 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.

2. Create a response in the following format:

**Syntax**

```
<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:

**Syntax**

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

### 1.16.3. Getting topic information

Returns information about specific topic. This 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>
         </EndPoint>
         <TopicArn>
         </TopicArn>
         <OpaqueData>
         </OpaqueData>
       </Topic>
     </GetTopicResult>
     <ResponseMetadata>
       <RequestId>
       </RequestId>
     </ResponseMetadata>
   </GetTopicResponse>
   ```

These are the tags and their definitions:

- **User**: Name of the user that created the topic.
- **Name**: Name of the topic.
- **EndpointAddress**: The endpoint URL. If the endpoint URL contains user and password information, request must be made over HTTPS. If not, the topic get request will be rejected.
- **EndPointArgs**: The endpoint arguments.
- **EndpointTopic**: The topic name that will be sent to the endpoint, can be different than the above topic name.
- **TopicArn**: Topic ARN.

### 1.16.4. Listing topics

List the topics that the 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:

```
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. If not, topic list request is rejected.

### 1.16.5. Deleting topics

Removing a deleted topic results with no operation and 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**

1. 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>
```

### 1.16.6. 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 a JSON format.

#### Example

```
{"Records":[
    {
        "eventVersion":"2.1",
        "eventSource":"ceph:s3",
        "awsRegion":"us-east-1",
        "eventTime":"2019-11-22T13:47:35.124724Z",
        "eventName":"s3: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.
- **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_amz_id_2**: 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.

**Additional Resources**

- See the *Event Message Structure* for more information.
- See the *Supported event types* section of the *Red Hat Ceph Storage Developer Guide* for more information.

### 1.16.7. Supported event types

The following event types are supported:

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

### 1.16.8. Additional Resources

- See the *Creating bucket notifications* section is the *Red Hat Ceph Storage Object Gateway Configuration and Administration Guide* for more details.

## 1.17. 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
```

**Table 1.27. Request Parameters**

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **bucket** | The bucket to return info on. | String | **foo_bucket** | No |
| **uid** | The user to retrieve bucket information for. | String | **foo_user** | No |
| **stats** | Return bucket statistics. | Boolean | True [False] | No |

**Table 1.28. Response Entities**

| Name | Description | Type | Parent |
|---|---|---|---|
| **stats** | Per bucket information. | Container | N/A |
| **buckets** | Contains a list of one or more bucket containers. | Container | **bucket** |
| Container for single bucket information. | Container | **buckets** | **name** |
| The name of the bucket. | String | **bucket** | **pool** |
| The pool the bucket is stored in. | String | **bucket** | **id** |
| The unique bucket ID. | String | **bucket** | **marker** |
| Internal bucket tag. | String | **bucket** | **owner** |
| The user ID of the bucket owner. | String | **bucket** | **usage** |
| Storage usage information. | Container | **bucket** | **index** |

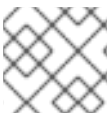
If successful the request returns a buckets container containing the desired bucket information.

**Table 1.29. Special Error Responses**

| Name | Description | Code |
|---|---|---|
| **IndexRepairFailed** | Bucket index repair failed. | 409 Conflict |

## 1.18. 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
```

**Table 1.30. Request Parameters**

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **bucket** | The bucket to return info on. | String | **foo_bucket** | Yes |
| **check-objects** | Check multipart object accounting. | Boolean | True [False] | No |
| **fix** | Also fix the bucket index when checking. | Boolean | False [False] | No |

**Table 1.31. Response Entities**

| Name | Description | Type |
|---|---|---|
| **index** | Status of bucket index. | String |

**Table 1.32. Special Error Responses**

| Name | Description | Code |
|---|---|---|
| **IndexRepairFailed** | Bucket index repair failed. | 409 Conflict |

## 1.19. REMOVE A BUCKET

Removes an existing bucket.

### Capabilities

```
`buckets=write`
```

### Syntax

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

**Table 1.33. Request Parameters**

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **bucket** | The bucket to remove. | String | **foo_bucket** | Yes |
| **purge-objects** | Remove a buckets objects before deletion. | Boolean | True [False] | No |

### Response Entities

None.

**Table 1.34. Special Error Responses**

| Name | Description | Code |
|---|---|---|
| **BucketNotEmpty** | Attempted to delete non-empty bucket. | 409 Conflict |
| **ObjectRemovalFailed** | Unable to remove objects. | 409 Conflict |

## 1.20. 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
```

**Table 1.35. Request Parameters**

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **bucket** | The bucket to unlink. | String | **foo_bucket** | Yes |
| **uid** | The user ID to link the bucket to. | String | **foo_user** | Yes |

**Table 1.36. Response Entities**

| Name | Description | Type | Parent |
|---|---|---|---|
| **bucket** | Container for single bucket information. | Container | N/A |
| **name** | The name of the bucket. | String | **bucket** |
| **pool** | The pool the bucket is stored in. | String | **bucket** |
| **id** | The unique bucket ID. | String | **bucket** |
| **marker** | Internal bucket tag. | String | **bucket** |
| **owner** | The user ID of the bucket owner. | String | **bucket** |
| **usage** | Storage usage information. | Container | **bucket** |
| **index** | Status of bucket index. | String | **bucket** |

Table 1.37. Special Error Responses

| Name | Description | Code |
|---|---|---|
| **BucketUnlinkFailed** | Unable to unlink bucket from specified user. | 409 Conflict |
| **BucketLinkFailed** | Unable to link bucket to specified user. | 409 Conflict |

## 1.21. UNLINK A BUCKET

Unlink a bucket from a specified user. Primarily useful for changing bucket ownership.

### Capabilities

```
`buckets=write`
```

### Syntax

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

Table 1.38. Request Parameters

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **bucket** | The bucket to unlink. | String | **foo_bucket** | Yes |
| **uid** | The user ID to unlink the bucket from. | String | **foo_user** | Yes |

### Response Entities

None.

Table 1.39. Special Error Responses

| Name | Description | Code |
|---|---|---|
| **BucketUnlinkFailed** | Unable to unlink bucket from specified user. | 409 Conflict |

## 1.22. GET A BUCKET OR OBJECT POLICY

Read the policy of an object or bucket.

### Capabilities

```
`buckets=read`
```

### Syntax

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

Table 1.40. Request Parameters

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **bucket** | The bucket to read the policy from. | String | **foo_bucket** | Yes |
| **object** | The object to read the policy from. | String | **foo.txt** | No |

Table 1.41. Response Entities

| Name | Description | Type | Parent |
|---|---|---|---|
| **policy** | Access control policy. | Container | N/A |

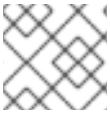
If successful, returns the object or bucket policy

**Table 1.42. Special Error Responses**

| Name | Description | Code |
|---|---|---|
| **IncompleteBody** | Either bucket was not specified for a bucket policy request or bucket and object were not specified for an object policy request. | 400 Bad Request |

## 1.23. REMOVE AN OBJECT

Remove an existing object.



NOTE

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
```

**Table 1.43. Request Parameters**

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **bucket** | The bucket containing the object to be removed. | String | **foo_bucket** | Yes |
| **object** | The object to remove. | String | **foo.txt** | Yes |

### Response Entities

None.

**Table 1.44. Special Error Responses**

| Name | Description | Code |
|---|---|---|
| **NoSuchObject** | Specified object does not exist. | 404 Not Found |
| **ObjectRemovalFailed** | Unable to remove objects. | 409 Conflict |

## 1.24. QUOTAS

The administrative Operations API enables you to set quotas on users and on bucket 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**.

## 1.25. 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
```

## 1.26. 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.

## 1.27. 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
```

**Table 1.45. Request Parameters**

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **bucket** | The bucket to return info on. | String | **foo_bucket** | No |
| **uid** | The user to retrieve bucket information for. | String | **foo_user** | No |
| **stats** | Return bucket statistics. | Boolean | True [False] | No |

**Table 1.46. Response Entities**

| Name | Description | Type | Parent |
|---|---|---|---|
| **stats** | Per bucket information. | Container | N/A |
| **buckets** | Contains a list of one or more bucket containers. | Container | **bucket** |
| Container for single bucket information. | Container | **buckets** | **name** |
| The name of the bucket. | String | **bucket** | **pool** |
| The pool the bucket is stored in. | String | **bucket** | **id** |
| The unique bucket ID. | String | **bucket** | **marker** |

| Name | Description | Type | Parent |
|---|---|---|---|
| Internal bucket tag. | String | **bucket** | **owner** |
| The user ID of the bucket owner. | String | **bucket** | **usage** |
| Storage usage information. | Container | **bucket** | **index** |

If successful the request returns a buckets container containing the desired bucket information.

**Table 1.47. Special Error Responses**

| Name | Description | Code |
|---|---|---|
| **IndexRepairFailed** | Bucket index repair failed. | 409 Conflict |

## 1.28. 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.

## 1.29. GET USAGE INFORMATION

Requesting bandwidth usage information.

### Capabilities

