



Red Hat Ceph Storage 5

開発者ガイド

Red Hat Ceph Storage の各種アプリケーションプログラミングインターフェイスの
使用

Red Hat Ceph Storage 5 開発者ガイド

Red Hat Ceph Storage の各種アプリケーションプログラミングインターフェイスの使用

法律上の通知

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本ガイドでは、AMD64 および Intel 64 のアーキテクチャーで実行している Red Hat Ceph Storage のさまざまなアプリケーションプログラミングインターフェイスを使用する方法を説明します。Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージをご覧ください。

目次

第1章 CEPH RESTFUL API	4
1.1. 前提条件	4
1.2. CEPH API のバージョン	4
1.3. CEPH API の認証および認可	5
1.4. CEPH API モジュールの有効化と保護	5
1.5. 質問および回答	7
1.6. 関連情報	32
第2章 CEPH OBJECT GATEWAY 管理 API	33
2.1. 前提条件	35
2.2. 管理操作	35
2.3. 管理認証要求	35
2.4. 管理ユーザーの作成	43
2.5. ユーザー情報の取得	45
2.6. ユーザーの作成	47
2.7. ユーザーの変更	53
2.8. ユーザーの削除	58
2.9. サブユーザーの作成	59
2.10. サブユーザーの変更	62
2.11. サブユーザーの削除	65
2.12. ユーザーへの機能の追加	66
2.13. ユーザーからの機能の削除	68
2.14. キーの作成	70
2.15. 鍵の削除	73
2.16. バケット通知	74
2.17. バケット情報の取得	83
2.18. バケットインデックスを確認します。	86
2.19. バケットの削除	88
2.20. バケットのリンク	89
2.21. バケットのリンクを解除します。	92
2.22. バケットまたはオブジェクトポリシーを取得する	93
2.23. オブジェクトの削除	94
2.24. QUOTAS	95
2.25. ユーザークォータの取得	96
2.26. ユーザークォータの設定	96
2.27. バケットクォータの取得	96
2.28. バケットクォータの設定	99
2.29. 使用方法情報の取得	99
2.30. 使用方法に関する情報を削除	103
2.31. 標準エラーレスポンス	104
第3章 CEPH OBJECT GATEWAY および S3 API	106
3.1. 前提条件	106
3.2. S3 の制限	106
3.3. S3 API を使用した CEPH OBJECT GATEWAY へのアクセス	106
3.4. S3 バケット操作	138
3.5. S3 オブジェクト操作	179
3.6. S3 選択操作 (テクノロジープレビュー)	206
3.7. 関連情報	217
第4章 CEPH OBJECT GATEWAY および SWIFT API	218
4.1. 前提条件	219

4.2. SWIFT API の制限	219
4.3. SWIFT ユーザーの作成	219
4.4. ユーザーの SWIFT 認証	222
4.5. SWIFT コンテナ操作	222
4.6. SWIFT オブジェクト操作	230
4.7. SWIFT の一時 URL 操作	236
4.8. SWIFT マルチテナンシーコンテナの操作	237
4.9. 関連情報	238
付録A CEPH RESTFUL API 仕様	239
A.1. 前提条件	240
A.2. CEPH の概要	240
A.3. 認証	240
A.4. CEPH ファイルシステム	242
A.5. ストレージクラスターの設定	249
A.6. CRUSH ルール	253
A.7. イレイジャーコードプロファイル	255
A.8. 機能トグル	257
A.9. GRAFANA	257
A.10. ストレージクラスターの正常性	259
A.11. ホスト	260
A.12. ISCSI	265
A.13. ログ	269
A.14. CEPH MANAGER モジュール	270
A.15. CEPH MONITOR	273
A.16. CEPH OSD	274
A.17. CEPH OBJECT GATEWAY	284
A.18. ロールを操作する REST API	297
A.19. NFS GANESHA	301
A.20. CEPH ORCHESTRATOR	306
A.21. POOLS	307
A.22. PROMETHEUS	310
A.23. RADOS ブロックデバイス	313
A.24. パフォーマンスカウンター	332
A.25. ロール	336
A.26. サービス	339
A.27. 設定	342
A.28. CEPH タスク	345
A.29. テレメトリー	346
A.30. CEPH ユーザー	347
付録B S3 の一般的なリクエストヘッダー	352
付録C S3 の一般的なレスポンスステータスコード	353
付録D S3 サポートされないヘッダーフィールド	355
付録E SWIFT リクエストヘッダー	356
付録F SWIFT レスポンスヘッダー	357
付録G SECURE TOKEN SERVICE API の使用例	358

第1章 CEPH RESTFUL API

ストレージ管理者は、Ceph RESTful API または単に Red Hat Ceph Storage Dashboard が提供する Ceph API を使用して、Red Hat Ceph Storage クラスターと対話することができます。Ceph Monitors および OSD に関する情報と、それぞれの設定オプションを表示できます。Ceph プールを作成または編集することもできます。

Ceph API は次の標準を使用します。

- HTTP 1.1
- JSON
- MIME および HTTP コンテンツ Negotiation
- JWT

これらの標準は OpenAPI 3.0 に準拠しており、API 構文、セマンティクス、コンテンツエンコーディング、バージョン管理、認証、および承認を規制しています。

1.1. 前提条件

- 正常かつ実行中の Red Hat Ceph Storage クラスター
- Ceph Manager を実行するノードへのアクセス。

1.2. CEPH API のバージョン

Ceph RESTful API の主な目的は、安定したインターフェイスを提供することです。安定したインターフェイスを実現するには、Ceph API は以下の原則で構築されます。

- 暗黙的なデフォルトを回避するために、すべてのエンドポイントに対する明示的なデフォルトバージョン。
- 粒度の細かい変更をエンドポイントごとに制御します。
 - 特定のエンドポイントからの予想されるバージョンは HTTP ヘッダーに記載されます。

構文

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

例

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

現在の Ceph API サーバーがその特定のバージョンを対応できない場合は、**415 - Unsupported Media Type** の応答が返されます。

- セマンティックバージョンニングの使用。
 - 主な変更点は後方互換性がありません。変更すると、リクエストに無関係な変更や、特定のエンドポイントの応答形式に変更が加えられる可能性があります。

- マイナーな変更は、後方互換性および転送の後方的です。変更は、特定のエンドポイントの要求形式または応答形式への追加変更で設定されます。

1.3. CEPH API の認証および認可

Ceph RESTful API へのアクセスは、2つのチェックポイントを通過します。まず、有効なユーザーと既存ユーザーの代理で要求が行われていることを認証します。2つ目は、認証前のユーザーの作成者は、ターゲットのエンドユーザーでの作成、読み取り、更新、削除などの特定のアクションを実行できます。

ユーザーが Ceph API の使用を開始する前に、有効な JSON Web Token (JWT) が必要です。`/api/auth` エンドポイントでは、このトークンを取得できます。

例

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

このトークンは、**Authorization** HTTP ヘッダー内に配置して、すべての API 要求と共に使用する必要があります。

構文

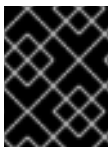
```
curl -H "Authorization: Bearer TOKEN" ...
```

関連情報

- 詳細は、Red Hat Ceph Storage 管理ガイドの [Ceph ユーザー管理](#) の章を参照してください。

1.4. CEPH API モジュールの有効化と保護

Red Hat Ceph Storage Dashboard モジュールは、SSL セキュアな接続で、ストレージクラスターに RESTful API アクセスを提供します。



重要

SSL を無効にすると、ユーザー名およびパスワードが暗号化されずに Red Hat Ceph ストレージダッシュボードに送信されます。

前提条件

- Ceph Monitor ノードへの root レベルのアクセス。
- 少なくとも1つの **ceph-mgr** デーモンがアクティブであることを確認します。
- ファイアウォールを使用する場合は、SSL の場合は TCP ポート **8443**、SSL なしの TCP ポート **8080** が、アクティブな **ceph-mgr** デーモンのあるノードでオープンになっていることを確認してください。

手順

1. Cephadm シェルにログインします。

例

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

2. RESTful プラグインを有効にします。

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

3. SSL 証明書を設定します。

- a. 組織の認証局 (CA) が証明書を提供する場合は、証明書ファイルを使用して設定されます。

構文

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

例

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

一意のノードベースの証明書を設定する場合は、**HOST_NAME** をコマンドに追加します。

例

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

- b. または、自己署名証明書を生成することもできます。ただし、自己署名証明書を使用しても、HTTPS プロトコルのセキュリティ上の利点を十分に享受することはできません。

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



警告

自己署名証明書についてエラーを表示する最新の Web ブラウザーのほとんどは、安全な接続を確立する前に確認する必要があります。

4. ユーザーを作成し、パスワードを設定し、ロールを設定します。

構文

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

例

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

この例では、**administrator** ロールを持つ **user1** という名前のユーザーを作成します。

5. RESTful プラグインの Web ページに接続します。Web ブラウザーを開き、以下の URL を入力します。

構文

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

例

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

自己署名証明書を使用した場合は、セキュリティー例外を確認します。

関連情報

- **ceph dashboard --help** コマンド
- **https://HOST_NAME:8443/doc** ページ。HOST_NAME は、実行中の **ceph-mgr** インスタンスを持つノードの IP アドレスまたは名前です。
- [Red Hat Enterprise Linux 8 セキュリティー強化ガイド](#) の内容を確認してください。

1.5. 質問および回答

1.5.1. 情報の取得

このセクションでは、Ceph API を使用して、ストレージクラスター、Ceph モニター、OSD、プール、およびホストに関する情報を表示する方法について説明します。

- [「すべてのクラスター設定オプションを表示する方法」](#)
- [「特定のクラスター設定オプションを表示する方法」](#)
- [「OSD のすべての設定オプションを表示する方法」](#)
- [「CRUSH ルールの表示方法」](#)
- [「Monitor に関する情報を表示する方法」](#)
- [「特定のモニターに関する情報を表示する方法」](#)
- [「OSD に関する情報を表示する方法」](#)
- [「特定の OSD に関する情報を表示する方法」](#)
- [「OSD でどのプロセスがスケジュールされるのかを指定する方法」](#)

- [「プールに関する情報の表示方法」](#)
- [「特定のプールに関する情報を表示する方法」](#)
- [「ホストに関する情報を表示する方法」](#)
- [「特定のホストに関する情報を表示する方法」](#)

1.5.1.1. すべてのクラスター設定オプションを表示する方法

本セクションでは、RESTful プラグインを使用してクラスター設定オプションおよびその値を表示する方法を説明します。

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **CEPH_MANAGER_PORT** は、TCP ポート番号に置き換えます。デフォルトの TCP ポート番号は 8443 です。

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

```
$ python
```

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

Web ブラウザー

Web ブラウザーで以下を入力します。

```
https://CEPH_MANAGER:8080/api/cluster_conf
```

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**

プロンプトが表示されたら、ユーザー名とパスワードを入力します。

関連情報

- Red Hat Ceph Storage 5 の [設定ガイド](#)

1.5.1.2. 特定のクラスター設定オプションを表示する方法

本セクションでは、特定のクラスターオプションとその値を表示する方法を説明します。

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **ARGUMENT** は、表示する設定オプションに置き換えます。

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **ARGUMENT** は、表示する設定オプションに置き換えます。
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

Web ブラウザー

Web ブラウザーで以下を入力します。

```
https://CEPH_MANAGER:8080/api/cluster_conf/ARGUMENT
```

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **ARGUMENT** は、表示する設定オプションに置き換えます。

プロンプトが表示されたら、ユーザー名とパスワードを入力します。

関連情報

- Red Hat Ceph Storage 5 の [設定ガイド](#)

1.5.1.3. OSD のすべての設定オプションを表示する方法

本セクションでは、OSD のすべての設定オプションおよびその値を表示する方法を説明します。

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

Web ブラウザー

Web ブラウザーで以下を入力します。

```
https://CEPH_MANAGER:8080/api/osd/flags
```

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**

プロンプトが表示されたら、ユーザー名とパスワードを入力します。

関連情報

- Red Hat Ceph Storage 5 の [設定ガイド](#)

1.5.1.4. CRUSH ルールの表示方法

このセクションでは、CRUSH ルールを表示する方法を説明します。

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

Web ブラウザー

Web ブラウザーで以下を入力します。

```
https://CEPH_MANAGER:8080/api/crush_rule
```

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**

プロンプトが表示されたら、ユーザー名とパスワードを入力します。

関連情報

- Red Hat Ceph Storage 5 の [管理ガイド](#) の [CRUSH ルール](#) セクション

1.5.1.5. Monitor に関する情報を表示する方法

本セクションでは、以下のような特定の Monitor に関する情報を表示する方法を説明します。

- IP アドレス
- 名前
- クォーラムのステータス

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

Web ブラウザー

Web ブラウザーで以下を入力します。

```
https://CEPH_MANAGER:8080/api/monitor
```

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**

プロンプトが表示されたら、ユーザー名とパスワードを入力します。

1.5.1.6. 特定のモニターに関する情報を表示する方法

本セクションでは、以下のような特定の Monitor に関する情報を表示する方法を説明します。

- IP アドレス
- 名前
- クォーラムのステータス

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **NAME** は、Monitor の短縮ホスト名に置き換えます。

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **NAME** は、Monitor の短縮ホスト名に置き換えます。

- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

Web ブラウザー

Web ブラウザーで以下を入力します。

```
https://CEPH_MANAGER:8080/api/monitor/NAME
```

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **NAME** は、Monitor の短縮ホスト名に置き換えます。

プロンプトが表示されたら、ユーザー名とパスワードを入力します。

1.5.1.7. OSD に関する情報を表示する方法

本セクションでは、以下のような OSD に関する情報を表示する方法を説明します。

- IP アドレス
- そのプール
- アフィニティー
- 重み

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

Web ブラウザー

Web ブラウザーで以下を入力します。

```
https://CEPH_MANAGER:8080/api/osd
```

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**

プロンプトが表示されたら、ユーザー名とパスワードを入力します。

1.5.1.8. 特定の OSD に関する情報を表示する方法

本セクションでは、以下のような特定の OSD に関する情報を表示する方法を説明します。

- IP アドレス
- そのプール
- アフィニティー
- 重み

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **osd** フィールドにリストされている OSD の **ID** を持つ ID

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **osd** フィールドにリストされている OSD の **ID** を持つ ID
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

Web ブラウザー

Web ブラウザーで以下を入力します。

```
https://CEPH_MANAGER:8080/api/osd/ID
```

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **osd** フィールドにリストされている OSD の **ID** を持つ ID

プロンプトが表示されたら、ユーザー名とパスワードを入力します。

1.5.1.9. OSD でどのプロセスがスケジュールされるのかを指定する方法

本セクションでは、RESTful プラグインを使用して、スクラビングやディープスクラビングなどのプロセスを OSD にスケジュールする方法を説明します。

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **osd** フィールドにリストされている OSD の **ID** を持つ ID