```
`usage=read`
```

### Syntax

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

**Table 1.48. Request Parameters**

| Name | Description | Type | Required |
|---|---|---|---|
| **uid** | The user for which the information is requested. | String. | Yes |
| **start** | Date and (optional) time that specifies the start time of the requested data. E.g., **2012-09-25 16:00:00** | String | No |
| **end** | Date and (optional) time that specifies the end time of the requested data (non-inclusive). E.g., **2012-09-25 16:00:00** | String | No |
| **show-entries** | Specifies whether data entries should be returned. | Boolean | No |
| **show-summary** | Specifies whether data summary should be returned. | Boolean | No |

Table 1.49. Response Entities

| Name | Description | Type |
|---|---|---|
| **usage** | A container for the usage information. | Container |
| **entries** | A container for the usage entries information. | Container |
| **user** | A container for the user data information. | Container |
| **owner** | The name of the user that owns the buckets. | String |
| **bucket** | The bucket name. | String |
| **time** | Time lower bound for which data is being specified (rounded to the beginning of the first relevant hour). | String |
| **epoch** | The time specified in seconds since **1/1/1970**. | String |
| **categories** | A container for stats categories. | Container |
| **entry** | A container for stats entry. | Container |
| **category** | Name of request category for which the stats are provided. | String |
| **bytes_sent** | Number of bytes sent by the Ceph Object Gateway. | Integer |
| **bytes_received** | Number of bytes received by the Ceph Object Gateway. | Integer |

| Name | Description | Type |
|---|---|---|
| **ops** | Number of operations. | Integer |
| **successful_ops** | Number of successful operations. | Integer |
| **summary** | A container for stats summary. | Container |
| **total** | A container for stats summary aggregated total. | Container |

If successful, the response contains the requested information.

## 1.30. REMOVE USAGE INFORMATION

Remove usage information. With no dates specified, removes all usage information.

### Capabilities

```
`usage=write`
```

### Syntax

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

**Table 1.50. Request Parameters**

| Name | Description | Type | Example | Required |
|---|---|---|---|---|
| **uid** | The user for which the information is requested. | String | **foo_user** | No |
| **start** | Date and (optional) time that specifies the start time of the requested data. | String | **2012-09-25 16:00:00** | No |
| **end** | Date and (optional) time that specifies the end time of the requested data (none inclusive). | String | **2012-09-25 16:00:00** | No |
| **remove-all** | Required when **uid** is not specified, in order to acknowledge multi-user data removal. | Boolean | True [False] | No |

## 1.31. STANDARD ERROR RESPONSES

The following table details standard error responses and their descriptions.

| Name | Description | Code |
|---|---|---|
| **AccessDenied** | Access denied. | 403 Forbidden |
| **InternalError** | Internal server error. | 500 Internal Server Error |
| **NoSuchUser** | User does not exist. | 404 Not Found |
| **NoSuchBucket** | Bucket does not exist. | 404 Not Found |
| **NoSuchKey** | No such access key. | 404 Not Found |

# CHAPTER 2. CEPH OBJECT GATEWAY AND THE S3 API

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

## 2.1. PREREQUISITES

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

## 2.2. S3 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 Amazon S3:** Individual Amazon S3 objects can range in size from a minimum of 0B to a maximum of 5TB. The largest object that can be uploaded in a single **PUT** is 5GB. For objects larger than 100MB, you should 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 here is 200-300 bytes plus the length of the object name. Versioned objects consume 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.

**Additional Resources**

- See the *Red Hat Ceph Storage Developer Guide* for details on the *unsupported header fields*.

## 2.3. 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.

### 2.3.1. Prerequisites

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

### 2.3.2. S3 authentication

Requests to the Ceph Object Gateway can be either authenticated or unauthenticated. Ceph Object Gateway assumes 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 simply 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 above 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 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 using a SHA-1 hashing algorithm.
4. Encode the **hmac** result as base-64.

#### Normalize header

To 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 beginning 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.

**Additional Resources**

- For additional details, consult the Signing and Authenticating REST Requests section of Amazon Simple Storage Service documentation.

### 2.3.3. S3 server-side encryption

The Ceph Object Gateway supports server-side encryption of uploaded objects for the S3 application programing 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.

NOTE

Red Hat does NOT support S3 object encryption of Static Large Object (SLO) or Dynamic Large Object (DLO).

IMPORTANT

To use encryption, client requests **MUST** send requests over an SSL connection. Red Hat does not support S3 encryption from a client unless the Ceph Object Gateway uses SSL. However, for testing purposes, administrators may disable SSL during testing by setting the **rgw_crypt_require_ssl** configuration setting to **false** at runtime, setting it to **false** in the Ceph configuration file and restarting the gateway instance, or setting it to **false** in the Ansible configuration files and replaying the Ansible playbooks for the Ceph Object Gateway.

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

For information about how to configure HTTP with server-side encryption, see the *Additional Resources* section below.

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.

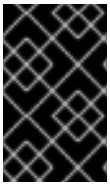
Ceph Object Gateway implements the customer-provided key behavior in the S3 API according to the Amazon SSE-C specification.

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

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.



**IMPORTANT**

Currently, the only tested key management implementation uses OpenStack Barbican. However, OpenStack Barbican is a Technology Preview and is not supported for use in production systems.

**Additional Resources**

- Amazon SSE-C
- Amazon SSE-KMS
- *Configuring server-side encryption*

## 2.3.4. 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:

**Table 2.1. 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. |

### 2.3.5. Preparing access to the Ceph Object Gateway using S3

You have to follow some pre-requisites on the Ceph Object Gateway node before attempting to access the gateway server.

**WARNING**

**DO NOT** modify the Ceph configuration file to use port **80** and let **Civetweb** use the default Ansible configured port of **8080**.

**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 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 the *Object Gateway Configuration and Administration Guide*.
   You can also set up the gateway node for local DNS caching. To do so, execute 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 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 as mentioned in the *Object Gateway Configuration and Administration Guide* and copy the generated **access_key** and **secret_key**. You will need these keys for **S3** access and subsequent bucket management tasks.

### 2.3.6. Accessing the Ceph Object Gateway using Ruby AWS S3

You can use Ruby programming language along with **aws-s3** gem for **S3** access. Execute the steps mentioned below on the node 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 accessing the Ceph Object Gateway.
- Internet access.

**Procedure**

1. Install the **ruby** package:

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

**NOTE**

The above command will install **ruby** and it's 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. Replace *MY-ACCESS-KEY* and *MY-SECRET-KEY* with the **access_key** and **secret_key** that was generated when you created the **radosgw** user for **S3** access as mentioned in the *Red Hat Ceph Storage Object Gateway Configuration and Administration Guide*.

**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 will be **0**.

8. Create a new 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 bucket `my-new-bucket1`was created successfully.

11. Create a new 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 new 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 will create 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 will look something like this:

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

20. Create a new 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 will return **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 above mentioned **del_empty_bucket.rb** file accordingly before trying to delete empty buckets.

23. Create a new file for deleting non-empty 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 will return **0** as output.

26. Create a new 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 will delete the object **hello.txt**.

### 2.3.7. Accessing the Ceph Object Gateway using Ruby AWS SDK

You can use the Ruby programming language along with **aws-sdk** gem for **S3** access. Execute the steps mentioned below on the node used for accessing the Ceph Object Gateway server with **Ruby AWS::SDK**.

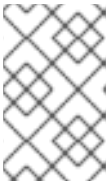
#### Prerequisites

- 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

The above command will install **ruby** and it's 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 was generated when you created the **radosgw** user for **S3** access as mentioned in the *Red Hat Ceph Storage Object Gateway Configuration and Administration Guide*.

**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 new 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 new 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 new 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 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'

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 new 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.

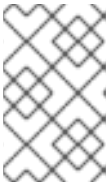
21. Make the file executable:

```
[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 above mentioned **del_empty_bucket.rb** file accordingly before trying to delete empty buckets.

23. Create a new 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 new 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**.

### 2.3.8. Accessing the Ceph Object Gateway using PHP

You can 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 given below are tested against **php v5.4.16** and **aws-sdk v2.8.24**. **DO NOT** use the latest version of **aws-sdk** for **php** as it requires **php >= 5.5+**.**php 5.5** is not available in the default repositories of **RHEL 7**. If you want to use **php 5.5**, you will have to enable **epel** and other third party repositories. Also, the configuration options for **php 5.5** and latest version of **aws-sdk** are different.

**Prerequisites**

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

**Procedure**

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 *MY_ACCESS_KEY* and *MY_SECRET_KEY* with the **access_key** and **secret_key** that was generated when creating the **radosgw** user for **S3** access as mentioned in the *Red Hat Ceph Storage Object Gateway Configuration and Administration Guide*. 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**

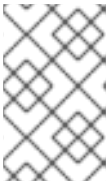
```
<?php

include 'conn.php';

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

Save the file and exit the editor.

18. Run the file:

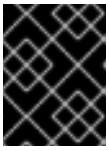
```
[user@dev ~]$ php -f del_empty_bucket.php | echo $?
```

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



NOTE

Edit the **create_bucket.php** file to create empty buckets, for example: **my-new-bucket4**, **my-new-bucket5**. Next, edit the above mentioned **del_empty_bucket.php** file accordingly before trying to delete empty buckets.



IMPORTANT

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

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**.

### 2.3.9. Secure Token Service

The Amazon Web Services' Secure Token Service (STS) returns a set of temporary security credentials for authenticating users. The Ceph Object Gateway implements a subset of the STS application programming interfaces (APIs) to provide temporary credentials for identity and access management (IAM). Using these temporary credentials authenticates S3 calls by utilizing the STS engine in the Ceph Object Gateway. You can restrict temporary credentials even further by using an IAM policy, which is a parameter passed to the STS APIs.

**Additional Resources**

- Amazon Web Services Secure Token Service welcome page.
- See the *Configuring and using STS Lite with Keystone* section of the *Red Hat Ceph Storage Developer Guide* for details on STS Lite and Keystone.
- See the *Working around the limitations of using STS Lite with Keystone* section of the *Red Hat Ceph Storage Developer Guide* for details on the limitations of STS Lite and Keystone.

#### 2.3.9.1. The Secure Token Service application programming interfaces

The Ceph Object Gateway implements the following Secure Token Service (STS) application programming interfaces (APIs):

##### AssumeRole

This API returns a set of temporary credentials for cross-account access. These temporary credentials allow for both, permission policies attached with *Role* and policies attached with *AssumeRole* API. The **RoleArn** and the **RoleSessionName** request parameters are required, but the other request parameters are optional.

**RoleArn**

**Description**

The role to assume for the Amazon Resource Name (ARN) with a length of 20 to 2048 characters.

**Type**

*String*

**Required**

Yes

**RoleSessionName**

**Description**

Identifying the role session name to assume. The role session name can uniquely identify a session when different principals or different reasons assume a role. This parameter's value has a length of 2 to 64 characters. The **=**, **,**, **.**, **@**, and **-** characters are allowed, but no spaces allowed.

**Type**

*String*

**Required**

Yes

**Policy**

**Description**

An identity and access management policy (IAM) in a JSON format for use in an inline session. This parameter's value has a length of 1 to 2048 characters.

**Type**

*String*

**Required**

No

**DurationSeconds**

**Description**

The duration of the session in seconds, with a minimum value of **900** seconds to a maximum value of **43200** seconds. The default value is **3600** seconds.

**Type**

*Integer*

**Required**

No

**ExternalId**

**Description**

When assuming a role for another account, provide the unique external identifier if available. This parameter's value has a length of 2 to 1224 characters.

**Type**

*String*

**Required**

No

**SerialNumber**

**Description**

A user's identification number from their associated multi-factor authentication (MFA) device. The parameter's value can be the serial number of a hardware device or a virtual device, with a length of 9 to 256 characters.

**Type**

*String*

**Required**

No

**TokenCode**

**Description**

The value generated from the multi-factor authentication (MFA) device, if the trust policy requires a MFA. If a MFA device is required, and if this parameter's value is empty or expired, then *AssumeRole* call returns an "access denied" error message. This parameter's value has a fixed length of 6 characters.

**Type**

*String*

**Required**

No

### AssumeRoleWithWebIdentity

This API returns a set of temporary credentials for users who have been authenticated by an application, such as OpenID Connect or OAuth 2.0 Identity Provider. The **RoleArn** and the **RoleSessionName** request parameters are required, but the other request parameters are optional.

**RoleArn**

**Description**

The role to assume for the Amazon Resource Name (ARN) with a length of 20 to 2048 characters.

**Type**

*String*

**Required**

Yes

### RoleSessionName

**Description**

Identifying the role session name to assume. The role session name can uniquely identify a session when different principals or different reasons assume a role. This parameter's value has a length of 2 to 64 characters. The **=**, **,**, **.**, **@**, and **-** characters are allowed, but no spaces allowed.

**Type**

*String*

**Required**

Yes

### Policy

**Description**

An identity and access management policy (IAM) in a JSON format for use in an inline session. This parameter's value has a length of 1 to 2048 characters.

**Type**

*String*

**Required**

No

### DurationSeconds

**Description**

The duration of the session in seconds, with a minimum value of **900** seconds to a maximum value of **43200** seconds. The default value is **3600** seconds.

**Type**

*Integer*

**Required**

No

### ProviderId

**Description**

The fully qualified host component of the domain name from the identity provider. This parameter's value is only valid for OAuth 2.0 access tokens, with a length of 4 to 2048 characters.

**Type**

*String*

**Required**

No

### WebIdentityToken

**Description**

The OpenID Connect identity token or OAuth 2.0 access token provided from an identity provider. This parameter's value has a length of 4 to 2048 characters.

**Type**

*String*

**Required**

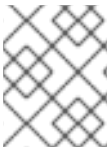
No

**Additional Resources**

- See the *Examples using the Secure Token Service APIs* section of the *Red Hat Ceph Storage Developer Guide* for more details.
- Amazon Web Services Security Token Service, the *AssumeRole* action.
- Amazon Web Services Security Token Service, the *AssumeRoleWithWebIdentity* action.

### 2.3.9.2. Configuring the Secure Token Service

Configure the Secure Token Service (STS) for use with the Ceph Object Gateway using Ceph Ansible.



**NOTE**

The S3 and STS APIs co-exist in the same namespace, and both can be accessed from the same endpoint in the Ceph Object Gateway.

**Prerequisites**

- A Ceph Ansible administration node.
- A running Red Hat Ceph Storage cluster.
- A running Ceph Object Gateway.

**Procedure**

1. Open for editing the **group_vars/rgws.yml** file.
   a. Add the following lines:

   ```
   rgw_sts_key = STS_KEY
   rgw_s3_auth_use_sts = true
   ```

   Replace:

   - *STS_KEY* with the key used to encrypted the session token.
2. Save the changes to the **group_vars/rgws.yml** file.
3. Rerun the appropriate Ceph Ansible playbook:
   a. **Bare-metal** deployments:

```
[user@admin ceph-ansible]$ ansible-playbook site.yml --limit rgws
```

b. **Container** deployments:

```
[user@admin ceph-ansible]$ ansible-playbook site-docker.yml --limit rgws
```

**Additional Resources**

- See the *Secure Token Service application programming interfaces* section in the *Red Hat Ceph Storage Developer Guide* for more details on the STS APIs.

### 2.3.9.3. Creating a user for an OpenID Connect provider

To establish trust between the Ceph Object Gateway and the OpenID Connect Provider create a user entity and a role trust policy.

**Prerequisites**

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

**Procedure**

1. Create a new Ceph user:

   **Syntax**

   ```
   radosgw-admin --uid USER_NAME --display-name "DISPLAY_NAME" --access_key
   USER_NAME --secret SECRET user create
   ```

   **Example**

   ```
   [user@rgw ~]$ radosgw-admin --uid TESTER --display-name "TestUser" --access_key
   TESTER --secret test123 user create
   ```

2. Configure the Ceph user capabilities:

   **Syntax**

   ```
   radosgw-admin caps add --uid="USER_NAME" --caps="oidc-provider=*"
   ```

   **Example**

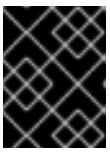
   ```
   [user@rgw ~]$ radosgw-admin caps add --uid="TESTER" --caps="oidc-provider=*"
   ```

3. Add a condition to the role trust policy using the Secure Token Service (STS) API:

   **Syntax**

   ```
   "{\"Version\":\"2020-01-17\",\"Statement\":[{\"Effect\":\"Allow\",\"Principal\":{\"Federated\":
   [\"arn:aws:iam:::oidc-provider/IDP_URL\"]},\"Action\":
   [\"sts:AssumeRoleWithWebIdentity\"],\"Condition\":{\"StringEquals\":{\"IDP_URL
   ```

```
:app_id\":\"AUD_FIELD\"\}\}\}\]\}"
```



**IMPORTANT**

The **app_id** in the syntax example above must match the *AUD_FIELD* field of the incoming token.

**Additional Resources**

- See the *Obtaining the Root CA Thumbprint for an OpenID Connect Identity Provider* article on Amazon's website.
- See the *Secure Token Service application programming interfaces* section in the *Red Hat Ceph Storage Developer Guide* for more details on the STS APIs.
- See the *Examples using the Secure Token Service APIs* section of the *Red Hat Ceph Storage Developer Guide* for more details.

### 2.3.9.4. Obtaining a thumbprint of an OpenID Connect provider

To get the OpenID Connect provider's (IDP) configuration document.

**Prerequisites**

- Installation of the **openssl** and **curl** packages.

**Procedure**

1. Get the configuration document from the IDP's URL:

   **Syntax**

   ```
   curl -k -v \
       -X GET \
       -H "Content-Type: application/x-www-form-urlencoded" \
       "IDP_URL:8000/CONTEXT/realms/REALM/.well-known/openid-configuration" \
     | jq .
   ```

   **Example**

   ```
   [user@client ~]$ curl -k -v \
       -X GET \
       -H "Content-Type: application/x-www-form-urlencoded" \
       "http://www.example.com:8000/auth/realms/quickstart/.well-known/openid-configuration" \
     | jq .
   ```

2. Get the IDP certificate:

   **Syntax**

   ```
   curl -k -v \
       -X GET \
       -H "Content-Type: application/x-www-form-urlencoded" \
   ```

```
"IDP_URL/CONTEXT/realms/REALM/protocol/openid-connect/certs" \
| jq .
```

**Example**

```
[user@client ~]$ curl -k -v \
    -X GET \
    -H "Content-Type: application/x-www-form-urlencoded" \
    "http://www.example.com/auth/realms/quickstart/protocol/openid-connect/certs" \
    | jq .
```

3. Copy the result of the "x5c" response from the previous command and paste it into the **certificate.crt** file. Include **—-BEGIN CERTIFICATE—-** at the beginning and **—-END CERTIFICATE—-** at the end.

4. Get the certificate thumbprint:

**Syntax**