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **osd** フィールドにリストされている OSD の **ID** を持つ ID
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

Web ブラウザー

Web ブラウザーで以下を入力します。

```
https://CEPH_MANAGER:8080/api/osd/ID/command
```

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **osd** フィールドにリストされている OSD の **ID** を持つ ID

プロンプトが表示されたら、ユーザー名とパスワードを入力します。

1.5.1.10. プールに関する情報の表示方法

本セクションでは、以下のようなプールの情報を表示する方法を説明します。

- フラグ
- サイズ
- 配置グループの数

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

■

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

Web ブラウザー

Web ブラウザーで以下を入力します。

```
https://CEPH_MANAGER:8080/api/pool
```

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**

プロンプトが表示されたら、ユーザー名とパスワードを入力します。

1.5.1.11. 特定のプールに関する情報を表示する方法

本セクションでは、以下のような特定のプールに関する情報を表示する方法を説明します。

- フラグ
- サイズ
- 配置グループの数

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **pool** フィールドにリストされているプールの **ID** を持つ ID

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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



```
"PASSWORD"))
>> print result.json()
```

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **pool** フィールドにリストされているプールの **ID** を持つ ID
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

Web ブラウザー

Web ブラウザーで以下を入力します。

```
https://CEPH_MANAGER:8080/api/pool/ID
```

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **pool** フィールドにリストされているプールの **ID** を持つ ID

プロンプトが表示されたら、ユーザー名とパスワードを入力します。

1.5.1.12. ホストに関する情報を表示する方法

本セクションでは、以下のようなホストに関する情報を表示する方法を説明します。

- ホスト名
- Ceph デーモンとその ID
- Ceph バージョン

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **USER** は、ユーザー名に置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

Web ブラウザー

Web ブラウザーで以下を入力します。

```
https://CEPH_MANAGER:8080/api/host
```

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**

プロンプトが表示されたら、ユーザー名とパスワードを入力します。

1.5.1.13. 特定のホストに関する情報を表示する方法

本セクションでは、以下のような特定のホストに関する情報を表示する方法を説明します。

- ホスト名
- Ceph デーモンとその ID
- Ceph バージョン

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **HOST_NAME** は、**hostname** フィールドに一覧表示されるホストのホスト名に置き換えます。

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **HOST_NAME** は、**hostname** フィールドに一覧表示されるホストのホスト名に置き換えます。
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

Web ブラウザー

Web ブラウザーで以下を入力します。

```
https://CEPH_MANAGER:8080/api/host/HOST_NAME
```

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **HOST_NAME** は、**hostname** フィールドに一覧表示されるホストのホスト名に置き換えます。

プロンプトが表示されたら、ユーザー名とパスワードを入力します。

1.5.2. 設定の変更

本セクションでは、Ceph API プラグインを使用して OSD 設定オプション、OSD の状態、プールに関する情報を変更する方法を説明します。

- [「OSD 設定オプションの変更方法」](#)
- [「OSD の状態を変更する方法」](#)
- [「OSD の重みを再設定する方法」](#)
- [「プールの情報の変更方法」](#)

1.5.2.1. OSD 設定オプションの変更方法

本セクションでは、RESTful プラグインを使用して OSD 設定オプションを変更する方法を説明します。

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **OPTION** を変更するオプションに置き換えます
(**pause**、**noup**、**nodown**、**noout**、**noin**、**nobackfill**、**norecover**、**noscrub**、**nodeep-scrub**)
- **true** または **false** の **VALUE**
- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

```
$ python
```

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **OPTION** を変更するオプションに置き換えます
(**pause**、**noup**、**nodown**、**noout**、**noin**、**nobackfill**、**norecover**、**noscrub**、**nodeep-scrub**)
- **VALUE** は **True** または **False** に置き換えます。
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

1.5.2.2. OSD の状態を変更する方法

本セクションでは、RESTful プラグインを使用して OSD の状態を変更する方法を説明します。

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **STATE** は、変更する状態 (**in** または **up**) に置き換えます。
- **true** または **false** の **VALUE**
- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **osd** フィールドにリストされている OSD の **ID** を持つ ID

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **osd** フィールドにリストされている OSD の **ID** を持つ ID
- **STATE** は、変更する状態 (**in** または **up**) に置き換えます。
- **VALUE** は **True** または **False** に置き換えます。
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

1.5.2.3. OSD の重みを再設定する方法

本セクションでは、OSD の重みを変更する方法を説明します。

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **VALUE** は、新しい重みに置き換えます。
- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **osd** フィールドにリストされている OSD の **ID** を持つ ID

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **osd** フィールドにリストされている OSD の **ID** を持つ ID
- **VALUE** は、新しい重みに置き換えます。
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

1.5.2.4. プールの情報の変更方法

本セクションでは、RESTful プラグインを使用して特定のプールの情報を変更する方法を説明します。

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **OPTION** は、変更するオプションに置き換えます。
- **VALUE** は、オプションの新しい値に置き換えます。
- **USER** は、ユーザー名に置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **pool** フィールドにリストされているプールの **ID** を持つ ID

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **pool** フィールドにリストされているプールの **ID** を持つ ID
- **OPTION** は、変更するオプションに置き換えます。
- **VALUE** は、オプションの新しい値に置き換えます。
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

1.5.3. クラスターの管理

本セクションでは、Ceph API プラグインを使用して OSD でのスクラビングまたはディープスクラビングの初期化、プールの作成、プールからのデータの削除、リクエストの削除、または要求の作成を行う方法を説明します。

- [「OSD でスケジュール済みプロセスを実行する方法」](#)
- [「新規プールの作成方法」](#)
- [「プールの削除方法」](#)

1.5.3.1. OSD でスケジュール済みプロセスを実行する方法

本セクションでは、OSD でのスクラビングやディープスクラビングなどの RESTful API を使用してスケジュール済みプロセスを実行する方法を説明します。

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- 起動するプロセス (**scrub**、**deep-scrub**、または **repair**) を持つ **COMMAND**。プロセスが OSD でサポートされていることを確認します。詳しくは「[OSD でどのプロセスがスケジュールされるのかを指定する方法](#)」を参照してください。
- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **osd** フィールドにリストされている OSD の **ID** を持つ ID

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **osd** フィールドにリストされている OSD の **ID** を持つ ID
- 起動するプロセス (**scrub**、**deep-scrub**、または **repair**) を持つ **COMMAND**。プロセスが OSD でサポートされていることを確認します。詳しくは「[OSD でどのプロセスがスケジュールされるのかを指定する方法](#)」を参照してください。
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

1.5.3.2. 新規プールの作成方法

本セクションでは、RESTful プラグインを使用して新しいプールを作成する方法を説明します。

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **NAME** は、新規プールの名前に置き換えます。
- **NUMBER** は、配置グループの数に置き換えます。
- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **NAME** は、新規プールの名前に置き換えます。
- **NUMBER** は、配置グループの数に置き換えます。
- **USER** は、ユーザー名に置き換えます。
- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

1.5.3.3. プールの削除方法

本セクションでは、RESTful プラグインを使用してプールを削除する方法を説明します。

この要求はデフォルトで禁止されています。そのためには、以下のパラメーターを Ceph 設定ガイドに追加します。

```
mon_allow_pool_delete = true
```

curl コマンド

コマンドラインで、以下を使用します。

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

以下を置き換えます。

- **USER** は、ユーザー名に置き換えます。
- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **pool** フィールドにリストされているプールの **ID** を持つ ID

プロンプトが表示されたら、ユーザーのパスワードを入力します。

自己署名証明書を使用した場合は、**--insecure** オプションを使用します。

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

Python

Python インタープリターで、以下を入力します。

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

以下を置き換えます。

- アクティブな **ceph-mgr** インスタンスを持つノードの IP アドレスまたは短いホスト名を持つ **CEPH_MANAGER**
- **pool** フィールドにリストされているプールの **ID** を持つ ID
- **USER** は、ユーザー名に置き換えます。

- **PASSWORD** は、ユーザーのパスワードに置き換えます。

自己署名証明書を使用した場合は、**verify=False** オプションを使用します。

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

1.6. 関連情報

- API の具体的な詳細は、[付録A Ceph RESTful API 仕様](#) を参照してください。
- GitHub の [API Python スクリプトのテスト](#) を参照してください。

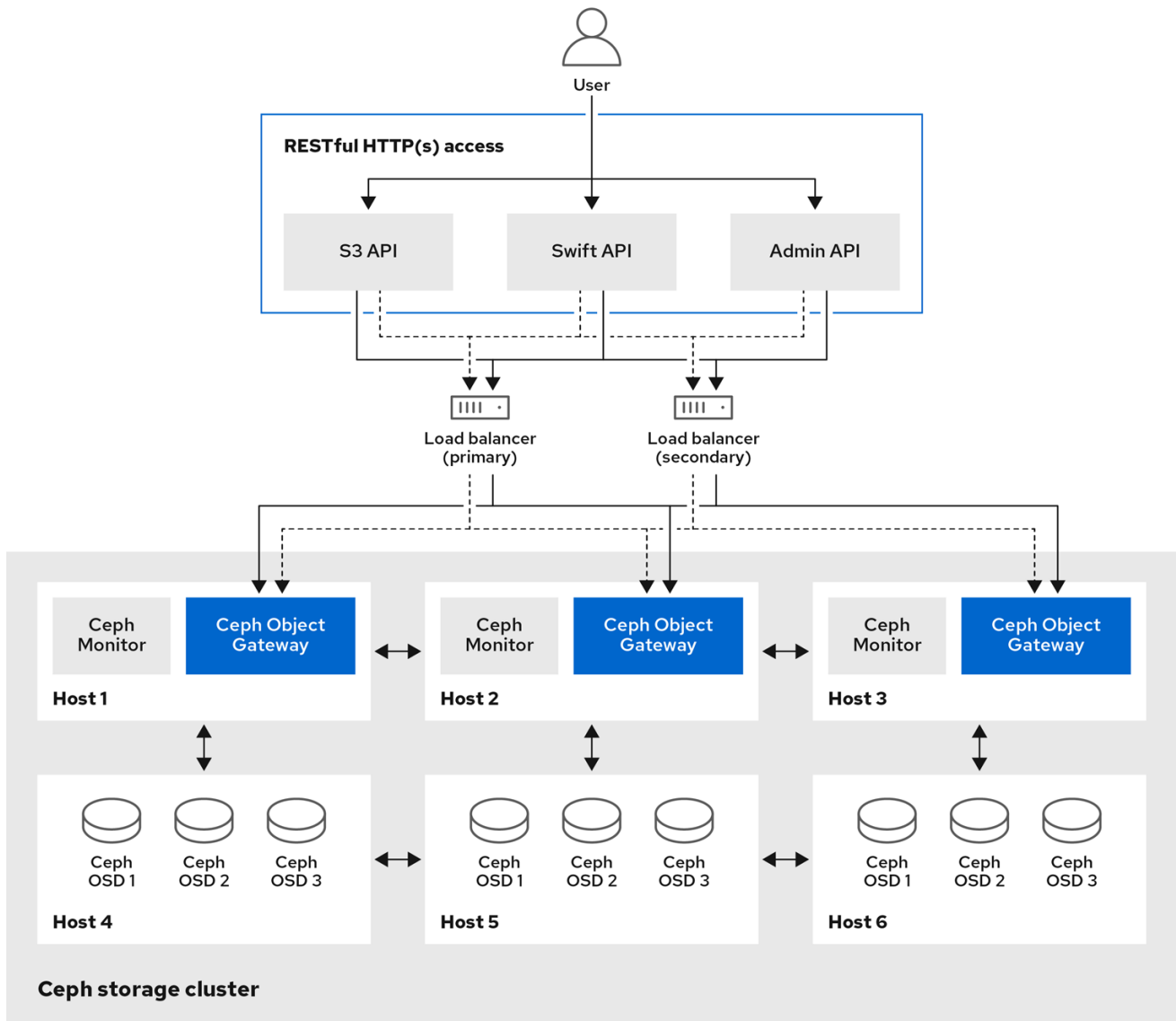
第2章 CEPH OBJECT GATEWAY 管理 API

開発者は、RESTful アプリケーションプログラミングインターフェイス (API) と対話して Ceph Object Gateway を管理することができます。Ceph Object Gateway は、RESTful API の **radosgw-admin** コマンドの機能を利用できます。他の管理プラットフォームと統合できるユーザー、データ、クォータ、および使用方法を管理できます。



注記

Red Hat では、Ceph Object Gateway の設定時にコマンドラインインターフェイスを使用することを推奨します。



250_Ceph_0522

管理 API は以下の機能を提供します。

- **認証要求**
- **ユーザーアカウントの管理**
 - **管理ユーザー**

- [ユーザー情報の取得](#)
- [作成](#)
- [修正](#)
- [削除](#)
- [サブユーザーの作成](#)
- [サブユーザーの変更](#)
- [サブユーザーの削除](#)
- [ユーザーのケーパビリティ管理](#)
 - [追加](#)
 - [削除](#)
- [キー管理](#)
 - [作成](#)
 - [削除](#)
- [バケット管理](#)
 - [バケット情報の取得](#)
 - [インデックスの確認](#)
 - [削除](#)
 - [リンク](#)
 - [リンク解除](#)
 - [ポリシー](#)
- [オブジェクト管理](#)
 - [削除](#)
 - [ポリシー](#)
- [クォータ管理](#)
 - [ユーザーの取得](#)
 - [ユーザーの設定](#)
 - [バケットの取得](#)
 - [バケットの設定](#)
- [使用方法の取得](#)
- [使用方法情報の削除](#)

- [標準エラーレスポンス](#)

2.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。
- RESTful クライアント。

2.2. 管理操作

管理アプリケーションプログラミングインターフェイス (API) リクエストは、設定可能な管理者リソースエントリーポイントで開始する URI で実行されます。管理 API の認可は S3 認可メカニズムを複製します。一部の操作では、ユーザーが特別な管理機能を保持する必要があります。XML または JSON のいずれかのレスポンスエンティティタイプはリクエストの format オプションとして指定され、指定されていないとデフォルトは JSON に設定されます。