```
openssl x509 -in CERT_FILE -fingerprint -noout
```

**Example**

```
[user@client ~]$ openssl x509 -in certificate.crt -fingerprint -noout
SHA1 Fingerprint=F7:D7:B3:51:5D:D0:D3:19:DD:21:9A:43:A9:EA:72:7A:D6:06:52:87
```

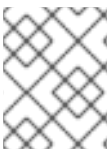
5. Remove all the colons from the SHA1 fingerprint and use this as the input for creating the IDP entity in the IAM request.

**Additional Resources**

- See the *Obtaining the Root CA Thumbprint for an OpenID Connect Identity Provider* article on Amazon's website.
- See the *Secure Token Service application programming interfaces* section in the *Red Hat Ceph Storage Developer Guide* for more details on the STS APIs.
- See the *Examples using the Secure Token Service APIs* section of the *Red Hat Ceph Storage Developer Guide* for more details.

### 2.3.9.5. Configuring and using STS Lite with Keystone (Technology Preview)

The Amazon Secure Token Service (STS) and S3 APIs co-exist in the same namespace. The STS options can be configured in conjunction with the Keystone options.



**NOTE**

Both S3 and STS APIs can be accessed using the same endpoint in Ceph Object Gateway.

**Prerequisites**

- Red Hat Ceph Storage 3.2 or higher.

- A running Ceph Object Gateway.
- Installation of the Boto Python module, version 3 or higher.

**Procedure**

1. Open and edit the **group_vars/rgws.yml** file with the following options:

```
rgw_sts_key = STS_KEY
rgw_s3_auth_use_sts = true
```

   Replace:

   - *STS_KEY* with the key used to encrypted the session token.

2. Rerun the appropriate Ceph Ansible playbook:

   a. **Bare-metal** deployments:

```
[user@admin ceph-ansible]$ ansible-playbook site.yml --limit rgws
```

   b. **Container** deployments:

```
[user@admin ceph-ansible]$ ansible-playbook site-docker.yml --limit rgws
```

3. Generate the EC2 credentials:

   **Example**

```
[user@osp ~]$ openstack ec2 credentials create

+------------+--------------------------------------------------------+
| Field      | Value                                                  |
+------------+--------------------------------------------------------+
| access     | b924dfc87d454d15896691182fdeb0ef                       |
| links      | {u'self': u'http://192.168.0.15/identity/v3/users/     |
|            | 40a7140e424f493d8165abc652dc731c/credentials/          |
|            | OS-EC2/b924dfc87d454d15896691182fdeb0ef'}              |
| project_id | c703801dccaf4a0aaa39bec8c481e25a                       |
| secret     | 6a2142613c504c42a94ba2b82147dc28                       |
| trust_id   | None                                                   |
| user_id    | 40a7140e424f493d8165abc652dc731c                       |
+------------+--------------------------------------------------------+
```

4. Use the generated credentials to get back a set of temporary security credentials using *GetSessionToken* API.

   **Example**

```
import boto3

access_key = b924dfc87d454d15896691182fdeb0ef
secret_key = 6a2142613c504c42a94ba2b82147dc28
```

```
client = boto3.client('sts',
aws_access_key_id=access_key,
aws_secret_access_key=secret_key,
endpoint_url=https://www.example.com/rgw,
region_name='',
)

response = client.get_session_token(
    DurationSeconds=43200
)
```

5. Obtaining the temporary credentials can be used for making S3 calls:

**Example**

```
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=https://www.example.com/s3,
  region_name='')

bucket = s3client.create_bucket(Bucket='my-new-shiny-bucket')
response = s3client.list_buckets()
for bucket in response["Buckets"]:
  print "{name}\t{created}".format(
    name = bucket['Name'],
    created = bucket['CreationDate'],
  )
```

6. Create a new *S3Access* role and configure a policy.

   a. Assign a user with administrative CAPS:

   **Syntax**

   ```
   radosgw-admin caps add --uid="USER" --caps="roles=*"
   ```

   **Example**

   ```
   [user@client]$ radosgw-admin caps add --uid="gwadmin" --caps="roles=*"
   ```

   b. Create the *S3Access* role:

   **Syntax**

   ```
   radosgw-admin role create --role-name=ROLE_NAME --path=PATH --assume-role-
   policy-doc=TRUST_POLICY_DOC
   ```

   **Example**

   ```
   [user@client]$ radosgw-admin role create --role-name=S3Access --
   path=/application_abc/component_xyz/ --assume-role-policy-doc=\{\"Version\":\"2012-10-
   ```

```
17\",\"Statement\":\[\{\"Effect\":\"Allow\",\"Principal\":\{\"AWS\":\
[\"arn:aws:iam:::user/TESTER\"\]\},\"Action\":\[\"sts:AssumeRole\"\]\}\]\}
```

c. Attach a permission policy to the *S3Access* role:

**Syntax**

```
radosgw-admin role-policy put --role-name=ROLE_NAME --policy-
name=POLICY_NAME --policy-doc=PERMISSION_POLICY_DOC
```

**Example**

```
[user@client]$ radosgw-admin role-policy put --role-name=S3Access --policy-
name=Policy --policy-doc=\{\"Version\":\"2012-10-17\",\"Statement\":\[\
{\"Effect\":\"Allow\",\"Action\":\[\"s3:*\"\],\"Resource\":\"arn:aws:s3:::example_bucket\"\}\]\}
```

d. Now another user can assume the role of the **gwadmin** user. For example, the **gwuser** user can assume the permissions of the **gwadmin** user.

e. Make a note of the assuming user's **access_key** and **secret_key** values.

**Example**

```
[user@client]$ radosgw-admin user info --uid=gwuser | grep -A1 access_key
```

7. Use the *AssumeRole* API call, providing the **access_key** and **secret_key** values from the assuming user:

**Example**

```
import boto3

access_key = 11BS02LGFB6AL6H1ADMW
secret_key = vzCEkuryfn060dfee4fgQPqFrncKEIkh3ZcdOANY

client = boto3.client('sts',
aws_access_key_id=access_key,
aws_secret_access_key=secret_key,
endpoint_url=https://www.example.com/rgw,
region_name='',
)

response = client.assume_role(
RoleArn='arn:aws:iam:::role/application_abc/component_xyz/S3Access',
RoleSessionName='Bob',
DurationSeconds=3600
)
```

**IMPORTANT**

The *AssumeRole* API requires the *S3Access* role.

**Additional Resources**

- See the *Test S3 Access* section in the *Red Hat Ceph Storage Object Gateway Guide* for more information on installing the Boto Python module.
- See the *Create a User* section in the *Red Hat Ceph Storage Object Gateway Guide* for more information.

#### 2.3.9.6. Working around the limitations of using STS Lite with Keystone (Technology Preview)

A limitation with Keystone is that it does not supports STS requests. Another limitation is the payload hash is not included with the request. To work around these two limitations the Boto authentication code must be modified.

**Prerequisites**

- A running Red Hat Ceph Storage cluster, version 3.2 or higher.
- A running Ceph Object Gateway.
- Installation of Boto Python module, version 3 or higher.

**Procedure**

1. Open and edit Boto's **auth.py** file.
   a. Add the following four lines to the code block:

```
class SigV4Auth(BaseSigner):
  """
  Sign a request with Signature V4.
  """
  REQUIRES_REGION = True

  def __init__(self, credentials, service_name, region_name):
     self.credentials = credentials
     # We initialize these value here so the unit tests can have
     # valid values.  But these will get overriden in ``add_auth``
     # later for real requests.
     self._region_name = region_name
     if service_name == 'sts': (1)
        self._service_name = 's3' (2)
     else: (3)
        self._service_name = service_name (4)
```

   b. Add the following two lines to the code block:

```
def _modify_request_before_signing(self, request):
       if 'Authorization' in request.headers:
          del request.headers['Authorization']
       self._set_necessary_date_headers(request)
       if self.credentials.token:
          if 'X-Amz-Security-Token' in request.headers:
             del request.headers['X-Amz-Security-Token']
          request.headers['X-Amz-Security-Token'] = self.credentials.token
```

```
if not request.context.get('payload_signing_enabled', True):
    if 'X-Amz-Content-SHA256' in request.headers:
        del request.headers['X-Amz-Content-SHA256']
    request.headers['X-Amz-Content-SHA256'] = UNSIGNED_PAYLOAD (1)
else: (2)
    request.headers['X-Amz-Content-SHA256'] = self.payload(request)
```

**Additional Resources**

- See the *Test S3 Access* section in the Red Hat Ceph Storage *Object Gateway Guide* for more information on installing the Boto Python module.

## 2.4. S3 BUCKET OPERATIONS

As a developer, you can perform bucket operations with the Amazon S3 application programing interface (API) through the Ceph Object Gateway.

The following table list the Amazon S3 functional operations for buckets, along with the function's support status.

**Table 2.2. Bucket operations**

| Feature | Status | Notes |
|---|---|---|
| List Buckets | Supported | |
| Create a Bucket | Supported | Different set of canned ACLs. |
| Bucket Lifecycle | Partially Supported | **Expiration**, **NoncurrentVersionExpiration** and **AbortIncompleteMultipartUpload** supported. |
| 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 | |

| Feature | Status | Notes |
|---|---|---|
| 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 | |
| Multi-tenant Bucket Operations | Supported | |

### 2.4.1. Prerequisites

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

### 2.4.2. 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** and **ObjectRemoved**. 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** and **s3:objectRemove** 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:*']
    }]})
```

IMPORTANT

Red Hat supports **ObjectCreate** events, such as, **put**, **post**, **multipartUpload**, and **copy**. Red Hat also supports **ObjectRemove** events, such as, **object_delete** and **s3_multi_object_delete**.