例

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

usage=read
```

2.3. 管理認証要求

Amazon の S3 サービスは、アクセスキー、要求ヘッダーのハッシュ、および秘密鍵を使用して要求を認証します。これは、認証されたリクエスト (特に SSL オーバーヘッドのない大規模なアップロード) を提供する利点があります。

S3 API のほとんどのユースケースには、Java や Python Boto 用の Amazon SDK の **AmazonS3Client** などのオープンソースの S3 クライアントを使用します。これらのライブラリーは、Ceph Object Gateway 管理 API をサポートしません。これらのライブラリーをサブクラス化および拡張して、Ceph Admin API をサポートすることができます。一意のゲートウェイクライアントを作成できます。

execute() メソッドの作成

本セクションの [CephAdminAPI](#) のサンプルクラスでは、要求パラメーターの取得、リクエストの認証、Ceph 管理 API を呼び出してレスポンスを受け取ることができる **execute()** メソッドの作成方法を説明します。

CephAdminAPI クラスの例は、商用としてはサポートされず、そのように意図されてもいません。これは説明のみを目的としています。

Ceph Object Gateway の呼び出し

[クライアントコード](#) には、CRUD 操作を示すために Ceph Object Gateway への 5 つの呼び出しが含まれます。

- ユーザーの作成
- ユーザーの取得
- ユーザーの変更

- サブユーザーの作成
- ユーザーを削除します。

この例を使用するには、**httpcomponents-client-4.5.3** Apache HTTP コンポーネントを取得します。たとえば、<http://hc.apache.org/downloads.cgi> からダウンロードできます。その後、tar ファイルを展開して **lib** ディレクトリーに移動し、**JAVA_HOME** ディレクトリーの **/jre/lib/ext** ディレクトリーまたはカスタムクラスパスにコピーします。

CephAdminAPI クラスの例を検査する際に、**execute()** メソッドは HTTP メソッド、リクエストパス、オプションのサブリソース、未指定の場合は **null**、およびパラメーターのマップを取得することに注意してください。サブリソース (例: **subuser**、**key** など) で実行するには、サブリソースを **execute()** メソッドの引数として指定する必要があります。

方法の例を以下に示します。

1. URI をビルドします。
2. HTTP ヘッダー文字列をビルドします。
3. HTTP リクエストをインスタンス化します (例: **PUT**、**POST**、**GET**、**DELETE**)。
4. **Date** ヘッダーを HTTP ヘッダー文字列および要求ヘッダーに追加します。
5. **Authorization** ヘッダーを HTTP リクエストヘッダーに追加します。
6. HTTP クライアントをインスタンス化し、インスタンス化された HTTP リクエストを渡します。
7. 要求を行います。
8. レスポンスを返します。

ヘッダー文字列のビルド

ヘッダー文字列のビルドは、Amazon の S3 認証手順を伴うプロセスの一部です。特に、サンプルメソッドは以下を行います。

1. **PUT**、**POST**、**GET**、**DELETE** などのリクエストタイプを追加します。
2. 日付を追加します。
3. **requestPath** を追加します。

リクエストタイプは、先頭または最後の空白のない大文字である必要があります。空白を削除しないと、認証は失敗します。日付は GMT で表現される必要があります。さもないと、認証に失敗します。

例示的な方法には、他のヘッダーはありません。Amazon S3 認証手順は、**x-amz** ヘッダーの辞書式に並べ替えられます。したがって、**x-amz** ヘッダーを追加する場合は、必ず辞書式で追加する必要があります。

ヘッダー文字列をビルドしたら、次の手順は HTTP リクエストをインスタンス化し、URI を渡すことです。典型的なメソッドは、**PUT** を使用してユーザーおよびサブユーザーを作成し、**GET** を使用してユーザーを取得し、**POST** を使用してユーザーを変更し、**DELETE** を使用してユーザーを削除します。

リクエストをインスタンス化したら、**Date** ヘッダーに続けて **Authorization** ヘッダーを追加します。Amazon の S3 認証は標準の **Authorization** ヘッダーを使用し、以下の構造を持ちます。

Authorization: AWS **ACCESS_KEY:HASH_OF_HEADER_AND_SECRET**

CephAdminAPI のサンプルクラスには **base64Sha1Hmac()** メソッドがあります。これはヘッダー文字列と admin ユーザーの秘密鍵を取得し、SHA1 HMAC を base-64 でエンコードされた文字列として返します。それぞれの **execute()** 呼び出しは、同じコード行を呼び出して **Authorization** ヘッダーをビルドします。

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

以下の **CephAdminAPI** のサンプルクラスでは、アクセスキー、シークレットキー、およびエンドポイントをコンストラクターに渡す必要があります。クラスは実行時に変更するためのアクセスメソッドを提供します。

例

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

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);
    }
}
}

```

後続の **CephAdminAPIClient** の例は、**CephAdminAPI** クラスをインスタンス化する方法、リクエストパラメーターのマップをビルドし、**execute()** メソッドを使用してユーザーを作成、取得、更新、および削除する方法を示しています。

例

```

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",
        "Xac39eCAhITGcCAUreuwe1ZuH5oVQFa51lbEMVoT",
        "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();
}
}
}
}

```

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [S3 認証](#) セクションを参照してください。
- Amazon S3 認証手順の詳細は、Amazon Simple Storage Service ドキュメントの [Signing and Authenticating REST Requests](#) セクションを参照してください。

2.4. 管理ユーザーの作成



重要

Ceph Object Gateway ノードから **radosgw-admin** コマンドを実行するには、ノードに admin キーがあることを確認します。admin キーは、任意の Ceph Monitor ノードからコピーできます。

前提条件

- Ceph Object Gateway ノードへのルートレベルのアクセス。

手順

1. Object Gateway ユーザーを作成します。

構文

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

例

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

radosgw-admin コマンドラインインターフェイスはユーザーを返します。

出力例

```
{
  "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": "gr1VEGIV7rxcP3xvXDfCo4UDwwl2YoNrmtRIIAty"
      }
    ],
    "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. 作成するユーザーに管理権限を割り当てます。

構文

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

例

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

radosgw-admin コマンドラインインターフェイスはユーザーを返します。**"caps"**: には、ユーザーに割り当てられた権限があります。

出力例

```

{
  "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": "gr1VEGIV7rxcP3xvXDfCo4UDwwl2YoNrmtRIIAty"
    }
  ]
}

```



```

    }
  ],
  "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": []
}

```

これで、管理者権限を持つユーザーが作成されます。

2.5. ユーザー情報の取得

ユーザーの情報の取得

機能

users=read

構文

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

リクエストパラメーター

uid

詳細

情報が要求されるユーザー。

型

String

例

foo_user

必須

はい

レスポンスエンティティー

user

詳細

ユーザーデータ情報のコンテナ

型

Container

親

該当なし

user_id

詳細

ユーザー ID。

型

String

親

user

display_name

詳細

ユーザーの表示名。

型

String

親

user

suspended

詳細

ユーザーが一時停止してきめる場合は True。

型

Boolean

親

user

max_buckets

詳細

ユーザーが所有するバケットの最大数。

型

Integer

親

user

subusers

詳細

このユーザーアカウントに関連付けられたサブユーザー。

型

Container

親

user

keys

詳細

このユーザーアカウントに関連付けられた S3 キー。

型

Container

親

user

swift_keys

詳細

このユーザーアカウントに関連付けられた Swift 鍵。

型

Container

親

user

caps

詳細

ユーザー機能。

型

Container

親

user

成功すると、応答にはユーザー情報が含まれます。

特別なエラーレスポンス

なし。

2.6. ユーザーの作成

新しいユーザーを作成します。デフォルトでは、S3 キーペアが自動的に作成され、レスポンスで返されます。**access-key** または **secret-key** のいずれかのみを指定すると、省略キーが自動的に生成されます。デフォルトでは、生成されたキーは、既存のキーペアを置き換えることなくキーリングに追加されます。**access-key** が指定され、ユーザーが所有する既存のキーを参照すると、そのキーは変更されません。

機能

```
`users=write`
```

構文

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

リクエストパラメーター

uid

詳細

作成されるユーザー ID。

型

String

例

foo_user

必須

はい

display-name

詳細

作成するユーザーの表示名。

型

String

例

foo_user

必須

はい

email

詳細

ユーザーに関連付けられたメールアドレス。

型

String

例

foo@bar.com

必須

いいえ

key-type

詳細

生成されるキータイプ。オプションは swift、s3 (デフォルト) です。

型

String

例

s3 [s3]

必須

いいえ

access-key

詳細

アクセスキーを指定します。

型

String

例

ABCD0EF12GHIJ2K34LMN

必須

いいえ

secret-key

詳細

シークレットキーを指定します。

型

String

例

0AbCDEFG1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8

必須

いいえ

user-caps

詳細

ユーザー機能。

型

String

例

usage=read, write; users=read

必須

いいえ

generate-key

詳細

新しいキーペアを生成し、既存のキーリングに追加します。

型

Boolean

例

True [True]

必須

いいえ

max-buckets

詳細

ユーザーが所有できるバケットの最大数を指定します。

型

Integer

例

500 [1000]

必須

いいえ

suspended

詳細

ユーザーが一時停止するかどうかを指定します。

型

Boolean

例

False [False]

必須

いいえ

レスポンスエンティティ

user

詳細

ユーザーが一時停止するかどうかを指定します。

型

Boolean

親

いいえ

user_id

詳細

ユーザー ID。

型

String

親

user

display_name

詳細

ユーザーの表示名。

型

String

親**user****suspended****詳細**

ユーザーが一時停止してきえる場合は True。

型

Boolean

親**user****max_buckets****詳細**

ユーザーが所有するバケットの最大数。

型

Integer

親**user****subusers****詳細**

このユーザーアカウントに関連付けられたサブユーザー。

型

Container

親**user****keys****詳細**

このユーザーアカウントに関連付けられた S3 キー。

型

Container

親**user****swift_keys****詳細**

このユーザーアカウントに関連付けられた Swift 鍵。

型

Container

親

user

caps

詳細

ユーザー機能。

型

Container

親

成功すると、応答にはユーザー情報が含まれます。

特別なエラーレスポンス

UserExists

詳細

既存ユーザーの作成を試行。

コード

409 Conflict

InvalidAccessKey

詳細

無効なアクセスキーが指定されている。

コード

400 Bad Request

InvalidKeyType

詳細

無効なキータイプが指定されている。

コード

400 Bad Request

InvalidSecretKey

詳細

無効なシークレットキーが指定されている。

コード

400 Bad Request

KeyExists

詳細

提供されたアクセスキーが存在し、別のユーザーに属している。

コード

409 Conflict

EmailExists

詳細

提供されるメールアドレスが存在する。

コード

409 Conflict

InvalidCap

詳細

無効な管理者機能の付与を試行。

コード

400 Bad Request

関連情報

- サブユーザーの作成は、[Red Hat Ceph Storage 開発者ガイド](#)を参照してください。

2.7. ユーザーの変更

既存ユーザーの変更

機能

`users=write`

構文

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

リクエストパラメーター

uid

詳細

作成されるユーザー ID。

型

String

例

foo_user

必須

はい

display-name

詳細

作成するユーザーの表示名。

型

String

例

foo_user**必須**

はい

email**詳細**

ユーザーに関連付けられたメールアドレス。

型

String

例**foo@bar.com****必須**

いいえ

generate-key**詳細**

新しいキーペアを生成し、既存のキーリングに追加します。

型

Boolean

例

True [False]

必須

いいえ

access-key**詳細**

アクセスキーを指定します。

型

String

例**ABCD0EF12GHIJ2K34LMN****必須**

いいえ

secret-key**詳細**

シークレットキーを指定します。

型

String

例**0AbCDEFG1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8****必須**

いいえ

key-type**詳細**

生成されるキータイプ。オプションは swift、s3 (デフォルト) です。

型

String

例

s3

必須

いいえ

user-caps**詳細**

ユーザー機能。

型

String

例

usage=read, write; users=read

必須

いいえ

max-buckets**詳細**

ユーザーが所有できるバケットの最大数を指定します。

型

Integer

例

500 [1000]

必須

いいえ

suspended**詳細**

ユーザーが一時停止するかどうかを指定します。

型

Boolean

例

False [False]

必須

いいえ

レスポンスエンティティ**user**

詳細

ユーザーが一時停止するかどうかを指定します。

型

Boolean

親

いいえ

user_id**詳細**

ユーザー ID。

型

String

親

user

display_name**詳細**

ユーザーの表示名。

型

String

親

user

suspended**詳細**

ユーザーが一時停止してきめる場合は True。

型

Boolean

親

user

max_buckets**詳細**

ユーザーが所有するバケットの最大数。

型

Integer

親

user

subusers**詳細**

このユーザーアカウントに関連付けられたサブユーザー。

型

Container

親

user

keys

詳細

このユーザーアカウントに関連付けられた S3 キー。

型

Container

親

user

swift_keys

詳細

このユーザーアカウントに関連付けられた Swift 鍵。

型

Container

親

user

caps

詳細

ユーザー機能。

型

Container

親

成功すると、応答にはユーザー情報が含まれます。

特別なエラーレスポンス

InvalidAccessKey

詳細

無効なアクセスキーが指定されている。

コード

400 Bad Request

InvalidKeyType

詳細

無効なキータイプが指定されている。

コード

400 Bad Request

InvalidSecretKey

詳細

無効なシークレットキーが指定されている。

コード

400 Bad Request

KeyExists**詳細**

提供されたアクセスキーが存在し、別のユーザーに属している。

コード

409 Conflict

EmailExists**詳細**

提供されるメールアドレスが存在する。

コード

409 Conflict

InvalidCap**詳細**

無効な管理者機能の付与を試行。

コード

400 Bad Request

関連情報

- サブユーザーの変更については、[Red Hat Ceph Storage 開発者ガイド](#)を参照してください。

2.8. ユーザーの削除

既存のユーザーを削除します。

機能

```
`users=write`
```

構文

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

リクエストパラメーター**uid****詳細**

削除するユーザー ID。

型

String

例

foo_user

必須

はい

purge-data

詳細

指定すると、ユーザーに属するバケットとオブジェクトも削除されます。

型

Boolean

例

True

必須

いいえ

レスポンスエンティティ

なし。

特別なエラーレスポンス

なし。

関連情報

- サブユーザーの削除については、[Red Hat Ceph Storage 開発者ガイド](#)を参照してください。

2.9. サブユーザーの作成

Swift API を使用するクライアントに主に役立つ新しいサブユーザーを作成します。



注記

有効なリクエストには、**gen-subuser** または **subuser** のいずれかが必要です。通常、サブユーザーには、**access** を指定してパーミッションを付与する必要があります。ユーザー作成 (**subuser** が **secret** なしで指定されている場合) と同様に、シークレットキーは自動的に生成されます。

機能

`users=write`

構文

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

リクエストパラメーター

uid

詳細

サブユーザーを作成するユーザー ID。

型

String

例

foo_user

必須

はい

subuser

詳細

作成するサブユーザー ID を指定します。

型

String

例

sub_foo

必須

必須 (または **gen-subuser**)

gen-subuser

詳細

作成するサブユーザー ID を指定します。

型

String

例

sub_foo

必須

必須 (または **gen-subuser**)

secret-key

詳細

シークレットキーを指定します。

型

String

例

0AbCDEfg1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8

必須

いいえ

key-type

詳細

生成されるキータイプ。オプションは `swift` (デフォルト)、`s3` です。

型

String

例

swift [**swift**]

必須

いいえ

access**詳細**

サブユーザーのアクセスパーミッションを設定する場合は、**read**, **write**, **readwrite**, **full** のいずれかである必要があります。

型

String

例

read

必須

いいえ

generate-secret**詳細**

シークレットキーを生成します。

型

Boolean

例

True [False]

必須

いいえ

レスポンスエンティティ**subusers****詳細**

ユーザーアカウントに関連付けられたサブユーザー

型

Container

親

該当なし

permissions**詳細**

ユーザーアカウントへのサブユーザーアクセス

型

String

親**subusers**

成功すると、レスポンスにはサブユーザー情報が含まれます。

特別なエラーレスポンス**SubuserExists****詳細**

指定したサブユーザーが存在する。

コード

409 Conflict

InvalidKeyType**詳細**

無効なキータイプが指定されている。

コード

400 Bad Request

InvalidSecretKey**詳細**

無効なシークレットキーが指定されている。

コード

400 Bad Request

InvalidAccess**詳細**

無効なサブユーザーアクセスが指定されている。

コード

400 Bad Request

2.10. サブユーザーの変更

既存のサブユーザーを変更します。

機能