**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),the topic must be created before.

**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.

id="s3-get-bucket-notifications_dev"]

### 2.4.3. S3 get bucket notifications

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

#### Syntax

```
Get /BUCKET?notification=NOTIFICATION_ID HTTP/1.1
Host: cname.domain.com
Date: date
Authorization: AWS ACCESS_KEY:_HASH_OF_HEADER_AND_SECRET_
```

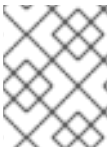
#### Example

```
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 that the topic must be created before.

**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.

### 2.4.4. 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.

## 2.4.5. Accessing bucket host names

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

### 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**

Red Hat prefers the first method, because the second method requires expensive domain certification and DNS wild cards.

### 2.4.6. 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_
```

**Table 2.3. Response Entities**

| Name | Type | Description |
|---|---|---|
| **Buckets** | Container | Container for list of buckets. |
| **Bucket** | Container | Container for bucket information. |
| **Name** | String | Bucket name. |
| **CreationDate** | Date | UTC time when the bucket was created. |
| **ListAllMyBucketsResult** | Container | A container for the result. |
| **Owner** | Container | A container for the bucket owner's **ID** and **DisplayName**. |
| **ID** | String | The bucket owner's ID. |
| **DisplayName** | String | The bucket owner's display name. |

### 2.4.7. 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
```

Table 2.4. Parameters

| Name | Type | Description |
|---|---|---|
| **prefix** | String | Only returns objects that contain the specified prefix. |
| **delimiter** | String | The delimiter between the prefix and the rest of the object name. |
| **marker** | String | A beginning index for the list of objects returned. |
| **max-keys** | Integer | The maximum number of keys to return. Default is 1000. |

Table 2.5. HTTP Response

| HTTP Status | Status Code | Description |
|---|---|---|
| **200** | OK | Buckets retrieved |

**GET** /***BUCKET*** returns a container for buckets with the following fields:

Table 2.6. Bucket Response Entities

| Name | Type | Description |
|---|---|---|
| **ListBucketResult** | Entity | The container for the list of objects. |
| **Name** | String | The name of the bucket whose contents will be returned. |
| **Prefix** | String | A prefix for the object keys. |
| **Marker** | String | A beginning index for the list of objects returned. |
| **MaxKeys** | Integer | The maximum number of keys returned. |
| **Delimiter** | String | If set, objects with the same prefix will appear in the **CommonPrefixes** list. |
| **IsTruncated** | Boolean | If **true**, only a subset of the bucket's contents were returned. |
| **CommonPrefixes** | Container | If multiple objects contain the same prefix, they will appear in this list. |

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

Table 2.7. Object Response Entities

| Name | Type | Description |
|---|---|---|
| **Contents** | Object | A container for the object. |
| **Key** | String | The object's key. |
| **LastModified** | Date | The object's last-modified date/time. |
| **ETag** | String | An MD-5 hash of the object. (entity tag) |
| **Size** | Integer | The object's size. |
| **StorageClass** | String | Should always return **STANDARD**. |

### 2.4.8. S3 create a new bucket

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

#### Constraints

In general, bucket names should follow domain name constraints.

- Bucket names must be unique.
- Bucket names must begin and end with a lowercase letter.
- Bucket names can contain a dash (-).

#### 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_
```

**Table 2.8. Parameters**

| Name | Description | Valid Values | Required |
|---|---|---|---|
| **x-amz-acl** | Canned ACLs. | **private**, **public-read**,**public-read-write**, **authenticated-read** | No |

#### HTTP Response

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

| HTTP Status | Status Code | Description |
|---|---|---|
| **409** | BucketAlreadyExists | Bucket already exists under different user's ownership. |

### 2.4.9. 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_
```

**Table 2.9. HTTP Response**

| HTTP Status | Status Code | Description |
|---|---|---|
| **204** | No Content | Bucket removed. |

### 2.4.10. 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 defines the lifespan of objects within a bucket. It takes the number of days the object should live or an expiration date, at which point Ceph Object Gateway will delete the object. If the bucket doesn't enable versioning, Ceph Object Gateway will delete the object permanently. If the bucket enables versioning, Ceph Object Gateway will create a delete marker for the current version, and then delete the current version.
- **NoncurrentVersionExpiration**: This defines the lifespan of non-current object versions within a bucket. To use this feature, the bucket must enable versioning. It takes the number of days a non-current object should live, at which point Ceph Object Gateway will delete the non-current object.
- **AbortIncompleteMultipartUpload**: This defines the number of days an incomplete multipart upload should live before it is aborted.

The lifecycle configuration contains one or more rules using 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 several ways:

- Key prefixes
- Object tags
- Both key prefix 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/>** would apply 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 using the **<Key>** and **<Value>** elements:

```
<LifecycleConfiguration>
    <Rule>
        <Status>Enabled</Status>
        <Filter>
            <Tag>
                <Key>key</Key>
                <Value>value</Value>
            </Tag>
```

```
        </Filter>
    </Rule>
</LifecycleConfiguration>
```

### Both prefix 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>
```

#### Additional Resources

- See the *Red Hat Ceph Storage Developer Guide* for details on getting a bucket lifecycle.
- See the *Red Hat Ceph Storage Developer Guide* for details on creating a bucket lifecycle.
- See the *Red Hat Ceph Storage Developer Guide* for details to delete a bucket lifecycle.

## 2.4.11. 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

See the Common Request Headers for more information.

### Response

The response contains the bucket lifecycle and its elements.

### 2.4.12. 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>
```

**Table 2.10. Request Headers**

| Name | Description | Valid Values | Required |
|---|---|---|---|
| **content-md5** | A base64 encoded MD-5 hash of the message. | A string. No defaults or constraints. | No |

#### Additional Resources

- See the *Red Hat Ceph Storage Developer Guide* for details on common Amazon S3 request headers.
- See the *Red Hat Ceph Storage Developer Guide* for details on Amazon S3 bucket lifecycles.

### 2.4.13. 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.

**Additional Resources**

- See *Appendix A* for Amazon S3 common request headers.
- See *Appendix B* for Amazon S3 common response status codes.

### 2.4.14. 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 bucket resource as shown below.

**Syntax**

```
GET /<bucket>?location HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:_HASH_OF_HEADER_AND_SECRET_
```

**Table 2.11. Response Entities**

| Name | Type | Description |
|---|---|---|
| **LocationConstraint** | String | The zone group where bucket resides, empty string for default zone group |

### 2.4.15. 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 bucket resource as shown below.

**Syntax**

```
GET /<bucket>?versioning HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:_HASH_OF_HEADER_AND_SECRET_
```

### 2.4.16. S3 put the 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
```

**Table 2.12. Bucket Request Entities**

| Name | Type | Description |
|---|---|---|
| **VersioningConfiguration** | container | A container for the request. |
| **Status** | String | Sets the versioning state of the bucket. Valid Values: Suspended/Enabled |

## 2.4.17. 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 as shown below.

### Syntax

```
GET /<bucket>?acl HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:_HASH_OF_HEADER_AND_SECRET_
```

**Table 2.13. Response Entities**

| Name | Type | Description |
|---|---|---|
| **AccessControlPolicy** | Container | A container for the response. |
| **AccessControlList** | Container | A container for the ACL information. |
| **Owner** | Container | A container for the bucket owner's **ID** and **DisplayName**. |
| **ID** | String | The bucket owner's ID. |
| **DisplayName** | String | The bucket owner's display name. |
| **Grant** | Container | A container for **Grantee** and **Permission**. |
| **Grantee** | Container | A container for the **DisplayName** and **ID** of the user receiving a grant of permission. |

| Name | Type | Description |
|---|---|---|
| **Permission** | String | The permission given to the **Grantee** bucket. |

### 2.4.18. 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 as shown below.

#### Syntax

```
PUT /<bucket>?acl HTTP/1.1
```

**Table 2.14. Request Entities**

| Name | Type | Description |
|---|---|---|
| **AccessControlPolicy** | Container | A container for the request. |
| **AccessControlList** | Container | A container for the ACL information. |
| **Owner** | Container | A container for the bucket owner's **ID** and **DisplayName**. |
| **ID** | String | The bucket owner's ID. |
| **DisplayName** | String | The bucket owner's display name. |
| **Grant** | Container | A container for **Grantee** and **Permission**. |
| **Grantee** | Container | A container for the **DisplayName** and **ID** of the user receiving a grant of permission. |
| **Permission** | String | The permission given to the **Grantee** bucket. |

### 2.4.19. 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 as shown below.

#### Syntax

```
GET /<bucket>?cors HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:_HASH_OF_HEADER_AND_SECRET_
```

### 2.4.20. 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 as shown below.

#### Syntax

```
PUT /<bucket>?cors HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:_HASH_OF_HEADER_AND_SECRET_
```

### 2.4.21. 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 as shown below.

#### Syntax

```
DELETE /<bucket>?cors HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:_HASH_OF_HEADER_AND_SECRET_
```

### 2.4.22. S3 list bucket object versions

Returns a list of metadata about all the version of objects within a bucket. Requires READ access to the bucket.

Add the **versions** subresource to the bucket request as shown below.

#### Syntax

```
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.

**Table 2.15. Parameters**

| Name | Type | Description |
|---|---|---|
| **prefix** | String | Returns in-progress uploads whose keys contains the specified prefix. |
| **delimiter** | String | The delimiter between the prefix and the rest of the object name. |
| **key-marker** | String | The beginning marker for the list of uploads. |
| **max-keys** | Integer | The maximum number of in-progress uploads. The default is 1000. |
| **version-id-marker** | String | Specifies the object version to begin the list. |

Table 2.16. Response Entities

| Name | Type | Description |
|---|---|---|
| **KeyMarker** | String | The key marker specified by the **key-marker** request parameter (if any). |
| **NextKeyMarker** | String | The key marker to use in a subsequent request if **IsTruncated** is **true**. |
| **NextUploadIdMarker** | String | The upload ID marker to use in a subsequent request if **IsTruncated** is **true**. |
| **IsTruncated** | Boolean | If **true**, only a subset of the bucket's upload contents were returned. |
| **Size** | Integer | The size of the uploaded part. |
| **DisplayName** | String | The owners's display name. |
| **ID** | String | The owners's ID. |
| **Owner** | Container | A container for the **ID** and **DisplayName** of the user who owns the object. |
| **StorageClass** | String | The method used to store the resulting object. **STANDARD** or **REDUCED_REDUNDANCY** |
| **Version** | Container | Container for the version information. |

| Name | Type | Description |
|---|---|---|
| **versionId** | String | Version ID of an object. |
| **versionIdMarker** | String | The last version of the key in a truncated response. |

### 2.4.23. 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_
```

### 2.4.24. 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.

**Table 2.17. Parameters**

| Name | Type | Description |
|---|---|---|
| **prefix** | String | Returns in-progress uploads whose keys contains the specified prefix. |
| **delimiter** | String | The delimiter between the prefix and the rest of the object name. |
| **key-marker** | String | The beginning marker for the list of uploads. |
| **max-keys** | Integer | The maximum number of in-progress uploads. The default is 1000. |
| **max-uploads** | Integer | The maximum number of multipart uploads. The range from 1-1000. The default is 1000. |

| Name | Type | Description |
|---|---|---|
| **version-id-marker** | String | Ignored if **key-marker** isn't specified. Specifies the **ID** of first upload to list in lexicographical order at or following the **ID**. |

Table 2.18. Response Entities

| Name | Type | Description |
|---|---|---|
| **ListMultipartUploadsResult** | Container | A container for the results. |
| **ListMultipartUploadsResult.Prefix** | String | The prefix specified by the **prefix** request parameter (if any). |
| **Bucket** | String | The bucket that will receive the bucket contents. |
| **KeyMarker** | String | The key marker specified by the **key-marker** request parameter (if any). |
| **UploadIdMarker** | String | The marker specified by the **upload-id-marker** request parameter (if any). |
| **NextKeyMarker** | String | The key marker to use in a subsequent request if **IsTruncated** is **true**. |
| **NextUploadIdMarker** | String | The upload ID marker to use in a subsequent request if **IsTruncated** is **true**. |
| **MaxUploads** | Integer | The max uploads specified by the **max-uploads** request parameter. |
| **Delimiter** | String | If set, objects with the same prefix will appear in the **CommonPrefixes** list. |
| **IsTruncated** | Boolean | If **true**, only a subset of the bucket's upload contents were returned. |
| **Upload** | Container | A container for **Key**, **UploadId**, **InitiatorOwner**, **StorageClass**, and **Initiated** elements. |
| **Key** | String | The key of the object once the multipart upload is complete. |
| **UploadId** | String | The **ID** that identifies the multipart upload. |

| Name | Type | Description |
|---|---|---|
| **Initiator** | Container | Contains the **ID** and **DisplayName** of the user who initiated the upload. |
| **DisplayName** | String | The initiator's display name. |
| **ID** | String | The initiator's ID. |
| **Owner** | Container | A container for the **ID** and **DisplayName** of the user who owns the uploaded object. |
| **StorageClass** | String | The method used to store the resulting object. **STANDARD** or **REDUCED_REDUNDANCY** |
| **Initiated** | Date | The date and time the user initiated the upload. |
| **CommonPrefixes** | Container | If multiple objects contain the same prefix, they will appear in this list. |
| **CommonPrefixes.Prefix** | String | The substring of the key after the prefix as defined by the **prefix** request parameter. |

### 2.4.25. S3 bucket policies

The Ceph Object Gateway supports a subset of the Amazon S3 policy language 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 may 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 only supports the following S3 actions:

- **s3:AbortMultipartUpload**
- **s3:CreateBucket**
- **s3:DeleteBucketPolicy**
- **s3:DeleteBucket**
- **s3:DeleteBucketWebsite**
- **s3:DeleteObject**
- **s3:DeleteObjectVersion**
- **s3:GetBucketAcl**
- **s3:GetBucketCORS**
- **s3:GetBucketLocation**
- **s3:GetBucketPolicy**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **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: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. Ceph Object Gateway administrators who want to use policies between Amazon Web Service (AWS) S3 and Ceph Object Gateway S3 will have to use the Amazon account ID as the tenant ID when creating users.

With AWS S3, all tenants share a single namespace. By contrast, Ceph Object Gateway gives every tenant its own namespace of buckets. At present, Ceph Object Gateway clients trying to access a bucket belonging 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 will 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 functionality to set bucket policies under the Swift API. However, bucket policies that have been set with the S3 API govern Swift as well as S3 operations.

Ceph Object Gateway matches Swift credentials against Principals specified in a policy.

### 2.4.26. 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 as shown below.

#### Syntax

```
GET /<bucket>?requestPayment HTTP/1.1
Host: cname.domain.com

Authorization: AWS ACCESS_KEY:_HASH_OF_HEADER_AND_SECRET_
```

### 2.4.27. 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 as shown below.

#### Syntax

```
PUT /<bucket>?requestPayment HTTP/1.1
Host: cname.domain.com
```

**Table 2.19. Request Entities**

| Name | Type | Description |
|---|---|---|
| **Payer** | Enum | Specifies who pays for the download and request fees. |
| **RequestPaymentConfiguration** | Container | A container for **Payer**. |

## 2.4.28. Multi-tenant bucket operations

When a client application accesses buckets, it always operates with credentials of a particular user. In storage-product} cluster, every user belongs to a tenant. 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.

#### Additional Resources

- See Multi Tenancy for additional details.

## 2.4.29. Additional Resources

- See the *Red Hat Ceph Storage Object Gateway Configuration and Administration Guide* for details on configuring a bucket website.

## 2.5. S3 OBJECT OPERATIONS

As a developer, you can perform object operations with the Amazon S3 application programing interface (API) through the Ceph Object Gateway.

The following table list the Amazon S3 functional operations for objects, along with the function's support status.

**Table 2.20. Object operations**

| Get Object | Supported | |
|---|---|---|
| Get Object Information | Supported | |
| Put Object | Supported | |
| Delete Object | Supported | |
| Delete Multiple Objects | Supported | |
| Get Object ACLs | Supported | |
| 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 | |

### 2.5.1. Prerequisites

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

### 2.5.2. S3 get an object from a bucket

Retrieves an object from a bucket:

#### Syntax

```
GET /BUCKET/OBJECT HTTP/1.1
```

Add the **versionId** subresource to retrieve a particular version of the object:

#### Syntax

```
GET /BUCKET/OBJECT?versionId=VERSIONID HTTP/1.1
```

**Table 2.21. Request Headers**

| Name | 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-unmodified-since** | Gets only if not 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 |

**Table 2.22. Response Headers**

| Name | Description |
|---|---|
| **Content-Range** | Data range, will only be returned if the range header field was specified in the request |
| **x-amz-version-id** | Returns the version ID or null. |

### 2.5.3. S3 get information on an object

Returns information about an object. This request will return the same header information as with the Get Object request, but will include the metadata only, not the object data payload.

Retrieves the current version of the object:

#### Syntax

```
HEAD /BUCKET/OBJECT HTTP/1.1
```

Add the **versionId** subresource to retrieve info for a particular version:

### Syntax

```
HEAD /BUCKET/OBJECT?versionId=VERSIONID HTTP/1.1
```

**Table 2.23. Request Headers**

| Name | 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-unmodified-since** | Gets only if not 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 |

**Table 2.24. Response Headers**

| Name | Description |
|---|---|
| **x-amz-version-id** | Returns the version ID or null. |

## 2.5.4. 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
```

**Table 2.25. Request Headers**

| Name | Description | Valid Values | Required |
|---|---|---|---|
| **content-md5** | A base64 encoded MD-5 hash of the message. | A string. No defaults or constraints. | No |

| Name | Description | Valid Values | Required |
|---|---|---|---|
| **content-type** | A standard MIME type. | Any MIME type. Default: **binary/octet-stream** | No |
| **x-amz-meta-<...>** | User metadata. Stored with the object. | A string up to 8kb. No defaults. | No |
| **x-amz-acl** | A canned ACL. | **private**, **public-read**, **public-read-write**, **authenticated-read** | No |

Table 2.26. Response Headers

| Name | Description |
|---|---|
| **x-amz-version-id** | Returns the version ID or null. |

### 2.5.5. S3 delete an object

Removes an object. Requires WRITE permission set on the containing bucket.

Deletes an object. If object versioning is on, it creates a marker.

#### Syntax

```
DELETE /BUCKET/OBJECT HTTP/1.1
```

To delete an object when versioning is on, you must specify the **versionId** subresource and the version of the object to delete.