```
`users=write`
```

構文

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

リクエストパラメーター

uid

詳細

サブユーザーを作成するユーザー ID。

型

String

例

foo_user

必須

はい

subuser

詳細

変更するサブユーザー ID。

型

String

例

sub_foo

必須

generate-secret

詳細

サブユーザーの新しい秘密鍵を生成し、既存のキーを置き換えます。

型

Boolean

例

True [False]

必須

いいえ

secret

詳細

シークレットキーを指定します。

型

String

例

0AbCDEfg1h2i34JkIM5nop6QrSTUV+WxyzaBC7D8

必須

いいえ

key-type

詳細

生成されるキータイプ。オプションは `swift` (デフォルト)、`s3` です。

型

String

例

swift [**swift**]

必須

いいえ

access**詳細**

サブユーザーのアクセスパーミッションを設定する場合は、**read**, **write**, **readwrite**, **full** のいずれかである必要があります。

型

String

例

read

必須

いいえ

レスポンスエンティティ**subusers****詳細**

ユーザーアカウントに関連付けられたサブユーザー

型

Container

親

該当なし

id**詳細**

サブユーザー ID

型

String

親

subusers

permissions**詳細**

ユーザーアカウントへのサブユーザーアクセス

型

String

親

subusers

成功すると、レスポンスにはサブユーザー情報が含まれます。

特別なエラーレスポンス

InvalidKeyType

詳細

無効なキータイプが指定されている。

コード

400 Bad Request

InvalidSecretKey

詳細

無効なシークレットキーが指定されている。

コード

400 Bad Request

InvalidAccess

詳細

無効なサブユーザーアクセスが指定されている。

コード

400 Bad Request

2.11. サブユーザーの削除

既存のサブユーザーを削除します。

機能

```
`users=write`
```

構文

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

リクエストパラメーター

uid

詳細

削除するユーザー ID。

型

String

例

foo_user**必須**

はい

subuser**詳細**

削除されるサブユーザー ID。

型

String

例**sub_foo****必須**

はい

purge-keys**詳細**

サブユーザーに属する鍵を削除します。

型

Boolean

例

True [True]

必須

いいえ

レスポンスエンティティ

なし。

特別なエラーレスポンス

なし。

2.12. ユーザーへの機能の追加

指定したユーザーに管理ケイパビリティを追加します。

機能``users=write``**構文**

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

リクエストパラメーター

uid**詳細**

管理機能を追加するユーザー ID。

型

String

例

foo_user

必須

はい

user-caps**詳細**

ユーザーに追加する管理機能。

型

String

例

usage=read, write

必須

はい

レスポンスエンティティ**user****詳細**

ユーザーデータ情報のコンテナ

型

Container

親

該当なし

user_id**詳細**

ユーザー ID

型

String

親

user

caps**詳細**

ユーザー機能。

型

Container

親

user

成功すると、レスポンスにはユーザーのケイパビリティが含まれます。

特別なエラーレスポンス

InvalidCap

詳細

無効な管理者機能の付与を試行。

コード

400 Bad Request

2.13. ユーザーからの機能の削除

指定したユーザーから管理ケイパビリティを削除します。

機能

```
`users=write`
```

構文

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

リクエストパラメーター

uid

詳細

管理機能を削除するユーザー ID。

型

String

例

foo_user

必須

はい

user-caps

詳細

ユーザーから削除する管理機能。

型

String

例

usage=read, write

必須

はい

レスポンスエンティティ**user****詳細**

ユーザーデータ情報のコンテナ

型

Container

親

該当なし

user_id**詳細**

ユーザー ID。

型

String

親**user****caps****詳細**

ユーザー機能。

型

Container

親**user**

成功すると、レスポンスにはユーザーのケイパビリティが含まれます。

特別なエラーレスポンス**InvalidCap****詳細**

無効な管理機能の削除を試行します。

コード

400 Bad Request

NoSuchCap**詳細**

ユーザーは指定されている機能进行处理しません。

コード

404 Not Found

2.14. キーの作成

新しいキーを作成します。**subuser** を指定すると、デフォルトで作成されたキーは swift タイプになります。**access-key** または **secret-key** のいずれかのみが指定された場合、コミットされたキーは自動的に生成されます。**secret-key** のみが指定されている場合、**access-key** は自動的に生成されます。デフォルトでは、生成されたキーは、既存のキーペアを置き換えることなくキーリングに追加されます。**access-key** が指定され、ユーザーが所有する既存のキーを参照すると、そのキーは変更されます。レスポンスは、作成された鍵と同じタイプの鍵をすべて一覧表示するコンテナです。



注記

swift キーの作成時に、**access-key** オプションを指定しても効果はありません。また、ユーザーまたはサブユーザーごとに 1 つの swift キーのみを保持することができます。

機能

```
`users=write`
```

構文

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

リクエストパラメーター

uid

詳細

新しいキーを受け取るユーザー ID。

型

String

例

foo_user

必須

はい

subuser

詳細

新しいキーを受け取るサブユーザー ID。

型

String

例

sub_foo

必須

いいえ

key-type

詳細

生成されるキータイプ。オプションは swift、s3 (デフォルト) です。

型

String

例

s3 [s3]

必須

いいえ

access-key

詳細

アクセスキーを指定します。

型

String

例

AB01C2D3EF45G6H7IJ8K

必須

いいえ

secret-key

詳細

シークレットキーを指定します。

型

String

例

0ab/CdeFGhij1klmnopqRSTUv1WxyZabcDEFgHij

必須

いいえ

generate-key

詳細

新しいキーペアを生成し、既存のキーリングに追加します。

型

Boolean

例

True [**True**]

必須

いいえ

レスポンスエンティティ

keys

詳細

このユーザーアカウントに関連付けられたタイプのキー。

型

Container

親

該当なし

user**詳細**

キーに関連付けられたユーザーアカウント。

型

String

親

keys

access-key**詳細**

アクセスキー。

型

String

親

keys

secret-key**詳細**

シークレットキー。

型

String

親

keys

特別なエラーレスポンス**InvalidAccessKey****詳細**

無効なアクセスキーが指定されている。

コード

400 Bad Request

InvalidKeyType**詳細**

無効なキータイプが指定されている。

コード

400 Bad Request

InvalidSecretKey

詳細

無効なシークレットキーが指定されている。

コード

400 Bad Request

InvalidKeyType

詳細

無効なキータイプが指定されている。

コード

400 Bad Request

KeyExists

詳細

提供されたアクセスキーが存在し、別のユーザーに属している。

コード

409 Conflict

2.15. 鍵の削除

既存のキーを削除します。

機能

```
`users=write`
```

構文

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

リクエストパラメーター

access-key

説明

削除する S3 キーペアに属する S3 アクセスキー。

型

String

例

AB01C2D3EF45G6H7IJ8K

必須

はい

uid**詳細**

キーの削除元のユーザー。

型

String

例

foo_user

必須

いいえ

subuser**詳細**

キーの削除元のサブユーザー。

型

String

例

sub_foo

必須

いいえ

key-type**詳細**

削除するキータイプ。オプションは swift、s3 です。

**注記**

swift キーを削除するために必要です。

型

String

例

swift

必須

いいえ

特別なエラーレスポンス

なし。

レスポンスエンティティー

なし。

2.16. バケット通知

ストレージ管理者は、これらの API を使用してバケット通知メカニズムの設定および制御インターフェイスを提供できます。API トピックは、特定のエンドポイントの定義が含まれる名前が付けられたオブジェクトです。バケット通知では、トピックを特定のバケットに関連付けます。[S3 バケット操作](#) セクションでは、バケット通知の詳細が表示されます。



注記

すべてのトピックアクションでは、パラメーターは URL エンコードされ、**application/x-www-form-urlencoded** コンテンツタイプを使用してメッセージのボディで送信されます。



注記

トピックの更新を有効にするには、トピックにすでに関連付けられているバケット通知を再作成する必要があります。

2.16.1. 前提条件

- Ceph Object Gateway 上にバケット通知を作成します。

2.16.2. バケット通知の概要

バケット通知により、バケットで特定のイベントが発生した場合に、Ceph Object Gateway から情報を送る方法が提供されます。バケット通知は HTTP、AMQP0.9.1、および Kafka エンドポイントに送信できます。特定バケットおよび特定のトピック上のイベントのバケット通知を送信するために、通知エントリーを作成する必要があります。バケット通知は、イベントタイプのサブセットに作成することも、デフォルトですべてのイベントタイプに対して作成できます。バケット通知は、キーの接頭辞または接尾辞、キーに一致する正規表現、オブジェクトに割り当てられたメタデータ属性、またはオブジェクトタグに基づいてイベントをフィルタリングできます。バケット通知には、バケット通知メカニズムの設定および制御インターフェイスを提供する REST API があります。

2.16.3. 永続通知

永続的な通知により、Ceph ObjectGateway からトピックで設定されたエンドポイントへの通知の信頼性の高い非同期配信が可能になります。エンドポイントへの配信は、リクエスト中に同期的に実行されるため、通常の通知も信頼性があります。永続的な通知により、Ceph Object Gateway はエンドポイントがダウンした場合や、操作中にネットワークの問題がある場合でも、通知は送信を再試行します。これは、エンドポイントに正常に配信されない場合に通知が再試行されます。通知は、通知操作に関連するその他のアクションがすべて成功する場合にのみ送信されます。エンドポイントが長期間ダウンすると、通知キューがいっぱいになり、これらのエンドポイントの通知が設定された S3 操作が失敗します。



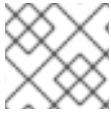
注記

kafka-ack-level=none を使用すると、メッセージの失敗が示されないため、ブローカーが停止している間に送信されたメッセージは、ブローカーの再起動時に再試行されます。ブローカーがもう一度開始されると、新しい通知のみが表示されます。

2.16.4. トピックの作成

バケット通知を作成する前に、トピックを作成できます。トピックは Simple Notification Service (SNS) エンティティーで、すべてのトピック操作 (つまり **create**、**delete**、**list**、および **get**) は SNS 操作です。トピックには、バケット通知の作成時に使用されるエンドポイントパラメーターが必要です。

リクエストが正常に行われると、レスポンスには、バケット通知要求でこのトピックを参照するために後で利用できるトピックの Amazon Resource Name (ARN) が含まれます。



注記

topic_arn はバケット通知設定を提供し、トピックの作成後に生成されます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。
- ルートレベルのアクセス。
- Ceph Object Gateway のインストール
- ユーザーアクセスキーおよびシークレットキー。
- エンドポイントパラメーター。

手順

- 以下の要求形式でトピックを作成します。

構文

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

リクエストパラメーターを以下に示します。

- Endpoint:** 通知を送信するエンドポイントの URL。
- OpaqueData:** 不透明なデータはトピック設定で設定され、トピックによって発生するすべての通知に追加されます。
- persistent:** このエンドポイントへの通知が永続的 (非同期) であるかを示します。デフォルト値は **false** です。
- HTTP エンドポイント:
 - URL:** `https://FQDN:PORT`
 - ポートのデフォルト:** HTTP または HTTPS にそれぞれ 80 または 443 を使用します。
 - verify-ssl:** サーバー証明書がクライアントによって検証されているかどうかを示します。デフォルトでは **true** です。

- AMQP0.9.1 エンドポイント:
 - **URL**: `amqp://USER:PASSWORD@FQDN:PORT[/VHOST]`.
 - ユーザーおよびグループのデフォルト値はそれぞれ **guest** と **guest** です。
 - ユーザーおよびパスワードの詳細は HTTPS 経由で提供する必要があります。そうしないと、トピック作成要求は拒否されます。
 - **Port のデフォルト値** は 5672 です。
 - **vhost** のデフォルトは / です。
 - **amqp-exchange**: 交換は存在し、トピックに基づいてメッセージをルーティングする必要があります。これは AMQP0.9.1 の必須パラメーターです。同じエンドポイントを参照するさまざまなトピックが同じ交換を使用する必要があります。
 - **amqp-ack-level**: 最終宛先に送信される前にメッセージがブローカーで永続化される可能性があるため、終了確認は不要です。承認メソッドは 3 つあります。
 - **none**: ブローカーに送信された場合にメッセージが **配信されている** と見なされます。
 - **broker**: デフォルトでは、メッセージはブローカーによって確認応答されると **配信されている** と見なされます。
 - **routable**: ブローカーがコンシューマーにルーティングできる場合、メッセージは **配信されている** と見なされます。



注記

特定のパラメーターのキーと値は、同じ行または特定の順序で存在する必要はありませんが、同じインデックスを使用する必要があります。属性インデックスは、特定の値から順番にしたり、開始したりする必要はありません。



注記

topic-name は AMQP トピックに使用されます。

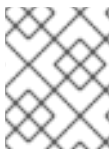
- Kafka エンドポイント:
 - **URL**: `kafka://USER:PASSWORD@FQDN:PORT`.
 - **use-ssl** がデフォルトで **false** に設定される場合。 **use-ssl** が **true** に設定されている場合は、ブローカーへの接続にセキュアな接続が使用されます。
 - **ca-location** が指定され、セキュアな接続が使用される場合は、ブローカーを認証するために、デフォルトの CA ではなく、指定された CA が使用されます。
 - ユーザーおよびパスワードは HTTP[S] でのみ提供できます。そうしないと、トピック作成要求は拒否されます。
 - ユーザーおよびパスワードは **use-ssl** とのみ提供でき、ブローカーへの接続に失敗していました。
 - **Port のデフォルト値** は 9092 です。

- **kafka-ack-level**: 最終宛先に送信される前にメッセージがブローカーで永続化される可能性があるため、終了確認は不要です。承認メソッドは2つあります。
 - **none**: ブローカーに送信された場合にメッセージが **配信されている** と見なされます。
 - **broker**: デフォルトでは、メッセージはブローカーによって確認応答されると **配信されている** と見なされます。

応答形式の例を以下に示します。

例

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



注記

レスポンスのトピックの Amazon Resource Name (ARN) の形式は、**arn:aws:sns:ZONE_GROUP:TENANT:TOPIC** になります。

以下は AMQP0.9.1 エンドポイントの例になります。

例

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

2.16.5. トピック情報の取得

指定したトピックに関する情報を返します。これには、指定されている場合にはエンドポイント情報を含めることができます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。
- ルートレベルのアクセス。
- Ceph Object Gateway のインストール
- ユーザーアクセスキーおよびシークレットキー。
- エンドポイントパラメーター。

手順

1. 以下の要求形式でトピック情報を取得します。

構文

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

レスポンスフォーマットの例を以下に示します。

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

以下は、タグおよび定義です。

- **User:** トピックを作成したユーザーの名前。
- **Name:** トピックの名前。
- JSON 形式のエンドポイントには以下が含まれます。
 - **EndpointAddress:** エンドポイントの URL。エンドポイント URL にユーザーおよびパスワード情報が含まれる場合、リクエストは HTTPS 経由で行う必要があります。それ以外の場合、トピックの取得要求は拒否されます。
 - **EndPointArgs:** エンドポイント引数。
 - **EndpointTopic:** エンドポイントに送信されるトピック名は、上記の例のトピック名とは異なる場合があります。
 - **HasStoredSecret:** エンドポイントの URL にユーザーおよびパスワード情報が含まれる場合に **true**。
 - **永続:** トピックが永続する場合は **true**。
- **TopicArn:** Topic ARN。
- **OpaqueData:** これはトピック上の不透明なデータセットです。

2.16.6. トピックの一覧表示

ユーザーが定義したトピックを一覧表示します。

前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。
- ルートレベルのアクセス。
- Ceph Object Gateway のインストール
- ユーザーアクセスキーおよびシークレットキー。
- エンドポイントパラメーター。

手順

- 以下の要求形式でトピック情報を一覧表示します。

構文