```
DELETE /BUCKET/OBJECT?versionId=versionID HTTP/1.1
```

### 2.5.6. S3 delete multiple objects

This API call deletes multiple objects from a bucket.

#### Syntax

```
POST /<bucket>/<object>?delete HTTP/1.1
```

### 2.5.7. 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=_VERSIONID&acl HTTP/1.1
```

**Table 2.27. Response Headers**

| Name | Description |
|---|---|
| **x-amz-version-id** | Returns the version ID or null. |

**Table 2.28. Response Entities**

| Name | Type | Description |
|---|---|---|
| **AccessControlPolicy** | Container | A container for the response. |
| **AccessControlList** | Container | A container for the ACL information. |
| **Owner** | Container | A container for the object owner's **ID** and **DisplayName**. |
| **ID** | String | The object owner's ID. |
| **DisplayName** | String | The object owner's display name. |
| **Grant** | Container | A container for **Grantee** and **Permission**. |
| **Grantee** | Container | A container for the **DisplayName** and **ID** of the user receiving a grant of permission. |
| **Permission** | String | The permission given to the **Grantee** object. |

## 2.5.8. 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
```

**Table 2.29. Request Entities**

| Name | Type | Description |
|---|---|---|
| **AccessControlPolicy** | Container | A container for the response. |
| **AccessControlList** | Container | A container for the ACL information. |
| **Owner** | Container | A container for the object owner's **ID** and **DisplayName**. |
| **ID** | String | The object owner's ID. |
| **DisplayName** | String | The object owner's display name. |
| **Grant** | Container | A container for **Grantee** and **Permission**. |
| **Grantee** | Container | A container for the **DisplayName** and **ID** of the user receiving a grant of permission. |
| **Permission** | String | The permission given to the **Grantee** object. |

### 2.5.9. 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
```

**Table 2.30. Request Headers**

| Name | Description | Valid Values | Required |
|---|---|---|---|
| **x-amz-copy-source** | The source bucket name + object name. | <bucket>/<object> | Yes |
| **x-amz-acl** | A canned ACL. | **private**, **public-read**, **public-read-write**, **authenticated-read** | No |
| **x-amz-copy-if-modified-since** | Copies only if modified since the timestamp. | Timestamp | No |

| Name | Description | Valid Values | Required |
|---|---|---|---|
| **x-amz-copy-if-unmodified-since** | Copies only if unmodified since the timestamp. | Timestamp | No |
| **x-amz-copy-if-match** | Copies only if object ETag matches ETag. | Entity Tag | No |
| **x-amz-copy-if-none-match** | Copies only if object ETag doesn't match. | Entity Tag | No |

**Table 2.31. Response Entities**

| Name | Type | Description |
|---|---|---|
| **CopyObjectResult** | Container | A container for the response elements. |
| **LastModified** | Date | The last modified date of the source object. |
| **Etag** | String | The ETag of the new object. |

**Additional Resources**

- <additional resource 1>
- <additional resource 2>

## 2.5.10. 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
```

## 2.5.11. 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
```

## 2.5.12. 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
```

Table 2.32. Request Headers

| Name | Description | Valid Values | Required |
|---|---|---|---|
| **content-md5** | A base64 encoded MD-5 hash of the message. | A string. No defaults or constraints. | No |
| **content-type** | A standard MIME type. | Any MIME type. Default: **binary/octet-stream** | No |
| **x-amz-meta-<...>** | User metadata. Stored with the object. | A string up to 8kb. No defaults. | No |
| **x-amz-acl** | A canned ACL. | **private**, **public-read**, **public-read-write**, **authenticated-read** | No |

Table 2.33. Response Entities

| Name | Type | Description |
|---|---|---|
| **InitiatedMultipartUploadsResult** | Container | A container for the results. |
| **Bucket** | String | The bucket that will receive the object contents. |
| **Key** | String | The key specified by the **key** request parameter (if any). |
| **UploadId** | String | The ID specified by the **upload-id** request parameter identifying the multipart upload (if any). |

## 2.5.13. 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:

**Table 2.34. HTTP Response**

| HTTP Status | Status Code | Description |
|---|---|---|
| **404** | NoSuchUpload | Specified upload-id does not match any initiated upload on this object |

### 2.5.14. 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
```

**Table 2.35. Response Entities**

| Name | Type | Description |
|---|---|---|
| **InitiatedMultipartUploadsResult** | Container | A container for the results. |
| **Bucket** | String | The bucket that will receive the object contents. |
| **Key** | String | The key specified by the **key** request parameter (if any). |
| **UploadId** | String | The ID specified by the **upload-id** request parameter identifying the multipart upload (if any). |
| **Initiator** | Container | Contains the **ID** and **DisplayName** of the user who initiated the upload. |
| **ID** | String | The initiator's ID. |
| **DisplayName** | String | The initiator's display name. |
| **Owner** | Container | A container for the **ID** and **DisplayName** of the user who owns the uploaded object. |
| **StorageClass** | String | The method used to store the resulting object. **STANDARD** or **REDUCED_REDUNDANCY** |
| **PartNumberMarker** | String | The part marker to use in a subsequent request if **IsTruncated** is **true**. Precedes the list. |

| Name | Type | Description |
|---|---|---|
| **NextPartNumberMarker** | String | The next part marker to use in a subsequent request if **IsTruncated** is **true**. The end of the list. |
| **MaxParts** | Integer | The max parts allowed in the response as specified by the **max-parts** request parameter. |
| **IsTruncated** | Boolean | If **true**, only a subset of the object's upload contents were returned. |
| **Part** | Container | A container for **Key**, **Part**, **InitiatorOwner**, **StorageClass**, and **Initiated** elements. |
| **PartNumber** | Integer | The identification number of the part. |
| **ETag** | String | The part's entity tag. |
| **Size** | Integer | The size of the uploaded part. |

### 2.5.15. 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= HTTP/1.1
```

**Table 2.36. Request Entities**

| Name | Type | Description | Required |
|---|---|---|---|
| **CompleteMultipartUpload** | Container | A container consisting of one or more parts. | Yes |
| **Part** | Container | A container for the **PartNumber** and **ETag**. | Yes |
| **PartNumber** | Integer | The identifier of the part. | Yes |
| **ETag** | String | The part's entity tag. | Yes |

**Table 2.37. Response Entities**

| Name | Type | Description |
|---|---|---|
| **CompleteMultipartUploadResult** | Container | A container for the response. |
| **Location** | URI | The resource identifier (path) of the new object. |
| **Bucket** | String | The name of the bucket that contains the new object. |
| **Key** | String | The object's key. |
| **ETag** | String | The entity tag of the new object. |

### 2.5.16. 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_
```

**Table 2.38. Request Headers**

| Name | Description | Valid Values | Required |
|---|---|---|---|
| **x-amz-copy-source** | The source bucket name and object name. | *BUCKET/OBJECT* | Yes |
| **x-amz-copy-source-range** | The range of bytes to copy from the source object. | 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. | No |

**Table 2.39. Response Entities**

| Name | Type | Description |
|---|---|---|
| **CopyPartResult** | Container | A container for all response elements. |
| **ETag** | String | Returns the ETag of the new part. |
| **LastModified** | String | Returns the date the part was last modified. |

.Additional Resources

- For more information about this feature, see the *Amazon S3 site*.

### 2.5.17. 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
```

### 2.5.18. 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.

### 2.5.19. Additional Resources

- See the *Red Hat Ceph Storage Object Gateway Configuration and Administration Guide* for details on multi-tenancy.

## 2.6. ADDITIONAL RESOURCES

- See *Appendix A* for Amazon S3 common request headers.
- See *Appendix B* for Amazon S3 common response status codes.
- See *Appendix C* for unsupported header fields.

# CHAPTER 3. CEPH OBJECT GATEWAY AND THE SWIFT API

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

The following table describes the support status for current Swift functional features:

**Table 3.1. Features**

| Feature | Status | Remarks |
|---|---|---|
| Authentication | Supported | |
| Get Account Metadata | Supported | No custom metadata |
| Swift ACLs | Supported | Supports a subset of Swift ACLs |
| List Containers | Supported | |
| List Container's Objects | Supported | |
| Create Container | Supported | |
| Delete Container | Supported | |
| Get Container Metadata | Supported | |
| Add/Update Container Metadata | Supported | |
| Delete Container Metadata | Supported | |
| Get Object | Supported | |
| Create/Update an Object | Supported | |
| Create Large Object | Supported | |
| Delete Object | Supported | |
| Copy Object | Supported | |
| Get Object Metadata | Supported | |
| Add/Update Object Metadata | Supported | |
| Temp URL Operations | Supported | |

| Feature | Status | Remarks |
|---|---|---|
| **CORS** | Not Supported | |
| **Expiring Objects** | Supported | |
| **Object Versioning** | Not Supported | |
| **Static Website** | Not Supported | |

## 3.1. PREREQUISITES

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

## 3.2. SWIFT API LIMITATIONS

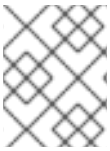
**IMPORTANT**

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 limited to 16,000 bytes.

## 3.3. 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.

**Procedure**

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@rgw]# 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@rgw]# 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"
}
```

## 3.4. 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.

### Additional Resources

- See the *Red Hat Ceph Storage Developer Guide* for Swift request headers.
- See the *Red Hat Ceph Storage Developer Guide* for Swift response headers.

## 3.5. SWIFT CONTAINER OPERATIONS

As a developer, you can perform container operations with the Swift application programing 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.

### 3.5.1. Prerequisites

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

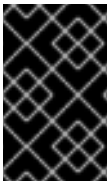
### 3.5.2. Swift container operations

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.

### 3.5.3. 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
```

**Table 3.2. Request Headers**

| Name | Description | Type | Required |
|---|---|---|---|
| **X-Container-Read** | The user IDs with read permissions for the container. | Comma-separated string values of user IDs. | No |

| Name | Description | Type | Required |
|---|---|---|---|
| **X-Container-Write** | The user IDs with write permissions for the container. | Comma-separated string values of user IDs. | No |

### 3.5.4. 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
```

**Table 3.3. Request Parameters**

| Name | Description | Type | Required | Valid Values |
|---|---|---|---|---|
| **limit** | Limits the number of results to the specified value. | Integer | No | N/A |
| **format** | Defines the format of the result. | String | No | **json** or **xml** |
| **marker** | Returns a list of results greater than the marker value. | String | No | N/A |

The response contains a list of containers, or returns with an HTTP 204 response code

**Table 3.4. Response Entities**

| Name | Description | Type |
|---|---|---|
| **account** | A list for account information. | Container |
| **container** | The list of containers. | Container |
| **name** | The name of a container. | String |

| Name | Description | Type |
|---|---|---|
| **bytes** | The size of the container. | Integer |

### 3.5.5. Swift list a container's objects

To list the objects within a container, make a **GET** request with the 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
```

Table 3.5. Parameters

| Name | Description | Type | Valid Values | Required |
|---|---|---|---|---|
| **format** | Defines the format of the result. | String | **json** or **xml** | No |
| **prefix** | Limits the result set to objects beginning with the specified prefix. | String | N/A | No |
| **marker** | Returns a list of results greater than the marker value. | String | N/A | No |
| **limit** | Limits the number of results to the specified value. | Integer | 0 - 10,000 | No |
| **delimiter** | The delimiter between the prefix and the rest of the object name. | String | N/A | No |
| **path** | The pseudo-hierarchical path of the objects. | String | N/A | No |

Table 3.6. Response Entities

| Name | Description | Type |
|---|---|---|
| **container** | The container. | Container |
| **object** | An object within the container. | Container |

| Name | Description | Type |
|---|---|---|
| **name** | The name of an object within the container. | String |
| **hash** | A hash code of the object's contents. | String |
| **last_modified** | The last time the object's contents were modified. | Date |
| **content_type** | The type of content within the object. | String |

### 3.5.6. 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, execute **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
```

**Table 3.7. Headers**

| Name | Description | Type | Required |
|---|---|---|---|
| **X-Container-Read** | The user IDs with read permissions for the container. | Comma-separated string values of user IDs. | No |

| Name | Description | Type | Required |
|---|---|---|---|
| **X-Container-Write** | The user IDs with write permissions for the container. | Comma-separated string values of user IDs. | No |
| **X-Container-Meta-*KEY*** | A user-defined meta data key that takes an arbitrary string value. | String | No |
| **X-Storage-Policy** | The key that identifies the storage policy under **placement_pools** for the Ceph Object Gateway. Execute **radosgw-admin zone get** for available keys. | String | 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.

**Table 3.8. HTTP Response**

| Name | Description | Status Code |
|---|---|---|
| **409** | The container already exists under a different user's ownership. | **BucketAlreadyExists** |

### 3.5.7. 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, execute 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
```

**Table 3.9. HTTP Response**

| Name | Description | Status Code |
|---|---|---|
| **204** | The container was removed. | **NoContent** |

### 3.5.8. 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
```

**Table 3.10. Request Headers**

| Name | Description | Type | Required |
|---|---|---|---|
| **X-Container-Meta-*KEY*** | A user-defined meta data key that takes an arbitrary string value. | String | No |

## 3.6. SWIFT OBJECT OPERATIONS

As a developer, you can perform object operations with the Swift application programing 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.

### 3.6.1. Prerequisites

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

### 3.6.2. 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.

### 3.6.3. 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
```

**Table 3.11. Request Headers**

| Name | Description | Type | Required |
|---|---|---|---|
| **range** | To retrieve a subset of an object's contents, you can specify a byte range. | Date | No |
| **If-Modified-Since** | Only copies if modified since the date/time of the source object's **last_modified** attribute. | Date | No |
| **If-Unmodified-Since** | Only copies if not modified since the date/time of the source object's **last_modified** attribute. | Date | No |
| **Copy-If-Match** | Copies only if the ETag in the request matches the source object's ETag. | ETag. | No |
| **Copy-If-None-Match** | Copies only if the ETag in the request does not match the source object's ETag. | ETag. | No |

**Table 3.12. Response Headers**

| Name | Description |
|---|---|
| **Content-Range** | The range of the subset of object contents. Returned only if the range header field was specified in the request. |

## 3.6.4. 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
```

**Table 3.13. Request Headers**

| Name | Description | Type | Required | Valid Values |
|---|---|---|---|---|
| **ETag** | An MD5 hash of the object's contents. Recommended. | String | No | N/A |
| **Content-Type** | The type of content the object contains. | String | No | N/A |
| **Transfer-Encoding** | Indicates whether the object is part of a larger aggregate object. | String | No | **chunked** |

### 3.6.5. 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
```

### 3.6.6. 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
```

Table 3.14. Request Headers

| Name | Description | Type | Required |
|---|---|---|---|
| **X-Copy-From** | Used with a **PUT** request to define the source container/object path. | String | Yes, if using **PUT** |
| **Destination** | Used with a **COPY** request to define the destination container/object path. | String | Yes, if using **COPY** |
| **If-Modified-Since** | Only copies if modified since the date/time of the source object's **last_modified** attribute. | Date | No |
| **If-Unmodified-Since** | Only copies if not modified since the date/time of the source object's **last_modified** attribute. | Date | No |
| **Copy-If-Match** | Copies only if the ETag in the request matches the source object's ETag. | ETag. | No |
| **Copy-If-None-Match** | Copies only if the ETag in the request does not match the source object's ETag. | ETag. | No |

### 3.6.7. 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
```

### 3.6.8. 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
```

Table 3.15. Request Headers

| Name | Description | Type | Required |
|---|---|---|---|
| **X-Object-Meta-*KEY*** | A user-defined meta data key that takes an arbitrary string value. | String | No |

## 3.7. 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.

### 3.7.1. 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 format of seconds since the epoch, that is, Unix time
- The request path starting from "v1" onwards

The above items are normalized with newlines appended between them, and a HMAC is generated using the SHA-1 hashing algorithm against one of the Temp URL Keys posted earlier.

A sample python script to demonstrate the above is given below:

**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
```

### 3.7.2. 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
```

**Table 3.16. Request Headers**

| Name | Description | Type | Required |
|---|---|---|---|
| **X-Account-Meta-Temp-URL-Key** | A user-defined key that takes an arbitrary string value. | String | Yes |
| **X-Account-Meta-Temp-URL-Key-2** | A user-defined key that takes an arbitrary string value. | String | No |

## 3.8. SWIFT MULTI-TENANCY CONTAINER OPERATIONS

When a client application accesses containers, it always operates with credentials of a particular user. In 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")
```

## 3.9. ADDITIONAL RESOURCES

- See the *Red Hat Ceph Storage Object Gateway Configuration and Administration Guide* for details on multi-tenancy.
- See *Appendix D* for Swift request headers.
- See *Appendix E* for Swift response headers.

# APPENDIX A. S3 COMMON REQUEST HEADERS

The following table lists the valid common request headers and their descriptions.

**Table A.1. 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. |

# APPENDIX B. S3 COMMON RESPONSE STATUS CODES

The following table lists the valid common HTTP response status and its corresponding code.

**Table B.1. 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 |

| HTTP Status | Response Code |
|---|---|
| **404** | NoSuchKey |
| **404** | NoSuchBucket |
| **404** | NoSuchUpload |
| **405** | MethodNotAllowed |
| **408** | RequestTimeout |
| **409** | BucketAlreadyExists |
| **409** | BucketNotEmpty |
| **411** | MissingContentLength |
| **412** | PreconditionFailed |
| **416** | InvalidRange |
| **422** | UnprocessableEntity |
| **500** | InternalError |

# APPENDIX C. S3 UNSUPPORTED HEADER FIELDS

Table C.1. 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 |

# APPENDIX D. SWIFT REQUEST HEADERS

Table D.1. 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 |

# APPENDIX E. 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.

**Table E.1. Response Headers**

| Name | Description | Type |
|---|---|---|
| **X-Storage-Token** | The authorization token for the **X-Auth-User** specified in the request. | String |
| **X-Storage-Url** | The URL and ***API_VERSION/ACCOUNT*** path for the user. | String |

# APPENDIX F. 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.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(
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=''
)

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 = 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()
```

### Additional Resources

- See the *Test S3 Access* section of the *Red Hat Ceph Storage Object Gateway Configuration and Administration Guide* for more details on using Python's **boto** module.