```
POST
Action=ListTopics
```

レスポンスフォーマットの例を以下に示します。

```
<ListTopicsResponse xmlns="https://sns.amazonaws.com/doc/2020-03-31/">
  <ListTopicsResult>
    <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>
```



注記

エンドポイント URL にユーザーおよびパスワード情報が含まれる場合は、トピックのいずれかで要求を行う必要があります。そうしないと、トピック一覧の要求は拒否されます。

2.16.7. トピックの削除

削除したトピックを削除すると、操作はなく、失敗は発生しません。

前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。
- ルートレベルのアクセス。
- Ceph Object Gateway のインストール
- ユーザーアクセスキーおよびシークレットキー。
- エンドポイントパラメーター。

手順

1. 以下の要求形式でトピックを削除します。

構文

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

レスポンスフォーマットの例を以下に示します。

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

2.16.8. イベントレコード

イベントは、Ceph Object Gateway によって行われる操作に関する情報を保持し、選択したエンドポイント (HTTP、HTTPS、Kafka、または AMQ0.9.1 など) 上のペイロードとして送信されます。イベントレコードは JSON 形式になります。

例

```
{
  "Records": [
    {
      "eventVersion": "2.1",
      "eventSource": "ceph:s3",
      "awsRegion": "us-east-1",
      "eventTime": "2019-11-22T13:47:35.124724Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "tester"
      },
      "requestParameters": {
        "sourceIPAddress": ""
      },
      "responseElements": {
```

```

    "x-amz-request-id":"503a4c37-85eb-47cd-8681-2817e80b4281.5330.903595",
    "x-amz-id-2":"14d2-zone1-zonegroup1"
  },
  "s3":{
    "s3SchemaVersion":"1.0",
    "configurationId":"mynotif1",
    "bucket":{
      "name":"mybucket1",
      "ownerIdentity":{
        "principalId":"tester"
      },
      "arn":"arn:aws:s3:us-east-1::mybucket1",
      "id":"503a4c37-85eb-47cd-8681-2817e80b4281.5332.38"
    },
    "object":{
      "key":"myimage1.jpg",
      "size":"1024",
      "eTag":"37b51d194a7513e45b56f6524f2d51f2",
      "versionId":"",
      "sequencer": "F7E6D75DC742D108",
      "metadata":[],
      "tags":[]
    }
  },
  "eventId":"",
  "opaqueData":"me@example.com"
}
}

```

以下はイベントレコードのキーおよびその定義です。

- **awsRegion:** Zonegroup。
- **eventTime:** イベントがトリガーされたタイミングを示すタイムスタンプ。
- **eventName:** イベントのタイプ。
- **userIdentity.principalId:** イベントを開始したユーザーの ID。
- **requestParameters.sourceIPAddress:** イベントをトリガーしたクライアントの IP アドレス。
このフィールドはサポートされません。
- **responseElements.x-amz-request-id:** イベントをトリガーしたリクエスト ID。
- **responseElements.x_amz_id_2:** イベントがトリガーされた Ceph Object Gateway の IP アドレスID 形式は **RGWID-ZONE-ZONEGROUP** です。
- **s3.configurationId:** イベントを作成した通知 ID。
- **s3.bucket.name:** バケットの名前。
- **s3.bucket.ownerIdentity.principalId:** バケットの所有者。
- **s3.bucket.arn:** バケットの Amazon Resource Name(ARN)。
- **s3.bucket.id:** バケットのアイデンティティ。

- **s3.object.key**: オブジェクトキー。
- **s3.object.size**: オブジェクトのサイズ
- **s3.object.eTag**: オブジェクト etag。
- **s3.object.version**: バージョン化されたバケットのオブジェクトバージョン。
- **s3.object.sequencer**: 16 進数形式でオブジェクトごとの変更識別子を増加させます。
- **s3.object.metadata**: **x-amz-meta** として送信されるオブジェクトにメタデータセット。
- **s3.object.tags**: オブジェクトに設定されたタグ。
- **s3.eventId**: イベントの一意のアイデンティティ
- **s3.opaqueData**: Opaque データはトピック設定で設定され、トピックによってトリガーされるすべての通知に追加されます。

関連情報

- 詳細は、[Event Message Structure](#) を参照してください。

2.16.9. サポートされるイベントタイプ

以下のイベントタイプがサポートされます。

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

2.17. バケット情報の取得

既存のバケットのサブセットに関する情報を取得します。**uid** が **bucket** なしで指定されると、そのユーザーに属するすべてのバケットが返されます。**bucket** のみが指定されている場合は、その特定のバケットの情報を取得します。

機能

```
`buckets=read`
```

構文

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

リクエストパラメーター

bucket

詳細

情報を返すバケット。

型

String

例

foo_bucket

必須

いいえ

uid

詳細

バケット情報を取得するユーザー。

型

String

例

foo_user

必須

いいえ

stats

詳細

バケットの統計を返します。

型

Boolean

例

True [False]

必須

いいえ

レスポンスエンティティ

stats

詳細

バケットごとの情報

型

Container

親

該当なし

buckets

詳細

1つ以上のバケットコンテナの一覧が含まれます。

型

Container

親

buckets

bucket

詳細

単一バケット情報用のコンテナ。

型

Container

親

buckets

name

詳細

バケットの名前。

型

String

親

bucket

pool

詳細

バケットが保存されているプール。

型

String

親

bucket

id

詳細

一意のバケット ID。

型

String

親

bucket

marker

詳細

内部バケットタグ。

型

String

親

bucket

owner**詳細**

バケット所有者のユーザー ID。

型

String

親

bucket

使用状況**詳細**

ストレージの使用情報。

型

Container

親

bucket

index**詳細**

バケットインデックスのステータス。

型

String

親

bucket

成功すると、要求はバケット情報と共にバケットコンテナを返します。

特別なエラーレスポンス**IndexRepairFailed****詳細**

バケットインデックスの修復に失敗しました。

コード

409 Conflict

2.18. バケットインデックスを確認します。

既存のバケットのインデックスを確認します。



注記

check-objects で複数パートオブジェクトアカウンティングを確認するには、**fix** を True に設定する必要があります。

機能

buckets=write

構文

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

リクエストパラメーター

bucket

詳細

情報を返すバケット。

型

String

例

foo_bucket

必須

はい

check-objects

詳細

複数パートオブジェクトアカウンティングを確認します。

型

Boolean

例

True [False]

必須

いいえ

fix

詳細

また、チェック時にバケットインデックスも修正します。

型

Boolean

例

False [False]

必須

いいえ

レスポンスエンティティ

index

詳細

バケットインデックスのステータス。

型

String

特別なエラーレスポンス

IndexRepairFailed

詳細

バケットインデックスの修復に失敗しました。

コード

409 Conflict

2.19. バケットの削除

既存のバケットを削除します。

機能

```
`buckets=write`
```

構文

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

リクエストパラメーター

bucket

詳細

削除するバケット。

型

String

例

foo_bucket

必須

はい

purge-objects

説明

削除する前に、バケットのオブジェクトを削除してください。

型

Boolean

例

True [False]

必須

いいえ

レスポンスエンティティ

なし。

特別なエラーレスポンス**BucketNotEmpty****詳細**

空でないバケットの削除を試行しました。

コード

409 Conflict

ObjectRemovalFailed**詳細**

オブジェクトを削除できません。

コード

409 Conflict

2.20. バケットのリンク

バケットを指定ユーザーにリンクし、直前のユーザーからバケットのリンクを解除します。

機能

```
`buckets=write`
```

構文

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

リクエストパラメーター**bucket****詳細**

リンクを解除するバケット。

型

String

例

foo_bucket

必須

はい

uid

詳細

バケットをリンクするユーザー ID。

型

String

例

foo_user

必須

はい

レスポンスエンティティ

bucket

詳細

単一バケット情報用のコンテナ。

型

Container

親

該当なし

name

詳細

バケットの名前。

型

String

親

bucket

pool

詳細

バケットが保存されているプール。

型

String

親

bucket

id

詳細

一意のバケット ID。

型

String

親

bucket

marker

詳細

内部バケットタグ。

型

String

親

bucket

owner

詳細

バケット所有者のユーザー ID。

型

String

親

bucket

使用状況

詳細

ストレージの使用情報。

型

Container

親

bucket

index

詳細

バケットインデックスのステータス。

型

String

親

bucket

特別なエラーレスポンス

BucketUnlinkFailed

詳細

指定されたユーザーからバケットのリンクを解除できません。

コード

409 Conflict

BucketLinkFailed

詳細

バケットを指定されたユーザーにリンクできません。

コード

409 Conflict

2.21. バケットのリンクを解除します。

指定されたユーザーからバケットのリンクを解除します。主にバケットの所有権を変更するのに役立ちます。

機能

```
`buckets=write`
```

構文

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

リクエストパラメーター

bucket

詳細

リンクを解除するバケット。

型

String

例

foo_bucket

必須

はい

uid

詳細

バケットをリンクするユーザー ID。

型

String

例

foo_user

必須

はい

レスポンスエンティティ

なし。

特別なエラーレスポンス

BucketUnlinkFailed

詳細

指定されたユーザーからバケットのリンクを解除できません。

型

409 Conflict

2.22. バケットまたはオブジェクトポリシーを取得する

オブジェクトまたはバケットのポリシーを読み取ります。

機能

```
`buckets=read`
```

構文

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

リクエストパラメーター

bucket

詳細

ポリシーを読み取るバケット。

型

String

例

foo_bucket

必須

はい

object

詳細

ポリシーの読み取り元となるオブジェクト。

型

String

例

foo.txt

必須

いいえ

レスポンスエンティティ

policy

詳細

アクセス制御ポリシー。

型

Container

親

該当なし

成功した場合には、オブジェクトまたはバケットポリシーを返します。

特別なエラーレスポンス

IncompleteBody

詳細

バケットポリシー要求にバケットが指定されていないか、またはオブジェクトがオブジェクトポリシー要求に指定されていません。

コード

400 Bad Request

2.23. オブジェクトの削除

既存のオブジェクトを削除します。



注記

所有者を一時停止せずに指定する必要はありません。

機能

```
`buckets=write`
```

構文

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

リクエストパラメーター

bucket

詳細

削除されるオブジェクトを含むバケット。

型

String

例

foo_bucket

必須

はい

object

詳細

削除するオブジェクト。

型

String

例

foo.txt

必須

はい

レスポンスエンティティ

なし。

特別なエラーレスポンス

NoSuchObject

詳細

指定されたオブジェクトは存在しません。

コード

404 Not Found

ObjectRemovalFailed

詳細

オブジェクトを削除できません。

コード

409 Conflict

2.24. QUOTAS

管理操作 API を使用すると、ユーザーおよびユーザーが所有するバケットにクォータを設定できます。クォータには、バケットのオブジェクトの最大数と、メガバイト単位のストレージの最大サイズが含まれます。

クォータを表示するには、ユーザーに **users=read** ケイパビリティが必要です。クォータを設定、変更、または無効にするには、ユーザーに **users=write** ケイパビリティが必要です。

クォータの有効なパラメーターには以下が含まれます。

- **Bucket: bucket** オプションでは、ユーザーが所有するバケットのクォータを指定できます。
- **Maximum Objects: max-objects** 設定では、オブジェクトの最大数を指定できます。負の値を設定すると、この設定が無効になります。

- **Maximum Size: max-size** オプションでは、バイトの最大数のクォータを指定できます。負の値を設定すると、この設定が無効になります。
- **Quota Scope: quota-scope** オプションは、クォータのスコープを設定します。オプションは **bucket** と **user** です。

2.25. ユーザークォータの取得

クォータを取得するには、**read** パーミッションを持つ **users** ケイパビリティが設定されている必要があります。

構文

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

2.26. ユーザークォータの設定

クォータを設定するには、ユーザーに **write** パーミッションを持つ **users** ケイパビリティを設定する必要があります。

構文

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

コンテンツには、対応する読み取り操作でエンコードされているクォータ設定の JSON 表現が含まれている必要があります。

2.27. バケットクォータの取得

既存のバケットのサブセットに関する情報を取得します。**uid** が **bucket** なしで指定されると、そのユーザーに属するすべてのバケットが返されます。**bucket** のみが指定されている場合は、その特定のバケットの情報を取得します。

機能

```
`buckets=read`
```

構文

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

リクエストパラメーター

bucket

詳細

情報を返すバケット。

型

String

例

foo_bucket

必須

いいえ

uid

詳細

バケット情報を取得するユーザー。

型

String

例

foo_user

必須

いいえ

stats

詳細

バケットの統計を返します。

型

Boolean

例

True [False]

必須

いいえ

レスポンスエンティティ

stats

詳細

バケットごとの情報

型

Container

親

該当なし

buckets

詳細

1つ以上のバケットコンテナの一覧が含まれます。

型

Container

親

該当なし

bucket

詳細

単一バケット情報用のコンテナ。

型

Container

親

buckets

name**詳細**

バケットの名前。

型

String

親

bucket

pool**詳細**

バケットが保存されているプール。

型

String

親

bucket

id**詳細**

一意のバケット ID。

型

String

親

bucket

marker**詳細**

内部バケットタグ。

型

String

親

bucket

owner**詳細**

バケット所有者のユーザー ID。

型

String

親

bucket

使用状況

詳細

ストレージの使用情報。

型

Container

親

bucket

index

詳細

バケットインデックスのステータス。

型

String

親

bucket

成功すると、要求はバケット情報と共にバケットコンテナを返します。

特別なエラーレスポンス

IndexRepairFailed

詳細

バケットインデックスの修復に失敗しました。

コード

409 Conflict

2.28. バケットクォータの設定

クォータを設定するには、ユーザーに **write** パーミッションを持つ **users** ケイパビリティを設定する必要があります。

構文

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

コンテンツには、対応する読み取り操作でエンコードされているクォータ設定の JSON 表現が含まれている必要があります。

2.29. 使用方法情報の取得

帯域幅の使用情報の要求。

機能

```
`usage=read`
```

構文

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

リクエストパラメーター

uid

詳細

情報が要求されるユーザー。

型

String

必須

はい

start

詳細

データリクエストが開始した時点の日付 (任意で時刻)。(例: **2012-09-25 16:00:00**)。

型

String

必須

いいえ

end

詳細

データリクエストが終了した時点の日付 (任意で時刻)。(例: **2012-09-25 16:00:00**)。

型

String

必須

いいえ

show-entries

詳細

データエントリーを返すかどうかを指定します。

型

Boolean

必須

いいえ

show-summary

詳細

データエントリーを返すかどうかを指定します。

型

Boolean

必須

いいえ

レスポンスエンティティ**使用状況****詳細**

使用方法に関する情報用のコンテナ。

型

Container

エントリー**詳細**

使用方法エントリー情報のコンテナ。

型

Container

user**詳細**

ユーザーデータ情報のコンテナ

型

Container

owner**詳細**

バケットを所有するユーザーの名前。

型

String

bucket**詳細**

バケット名。

型

String

time**詳細**

データが指定されている時間の下限 (最初の関連する時間の開始に丸められます)。

型

String

epoch

詳細

1/1/1970 からの経過時間 (秒単位)。

型

String

categories

詳細

統計情報カテゴリーのコンテナ。

型

Container

entry

詳細

stats エントリーのコンテナ。

型

Container

category

詳細

統計が提供される要求カテゴリーの名前。

型

String

bytes_sent

詳細

Ceph Object Gateway によって送信されるバイト数。

型

Integer

bytes_received

詳細

Ceph Object Gateway が受け取るバイト数。

型

Integer

ops

詳細

演算の数。

型

Integer

successful_ops**詳細**

成功した操作の数。

型

Integer

summary**詳細**

成功した操作の数。

型

Container

total**詳細**

統計情報の概要集計合計のコンテナ。

型

Container

成功すると、レスポンスには要求された情報が含まれます。

2.30. 使用方法に関する情報を削除

使用方法に関する情報を削除します。日付を指定しないと、すべての使用情報が削除されます。

機能

```
`usage=write`
```

構文

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

リクエストパラメーター**uid****詳細**

情報が要求されるユーザー。

型

String

例

foo_user

必須

はい

start**詳細**

データリクエストが開始した時点の日付 (任意で時刻)。 (例: **2012-09-25 16:00:00**)。

型

String

例

2012-09-25 16:00:00

必須

いいえ

end**詳細**

データリクエストが終了した時点の日付 (任意で時刻)。 (例: **2012-09-25 16:00:00**)。

型

String

例

2012-09-25 16:00:00

必須

いいえ

remove-all**詳細**

マルチユーザーデータの削除を確認するために **uid** が指定されていない場合に必須です。

型

Boolean

例

True [False]

必須

いいえ

2.31. 標準エラーレスポンス

以下のリストは、標準的なエラーレスポンスと説明の詳細を示しています。

AccessDenied**詳細**

アクセスが拒否されました。

コード

403 Forbidden

InternalError**詳細**

内部サーバーエラー。

コード

500 Internal Server Error

NoSuchUser**詳細**

ユーザーが存在しません。

コード

404 Not Found

NoSuchBucket**詳細**

バケットが存在しません。

コード

404 Not Found

NoSuchKey**詳細**

そのようなアクセスキーはありません。

コード

404 Not Found

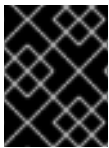
第3章 CEPH OBJECT GATEWAY および S3 API

開発者は、Amazon S3 データアクセスモデルと互換性のある RESTful アプリケーションプログラミングインターフェイス (API) を使用できます。Ceph Object Gateway を使用して、Red Hat Ceph Storage クラスタに保存されているバケットおよびオブジェクトを管理できます。

3.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。
- RESTful クライアント。

3.2. S3 の制限



重要

以下の制限事項を使用してください。お使いのハードウェアの選択には影響があるため、この要件を Red Hat アカウントチームと常に相談してください。

- Amazon S3 を使用する場合の最大オブジェクトサイズ:** 個別の Amazon S3 オブジェクトは、最小の 0B から最大 5TB のサイズに制限できます。1つの **PUT** でアップロードできる最大オブジェクトは 5 GB です。100MB を超えるオブジェクトの場合は、Multipart Upload ケイパビリティの使用を検討してください。
- Amazon S3 を使用する場合の最大メタデータサイズ:** オブジェクトに適用できるユーザーメタデータの合計サイズに定義された制限はありませんが、単一の HTTP リクエストは 16,000 バイトに制限されます。
- Red Hat Ceph Storage クラスタでは、S3 オブジェクトおよびメタデータを保存するために生成するデータオーバーヘッドのデータ量:** 推定時間は 200-300 バイトとオブジェクト名の長さです。バージョン管理されたオブジェクトは、バージョン数に比例する領域を追加で使用します。また、マルチパートアップロードなどのトランザクション更新中に一時的なオーバーヘッドが発生しますが、これらのオーバーヘッドはガベージコレクション中にリカバリーされます。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [サポートされないヘッダーフィールド](#) を参照してください。

3.3. S3 API を使用した CEPH OBJECT GATEWAY へのアクセス

開発者は、Amazon S3 API の使用を開始する前に、Ceph Object Gateway および Secure Token Service (STS) へのアクセスを設定する必要があります。

3.3.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。
- 実行中の Ceph Object Gateway。
- RESTful クライアント。

3.3.2. S3 認証

Ceph Object Gateway への要求は、認証または認証解除のいずれかになります。Ceph Object Gateway は、認証されていないリクエストが匿名ユーザーによって送信されることを前提としています。Ceph Object Gateway は、固定 ACL をサポートしています。

ほとんどのユースケースでは、クライアントは、Java や Python Boto 用の Amazon SDK の **AmazonS3Client** などの既存のオープンソースライブラリーを使用します。オープンソースライブラリーでは、アクセスキーおよびシークレットキーを渡すだけで、ライブラリーはユーザーの要求ヘッダーおよび認証署名をビルドします。ただし、リクエストを作成して署名することもできます。

リクエストの認証には、アクセスキーとベース 64 でエンコードされたハッシュベースのメッセージ認証コード (HMAC) が Ceph Object Gateway サーバーに送信される前に要求に追加する必要があります。Ceph Object Gateway は S3 互換の認証を使用します。

例

```
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
```

上記の例では、**ACCESS_KEY** をアクセスキー ID の値に置き換え、その後にコロン (:) を追加します。**HASH_OF_HEADER_AND_SECRET** を、正規化されたヘッダー文字列のハッシュとアクセスキー ID に対応するシークレットに置き換えます。

ヘッダー文字列およびシークレットのハッシュの生成

ヘッダー文字列およびシークレットのハッシュを生成するには、以下を実行します。

1. ヘッダー文字列の値を取得します。
2. 要求ヘッダー文字列を正規形式に正規化します。
3. SHA-1 ハッシュアルゴリズムを使用して HMAC を生成します。
4. **hmac** の結果を base-64 としてエンコードします。

ヘッダーを正規化

ヘッダーを正規の形式に正規化するには、以下を行います。

1. すべての **content-** ヘッダーを取得します。
2. **content-type** および **content-md5** 以外の **content-** ヘッダーをすべて削除します。
3. **content-** ヘッダー名が小文字であることを確認します。
4. **content-** ヘッダーの辞書式で並べ替えます。
5. **Date** ヘッダー AND があることを確認します。指定した日付が、オフセットではなく GMT を使用していることを確認してください。

6. **x-amz-** で始まるヘッダーをすべて取得します。
7. **x-amz-** ヘッダーがすべて小文字であることを確認します。
8. **x-amz-** ヘッダーの辞書式で並べ替えます。
9. 同じフィールド名の複数のインスタンスを単一のフィールドに組み合わせ、フィールド値をコンマで区切ります。
10. ヘッダー値の空白文字および改行文字を、単一スペースに置き換えます。
11. コロンの前後に空白を削除します。
12. 各ヘッダーの後に新しい行を追加します。
13. ヘッダーを要求ヘッダーにマージします。

HASH_OF_HEADER_AND_SECRET を、base-64 でエンコードされた HMAC 文字列に置き換えます。

関連情報

- 詳細は、Amazon Simple Storage Service ドキュメントの [Signing and Authenticating REST Requests](#) セクションを参照してください。

3.3.3. S3 サーバー側の暗号化

Ceph Object Gateway は、S3 アプリケーションプログラムインターフェイス (API) のアップロードされたオブジェクトのサーバー側の暗号化をサポートします。サーバー側の暗号化とは、S3 クライアントが暗号化されていない形式で HTTP 経由でデータを送信し、Ceph Object Gateway はそのデータを暗号化した形式で Red Hat Ceph Storage に保存することを意味します。



注記

- Red Hat は、Static Large Object (SLO) または Dynamic Large Object (DLO) の S3 オブジェクト暗号化をサポートしません。
- 現在、S3 Server-Side Encryption (SSE) モードのいずれも **CopyObject** のサポートを実装していません。これは、現在開発中です。[BZ#2149758]



重要

暗号化を使用するには、クライアントリクエストは、SSL 接続上でリクエストを送信する **必要があります**。Red Hat は、Ceph Object Gateway が SSL を使用しない限り、クライアントからの S3 暗号化をサポートしません。ただし、テストの目的で、管理者は **ceph config set client.rgw** コマンドを使用して、**rgw_crypt_require_ssl** 設定を **false** に設定してから、Ceph Object Gateway インスタンスを再起動することで、テスト中に SSL を無効にできます。

実稼働環境では、SSL 経由で暗号化された要求を送信できない場合があります。このような場合は、サーバー側の暗号化で HTTP を使用して要求を送信します。

サーバー側の暗号化で HTTP を設定する方法は、以下の **関連情報** セクションを参照してください。

暗号化キーの管理には、以下の2つのオプションがあります。

お客様提供のキー

お客様が提供する鍵を使用する場合、S3 クライアントは暗号鍵を各リクエストと共に渡して、暗号化されたデータの読み取りまたは書き込みを行います。これらのキーを管理するのは、お客様の責任です。各オブジェクトの暗号化に使用する Ceph Object Gateway の鍵を覚えておく必要があります。

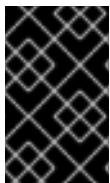
Ceph Object Gateway は、Amazon SSE-C 仕様に従って、S3 API で顧客提供のキー動作を実装します。

お客様がキー管理を処理し、S3 クライアントはキーを Ceph Object Gateway に渡すため、Ceph Object Gateway ではこの暗号化モードをサポートするための特別な設定は必要ありません。

キー管理サービス

キー管理サービスを使用する場合、セキュアなキー管理サービスはキーを格納し、Ceph Object Gateway はデータの暗号化または復号の要求に対応するためにキーをオンデマンドで取得します。

Ceph Object Gateway は、Amazon SSE-KMS 仕様に従って S3 API にキー管理サービスの動作を実装します。



重要

現時点で、テスト済み鍵管理の実装は HashiCorp Vault および OpenStack Barbican です。ただし、OpenStack Barbican はテクノロジープレビューであるため、実稼働システムでの使用はサポートされません。

関連情報

- [Amazon SSE-C](#)
- [Amazon SSE-KMS](#)
- [サーバー側の暗号化の設定](#)
- [HashiCorp Vault](#)

3.3.4. S3 アクセス制御リスト

Ceph Object Gateway は S3 と互換性のあるアクセス制御リスト (ACL) の機能をサポートします。ACL は、ユーザーがバケットまたはオブジェクトで実行できる操作を指定するアクセス権限の一覧です。それぞれの付与は、バケットに適用するか、またはオブジェクトに適用される場合の異なる意味を持ちます。

表3.1 ユーザー操作

パーミッション	バケット	Object
READ	パーミッションを得たユーザーは、バケットのオブジェクトを一覧表示できます。	パーミッションを得たユーザーは、オブジェクトを読み取りできます。
WRITE	パーミッションを得たユーザーは、バケットのオブジェクトを書き込みまたは削除できます。	該当なし

パーミッション	バケット	Object
READ_ACP	パーミッションを得たユーザーは、バケット ACL を読み取ることができます。	パーミッションを得たユーザーは、オブジェクト ACL を読み取ることができます。
WRITE_ACP	パーミッションを得たユーザーは、バケット ACL を書き込めます。	パーミッションを得たユーザーは、オブジェクト ACL に書き込めます。
FULL_CONTROL	Grantee にはバケットのオブジェクトに対する完全なパーミッションがあります。	パーミッションを得たユーザーは、オブジェクト ACL に読み取りまたは書き込みできます。

3.3.5. S3 を使用した Ceph Object Gateway へのアクセスの準備

ゲートウェイサーバーにアクセスする前に、Ceph Object Gateway ノードの前提条件に従う必要があります。

前提条件

- Ceph Object Gateway ソフトウェアのインストール。
- Ceph Object Gateway ノードへのルートレベルのアクセス

手順

1. **root** で、ファイアウォールのポート **8080** を開きます。

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

2. [オブジェクトゲートウェイ設定および管理ガイド](#) で説明されているように、ゲートウェイに使用する DNS サーバーにワイルドカードを追加します。
ローカル DNS キャッシュ用のゲートウェイノードを設定することもできます。これを実行するには、以下の手順を実行します。

- a. **root** で **dnsmasq** をインストールおよび設定します。

```
[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
```

IP_OF_GATEWAY_NODE および **FQDN_OF_GATEWAY_NODE** は、ゲートウェイノードの IP アドレスと FQDN に置き換えます。

- b. **root** で NetworkManager を停止します。

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

- c. **root** として、ゲートウェイサーバーの IP を名前空間として設定します。

```
[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
```

IP_OF_GATEWAY_NODE および **FQDN_OF_GATEWAY_NODE** は、ゲートウェイノードの IP アドレスと FQDN に置き換えます。

- d. サブドメイン要求を確認します。

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

FQDN_OF_GATEWAY_NODE は、ゲートウェイノードの FQDN に置き換えます。



警告

ローカルの DNS キャッシュ用にゲートウェイサーバーを設定することは、テスト目的のみを目的としています。これを行った後は、外部ネットワークにはアクセスできなくなります。Red Hat Ceph Storage クラスタおよびゲートウェイノードに適切な DNS サーバーを使用することを強く推奨します。

3. [オブジェクトゲートウェイの設定および管理ガイド](#) に説明されているように、**S3** アクセスに **radosgw** ユーザーを作成し、生成した **access_key** および **secret_key** をコピーします。**S3** アクセス、およびそれ以降のバケット管理タスクには、これらのキーが必要です。

3.3.6. Ruby AWS S3 を使用した Ceph Object Gateway へのアクセス

Ruby プログラミング言語は、**S3** アクセスに **aws-s3** gem と共に使用できます。**Ruby AWS::S3** で Ceph Object Gateway サーバーにアクセスするために使用されるノードで以下の手順を実行します。

前提条件

- Ceph Object Gateway へのユーザーレベルのアクセス。
- Ceph Object Gateway にアクセスするノードへのルートレベルのアクセス。
- インターネットアクセス。

手順

1. **ruby** パッケージをインストールします。

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



注記

上記のコマンドは **ruby** と、**rubygems**、**ruby-libs** などの基本的な依存関係をインストールします。コマンドによってすべての依存関係がインストールされていない場合は、個別にインストールします。

2. Ruby パッケージ **aws-s3** をインストールします。

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

3. プロジェクトディレクトリーを作成します。

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

4. コネクションファイルを作成します。

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

5. **conn.rb** ファイルに以下のコンテンツを貼り付けます。

構文

```
#!/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'
)
```

FQDN_OF_GATEWAY_NODE は、Ceph Object Gateway ノードの FQDN に置き換えます。[Red Hat Ceph Storage オブジェクトゲートウェイの設定および管理ガイド](#) に記載されたとおり、**MY_ACCESS_KEY** および **MY_SECRET_KEY** は、S3 アクセスの **radosgw** が作成された場合に生成された **access_key** および **secret_key** に置き換えます。

例

```
#!/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'
)

```

ファイルを保存して、エディターを終了します。

6. ファイルを実行可能にします。

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

7. コマンドを実行します。

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

ファイルに正しく値を指定した場合は、コマンドの出力は **0** になります。

8. バケットを作成するための新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

```

#!/usr/bin/env ruby

load 'conn.rb'

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

```

ファイルを保存して、エディターを終了します。

9. ファイルを実行可能にします。

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

10. コマンドを実行します。

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

コマンドの出力が **true** の場合は、バケット **my-new-bucket1** が正常に作成されたことを意味します。

11. 所有されるバケットを一覧表示するために新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

```

#!/usr/bin/env ruby

load 'conn.rb'

```

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

ファイルを保存して、エディターを終了します。

12. ファイルを実行可能にします。

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

13. コマンドを実行します。

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

出力は以下のようになります。

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

14. オブジェクトを作成するための新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

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

load 'conn.rb'

AWS::S3::S3Object.store(
  'hello.txt',
  'Hello World!',
  'my-new-bucket1',
  :content_type => 'text/plain'
)
```

ファイルを保存して、エディターを終了します。

15. ファイルを実行可能にします。

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

16. コマンドを実行します。

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

これで、文字列 **Hello World!** で **hello.txt** が作成されます。

17. バケットのコンテンツを一覧表示するための新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

```
#!/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
```

ファイルを保存して、エディターを終了します。

18. ファイルを実行可能にします。

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

19. コマンドを実行します。

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

出力は以下のようになります。

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

20. 空のバケットを削除するために新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

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

load 'conn.rb'

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

ファイルを保存して、エディターを終了します。

21. ファイルを実行可能にします。

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

22. コマンドを実行します。

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

バケットが正常に削除されると、コマンドは **0** を出力として返します。



注記

create_bucket.rb ファイルを編集し、空のバケットを作成します (例: **my-new-bucket4**、**my-new-bucket5**)。次に、空のバケットの削除を試みる前に、上記の **del_empty_bucket.rb** ファイルを適宜編集します。

23. 空でないバケットを削除する新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

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

load 'conn.rb'

AWS::S3::Bucket.delete('my-new-bucket1', :force => true)
```

ファイルを保存して、エディターを終了します。

24. ファイルを実行可能にします。

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

25. コマンドを実行します。

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

バケットが正常に削除されると、コマンドは **0** を出力として返します。

26. オブジェクトを削除する新しいファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

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

load 'conn.rb'

AWS::S3::S3Object.delete('hello.txt', 'my-new-bucket1')
```

ファイルを保存して、エディターを終了します。

27. ファイルを実行可能にします。

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

28. コマンドを実行します。

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

これにより、オブジェクト **hello.txt** が削除されます。

3.3.7. Ruby AWS SDK を使用した Ceph Object Gateway へのアクセス

Ruby プログラミング言語は、**S3** アクセスに **aws-sdk** gem と共に使用できます。**Ruby AWS::SDK** を使用して Ceph Object Gateway サーバーにアクセスするために使用されるノードで以下の手順を実行します。

前提条件

- Ceph Object Gateway へのユーザーレベルのアクセス。
- Ceph Object Gateway にアクセスするノードへのルートレベルのアクセス。
- インターネットアクセス。

手順

1. **ruby** パッケージをインストールします。

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



注記

上記のコマンドは **ruby** と、**rubygems**、**ruby-libs** などの基本的な依存関係をインストールします。コマンドによってすべての依存関係がインストールされていない場合は、個別にインストールします。

2. Ruby パッケージ **aws-sdk** をインストールします。

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

3. プロジェクトディレクトリーを作成します。

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

4. コネクションファイルを作成します。

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

5. **conn.rb** ファイルに以下のコンテンツを貼り付けます。

構文

```
#!/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'
)
```

FQDN_OF_GATEWAY_NODE は、Ceph Object Gateway ノードの FQDN に置き換えます。[Red Hat Ceph Storage オブジェクトゲートウェイの設定および管理ガイド](#) に記載されたとおり、**MY_ACCESS_KEY** および **MY_SECRET_KEY** は、S3 アクセスの **radosgw** が作成さ

れた場合に生成された **access_key** および **secret_key** に置き換えます。

例

```
#!/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'
)
```

ファイルを保存して、エディターを終了します。

6. ファイルを実行可能にします。

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

7. コマンドを実行します。

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

ファイルに正しく値を指定した場合は、コマンドの出力は **0** になります。

8. バケットを作成するための新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

構文

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

load 'conn.rb'

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

ファイルを保存して、エディターを終了します。

9. ファイルを実行可能にします。

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

10. コマンドを実行します。

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

コマンドの出力が **true** の場合は、バケット **my-new-bucket2** が正常に作成されていることを意味します。

11. 所有されるバケットを一覧表示するために新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

```
#!/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
```

ファイルを保存して、エディターを終了します。

12. ファイルを実行可能にします。

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

13. コマンドを実行します。

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

出力は以下のようになります。

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

14. オブジェクトを作成するための新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

```
#!/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'
)
```

ファイルを保存して、エディターを終了します。

15. ファイルを実行可能にします。

■

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

16. コマンドを実行します。

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

これで、文字列 **Hello World!** で **hello.txt** が作成されます。

17. バケットのコンテンツを一覧表示するための新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

```
#!/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
```

ファイルを保存して、エディターを終了します。

18. ファイルを実行可能にします。

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

19. コマンドを実行します。

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

出力は以下のようになります。

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

20. 空のバケットを削除するために新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

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

load 'conn.rb'

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

ファイルを保存して、エディターを終了します。

21. ファイルを実行可能にします。

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

22. コマンドを実行します。

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

バケットが正常に削除されると、コマンドは **0** を出力として返します。



注記

create_bucket.rb ファイルを編集し、空のバケットを作成します (例: **my-new-bucket6**、**my-new-bucket7**)。次に、空のバケットの削除を試みる前に、上記の **del_empty_bucket.rb** ファイルを適宜編集します。

23. 空でないバケットを削除する新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

```
#!/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')
```

ファイルを保存して、エディターを終了します。

24. ファイルを実行可能にします。

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

25. コマンドを実行します。

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

バケットが正常に削除されると、コマンドは **0** を出力として返します。

26. オブジェクトを削除する新しいファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

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

load 'conn.rb'

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

ファイルを保存して、エディターを終了します。

27. ファイルを実行可能にします。

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

28. コマンドを実行します。

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

これにより、オブジェクト **hello.txt** が削除されます。

3.3.8. PHP を使用した Ceph Object Gateway へのアクセス

S3 アクセスには PHP スクリプトを使用できます。この手順では、バケットやオブジェクトの削除など、さまざまなタスクを実行する PHP スクリプトの例を提供します。



重要

以下は、**php v5.4.16** および **aws-sdk v2.8.24** に対してテストされています。**php >= 5.5+** が必要なため、**php** に **aws-sdk** の最新バージョンを使用 **しません**。**php 5.5** は、**RHEL 7** のデフォルトリポジトリでは利用できません。**php 5.5** を使用する場合は、**epel** およびその他のサードパーティーのリポジトリを有効にする必要があります。また、**php 5.5** および最新バージョンの **aws-sdk** の設定オプションも異なります。

前提条件

- 開発ワークステーションへのルートレベルのアクセス。
- インターネットアクセス。

手順

1. **php** パッケージをインストールします。

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

2. PHP 用に **aws-sdk** の zip アーカイブを [ダウンロード](#) し、展開します。

3. プロジェクトディレクトリーを作成します。

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

4. 展開した **aws** ディレクトリーをプロジェクトのディレクトリーにコピーします。以下に例を示します。

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

5. コネクションファイルを作成します。

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

6. **conn.php** ファイルに以下のコンテンツを貼り付けます。

構文

```
<?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
));
?>
```

FQDN_OF_GATEWAY_NODE は、ゲートウェイノードの FQDN に置き換えます。 [Red Hat Ceph Storage オブジェクトゲートウェイの設定および管理ガイド](#) に記載されたとおり、**MY_ACCESS_KEY** および **MY_SECRET_KEY** は、S3 アクセスの **radosgw** が作成された場合に生成された **access_key** および **secret_key** に置き換えます。**PATH_TO_AWS** は、**php** プロジェクトディレクトリーにコピーした、展開した **aws** ディレクトリーへの絶対パスに置き換えます。

ファイルを保存して、エディターを終了します。

7. コマンドを実行します。

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

ファイルに正しく値を指定した場合は、コマンドの出力は **0** になります。

8. バケットを作成するための新規ファイルを作成します。

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

新しいファイルに以下の内容を貼り付けます。

構文

```
<?php

include 'conn.php';

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

?>
```

ファイルを保存して、エディターを終了します。

9. コマンドを実行します。

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

10. 所有されるバケットを一覧表示するために新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

構文

```
<?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";
}

?>
```

ファイルを保存して、エディターを終了します。

11. コマンドを実行します。

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

出力は以下のようになります。

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

12. まず **hello.txt** という名前のソースファイルを作成するオブジェクトを作成します。

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

13. 新しい php ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

構文

```
<?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);

?>
```

ファイルを保存して、エディターを終了します。

14. コマンドを実行します。

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

これにより、バケット **my-new-bucket3** でオブジェクト **hello.txt** が作成されます。

15. バケットのコンテンツを一覧表示するための新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

構文

```
<?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";
}
?>
```

ファイルを保存して、エディターを終了します。

16. コマンドを実行します。

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

出力は以下のようになります。

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

17. 空のバケットを削除するために新規ファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

構文

```
<?php
```

```
include 'conn.php';

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

ファイルを保存して、エディターを終了します。

18. コマンドを実行します。

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

バケットが正常に削除されると、コマンドは **0** を出力として返します。



注記

create_bucket.php ファイルを編集し、空のバケットを作成します (例: **my-new-bucket4**、**my-new-bucket5**)。次に、空のバケットの削除を試みる前に、上記の **del_empty_bucket.php** ファイルを適宜編集します。



重要

空でないバケットの削除は、現在 PHP 2 以降のバージョンの **aws-sdk** ではサポートされていません。

19. オブジェクトを削除する新しいファイルを作成します。

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

以下のコンテンツをファイルに貼り付けます。

構文

```
<?php

include 'conn.php';

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

ファイルを保存して、エディターを終了します。

20. コマンドを実行します。

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

これにより、オブジェクト **hello.txt** が削除されます。

3.3.9. セキュアなトークンサービス

Amazon Web Services の Secure Token Service (STS) は、ユーザーを認証するための一時セキュリティ認証情報のセットを返します。Ceph Object Gateway は STS アプリケーションプログラミングインターフェイス (API) のサブセットを実装し、ID およびアクセス管理 (IAM) の一時的な認証情報を提供します。これらの一時的な認証情報を使用して、Ceph Object Gateway の STS エンジンを使用して S3 呼び出しを認証します。IAM ポリシーを使用すると、STS API に渡されるパラメーターである一時認証情報をさらに制限できます。

関連情報

- Amazon Web Services Secure Token Service の [Welcome ページ](#)。
- STS Lite および Keystone の詳細は、Red Hat Ceph Storage 開発者ガイドの [STS Lite および Keystone の設定および使用](#) セクションを参照してください。
- STS Lite および Keystone の制限の詳細は、Red Hat Ceph Storage 開発者ガイドの [Keystone のある STS Lite を使用する際の制限の回避](#) を参照してください。

3.3.9.1. Secure Token Service アプリケーションのプログラミングインターフェイス

Ceph Object Gateway は、以下の Secure Token Service (STS) アプリケーションプログラミングインターフェイス (API) を実装します。

AssumeRole

この API は、アカウント間のアクセスのための一時的な認証情報のセットを返します。これらの一時的な認証情報により、**Role** と、**AssumeRole** API で割り当てられるポリシーの両方に割り当てられるパーミッションポリシーを使用することができます。**RoleArn** および **RoleSessionName** リクエストパラメーターは必須ですが、他の要求パラメーターは任意です。

RoleArn

詳細

長さが 20 ~ 2048 文字の Amazon Resource Name (ARN) について想定するロール。

型

String

必須

はい

RoleSessionName

詳細

仮定するロールセッション名を特定します。ロールセッション名は、異なるプリンシパルや別の理由がロールを想定する場合にセッションを一意に識別できます。このパラメーターの値は、2 文字から 64 文字までです。=、,、.、@、および - 文字は使用できますが、スペースは使用できません。

型

String

必須

はい

ポリシー

詳細

この API は、Amazon Web Services の Secure Token Service (STS) の一時的な認証情報を使用して、S3 のオブジェクトをアップロードおよびダウンロードします。

インラインセッションで使用する JSON 形式の ID およびアクセス管理ポリシー (IAM)。このパラメーターの値は1文字から 2048 文字までです。

型

String

必須

いいえ

DurationSeconds**詳細**

セッションの期間 (秒単位)。最小値は **900** 秒で、最大値は **43200** 秒です。デフォルト値は **3600** 秒です。

型

整数

必須

いいえ

ExternalId**詳細**

別のアカウントのロールを想定する場合には、利用可能な場合は一意の外部識別子を指定します。このパラメーターの値は、2 文字から 1224 文字までになります。

型

String

必須

いいえ

SerialNumber**詳細**

関連付けられたマルチファクター認証 (MFA) デバイスからのユーザーの識別番号。パラメーターの値は、9 文字から 256 文字までのハードウェアデバイスまたは仮想デバイスのシリアル番号になります。

型

String

必須

いいえ

TokenCode**説明**

信頼ポリシーに MFA が必要な場合は、マルチファクター認証 (MFA) デバイスから生成された値。MFA デバイスが必要で、このパラメーターの値が空または期限切れの場合には、**AssumeRole** の呼び出しは access denied エラーメッセージを返します。このパラメーターの値には、固定長は 6 文字です。

型

String

必須

いいえ

AssumeRoleWithWebIdentity

この API は、OpenID Connect や OAuth 2.0 アイデンティティプロバイダーなどのアプリケーションによって認証されたユーザーの一時認証情報のセットを返します。**RoleArn** および **RoleSessionName** リクエストパラメーターは必須ですが、他の要求パラメーターは任意です。

RoleArn

詳細

長さが 20 ~ 2048 文字の Amazon Resource Name (ARN) について想定するロール。

型

String

必須

はい

RoleSessionName

詳細

仮定するロールセッション名を特定します。ロールセッション名は、異なるプリンシパルや別の理由がロールを想定する場合にセッションを一意に識別できます。このパラメーターの値は、2 文字から 64 文字までです。=、,、.、@、および - 文字は使用できますが、スペースは使用できません。

型

String

必須

はい

ポリシー

詳細

インラインセッションで使用する JSON 形式の ID およびアクセス管理ポリシー (IAM)。このパラメーターの値は 1 文字から 2048 文字までです。

型

String

必須

いいえ

DurationSeconds

詳細

セッションの期間 (秒単位)。最小値は **900** 秒で、最大値は **43200** 秒です。デフォルト値は **3600** 秒です。

型

整数

必須

いいえ

ProviderId

詳細

この API は、OpenID Connect や OAuth 2.0 アイデンティティプロバイダーなどのアプリケーションによって認証されたユーザーの一時認証情報のセットを返します。

アイデンティティプロバイダーからのドメイン名の完全修飾ホストコンポーネント。このパラメーターの値は、長さが 4 ~ 2048 文字の OAuth 2.0 アクセストークンでのみ有効です。

型

String

必須

いいえ

WebIdentityToken

詳細

アイデンティティプロバイダーから提供される OpenID Connect アイデンティティトークンまたは OAuth 2.0 アクセストークン。このパラメーターの値は、4 文字から 2048 文字までです。

型

String

必須

いいえ

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Secure Token Service API の使用例](#) セクションを参照してください。
- Amazon Web Services Security Token Service ([AssumeRole](#) アクション)
- Amazon Web Services Security Token Service ([AssumeRoleWithWebIdentity](#) アクション)

3.3.9.2. セキュアなトークンサービスの設定

rgw_sts_key および **rgw_s3_auth_use_sts** オプションを設定して、CephObjectGateway で使用する Ceph Object Gateway (STS) を設定します。



注記

S3 と STS API は同じ名前空間に共存し、いずれも Ceph Object Gateway の同じエンドポイントからアクセスできます。

前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。
- 実行中の Ceph Object Gateway。
- Ceph Manager ノードへのルートレベルのアクセス。

手順

1. Ceph Object Gateway クライアントに以下の設定オプションを設定します。

構文

```
ceph config set RGW_CLIENT_NAME rgw_sts_key STS_KEY
ceph config set RGW_CLIENT_NAME rgw_s3_auth_use_sts true
```

rgw_sts_key は、セッショントークンを暗号化または復号化するための STS キーであり、正確に 16 進数の 16 文字です。

例

```
[root@mgr ~]# ceph config set client.rgw rgw_sts_key abcdefghijklmnop
[root@mgr ~]# ceph config set client.rgw rgw_s3_auth_use_sts true
```

関連情報

- STS API に関する詳細は、Red Hat Ceph Storage 開発者ガイドの [Secure Token Service アプリケーションのプログラミングインターフェイス](#) セクションを参照してください。
- Ceph 設定データベースの使用の詳細については、Red Hat Ceph Storage 設定ガイドの [Ceph 設定の基本](#) の章を参照してください。

3.3.9.3. OpenID Connect プロバイダー用のユーザーの作成

Ceph Object Gateway と OpenID Connect Provider との間の信頼を確立するには、ユーザーエンティティとロール信頼ポリシーを作成します。

前提条件

- Ceph Object Gateway ノードへのユーザーレベルのアクセス。

手順

1. 新しい Ceph ユーザーを作成します。

構文

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

例

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

2. Ceph ユーザー機能を設定します。

構文

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

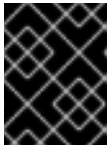
例

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

- Secure Token Service (STS) API を使用してロール信頼ポリシーに条件を追加します。

構文

```
{\"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\" } } } ] }
```



重要

上記の構文例の **app_id** は、着信トークンの **AUD_FIELD** フィールドと一致させる必要があります。

関連情報

- Amazon の Web サイトの [Obtaining the Root CA Thumbprint for an OpenID Connect Identity Provider](#) を参照してください。
- STS API に関する詳細は、Red Hat Ceph Storage 開発者ガイドの [Secure Token Service アプリケーションのプログラミングインターフェイス](#) セクションを参照してください。
- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Secure Token Service API の使用例](#) セクションを参照してください。

3.3.9.4. OpenID Connect プロバイダーのサムプリントの取得

OpenID Connect プロバイダー (IDP) の設定ドキュメントを取得するには、以下を実行します。

前提条件

- openssl** パッケージおよび **curl** パッケージのインストール。

手順

- IDP の URL から設定ドキュメントを取得します。

構文

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

例

```
[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. IDP 証明書を取得します。

構文

```
curl -k -v \
-X GET \
-H "Content-Type: application/x-www-form-urlencoded" \
"IDP_URL/CONTEXT/realms/REALM/protocol/openid-connect/certs" \
| jq .
```

例

```
[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. 直前のコマンドから x5c 応答の結果をコピーし、それを **certificate.crt** ファイルに貼り付けます。冒頭に **—BEGIN CERTIFICATE—**、末尾に **—END CERTIFICATE—** を含めます。
4. 証明書のサムプリントを取得します。

構文

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

例

```
[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. SHA1 フィンガープリントからコロンをすべて削除し、IAM 要求で IDP エンティティを作成するための入力として使用します。

関連情報

- Amazon の Web サイトの [Obtaining the Root CA Thumbprint for an OpenID Connect Identity Provider](#) を参照してください。
- STS API に関する詳細は、Red Hat Ceph Storage 開発者ガイドの [Secure Token Service アプリケーションのプログラミングインターフェイス](#) セクションを参照してください。
- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Secure Token Service API の使用例](#) セクションを参照してください。

3.3.9.5. Keystone での STS Lite の設定および使用 (テクノロジープレビュー)

Amazon Secure Token Service (STS) と S3 API は、同じ名前空間に共存します。STS オプションは、Keystone オプションと組み合わせて設定できます。



注記

S3 と STS の API の両方に、Ceph Object Gateway の同じエンドポイントを使用してアクセスできます。

前提条件

- Red Hat Ceph Storage 5.0 以降
- 実行中の Ceph Object Gateway。
- Boto Python モジュールのバージョン 3 以降のインストール
- Ceph Manager ノードへのルートレベルのアクセス。
- OpenStack ノードへのユーザーレベルのアクセス。

手順

1. Ceph Object Gateway クライアントに以下の設定オプションを設定します。

構文

```
ceph config set RGW_CLIENT_NAME rgw_sts_key STS_KEY
ceph config set RGW_CLIENT_NAME rgw_s3_auth_use_sts true
```

rgw_sts_key は、セッショントークンを暗号化または復号化するための STS キーであり、正確に 16 進数の 16 文字です。

例

```
[root@mgr ~]# ceph config set client.rgw rgw_sts_key abcdefghijklmnop
[root@mgr ~]# ceph config set client.rgw rgw_s3_auth_use_sts true
```

2. OpenStack ノードで EC2 認証情報を生成します。

例

```
[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 |
+-----+-----+
```

3. 生成された認証情報を使用して、**GetSessionToken** API を使用して一時的なセキュリティー認証情報のセットを取得します。

例

```
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
)
```

4. 一時認証情報の取得は、S3 呼び出しの作成に使用できます。

例

```
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'],
    )
```

5. 新しい **S3Access** ロールを作成し、ポリシーを設定します。
 - a. 管理 CAPS でユーザーを割り当てます。

構文

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

例

```
[root@mgr ~]# radosgw-admin caps add --uid="gwadmin" --caps="roles="
```

- b. **S3Access** ロールを作成します。

構文

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

例

```
[root@mgr ~]# 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. **S3Access** ロールにパーミッションポリシーを割り当てます。

構文

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

例

```
[root@mgr ~]# 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. 別のユーザーが **gwadmin** ユーザーのロールを想定できるようになりました。たとえば、**gwuser** ユーザーは、**gwadmin** ユーザーのパーミッションを想定できます。
- e. 仮定ユーザーの **access_key** および **secret_key** の値を書き留めておきます。

例

```
[root@mgr ~]# radosgw-admin user info --uid=gwuser | grep -A1 access_key
```

6. **AssumeRole** API 呼び出しを使用し、仮定のユーザーから **access_key** および **secret_key** の値を提供します。

例

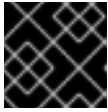
```
import boto3

access_key = 11BS02LGFB6AL6H1ADMW
secret_key = vzCEkuryfn060dfef4fgQPqFrncKElkh3ZcdOANY

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
)
```



重要

AssumeRole API には S3Access ロールが必要です。

関連情報

- Boto Python モジュールのインストールに関する詳細は、Red Hat Ceph Storage オブジェクトゲートウェイガイドの [S3 アクセスのテスト](#) セクションを参照してください。
- 詳細は、Red Hat Ceph Storage オブジェクトゲートウェイガイドの [ユーザーの作成](#) セクションを参照してください。

3.3.9.6. Keystone で STS Lite を使用するための制限の回避 (テクノロジープレビュー)

Keystone の制限は、Secure Token Service (STS) 要求をサポートしないことです。もう1つの制限は、ペイロードハッシュがリクエストに含まれていないことです。この2つの制限を回避するには、Boto 認証コードを変更する必要があります。

前提条件

- 稼働中の Red Hat Ceph Storage クラスター (バージョン 5.0 以降)。
- 実行中の Ceph Object Gateway。
- Boto Python モジュールのバージョン 3 以降のインストール

手順

1. Boto の **auth.py** ファイルを開いて編集します。
 - a. 以下の4つの行をコードブロックに追加します。

```
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 overridden in ``add_auth``
        # later for real requests.
        self._region_name = region_name
        if service_name == 'sts': ❶
            self._service_name = 's3' ❷
        else: ❸
            self._service_name = service_name ❹
```

- b. 以下の2つの行をコードブロックに追加します。

```

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 ❶
    else: ❷
        request.headers['X-Amz-Content-SHA256'] = self.payload(request)

```

関連情報

- Boto Python モジュールのインストールに関する詳細は、Red Hat Ceph Storage オブジェクトゲートウェイガイドの [S3 アクセスのテスト](#) セクションを参照してください。

3.4. S3 バケット操作

開発者は、Ceph Object Gateway 経由で Amazon S3 アプリケーションプログラミングインターフェイス (API) を使用してバケット操作を実行できます。

以下の表は、バケットの Amazon S3 機能操作と関数のサポートステータスを示しています。

表3.2 バケット操作

機能	状態	注記
バケットの一覧表示	サポート対象	
バケットの作成	サポート対象	固定 ACL のさまざまなセット。
Put Bucket Website	サポート対象	
Get Bucket Website	サポート対象	
Delete Bucket Website	サポート対象	
バケットライフサイクル	一部サポート対象	Expiration 、 NoncurrentVersionExpiration および AbortIncompleteMultipartUpload がサポートされます。
PUT バケットライフサイクル	一部サポート対象	Expiration 、 NoncurrentVersionExpiration および AbortIncompleteMultipartUpload がサポートされます。

機能	状態	注記
バケットライフサイクルの削除	サポート対象	
バケットオブジェクトの取得	サポート対象	
バケットの場所	サポート対象	
バケットバージョンの取得	サポート対象	
バケットバージョンの送信	サポート対象	
バケットの削除	サポート対象	
バケット ACL の取得	サポート対象	固定 ACL のさまざまなセット
バケット ACL の送信	サポート対象	固定 ACL のさまざまなセット
バケットに関する CORS 設定情報を取得	サポート対象	
バケットに対し CORS 設定を行う	サポート対象	
バケットの CORS 設定を削除	サポート対象	
バケットオブジェクトバージョンの一覧表示	サポート対象	
HEAD バケット	サポート対象	
バケットマルチパートアップロードの一覧表示	サポート対象	
バケットポリシー	一部サポート対象	
バケットリクエストの支払いの取得	サポート対象	
バケットリクエストの支払いを行う	サポート対象	
マルチテナントバケット操作	サポート対象	
Get PublicAccessBlock	サポート対象	
Put PublicAccessBlock	サポート対象	
Delete PublicAccessBlock	サポート対象	

3.4.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。
- RESTful クライアント。

3.4.2. S3 create bucket notifications

バケットレベルでバケット通知を作成します。通知設定には、Red Hat Ceph Storage Object Gateway S3 イベント (**ObjectCreated** および **ObjectRemoved**) があります。これらは公開され、バケット通知を送信する宛先である必要があります。バケット通知は S3 オペレーションです。

s3:objectCreate および **s3:objectRemove** イベントのバケット通知を作成するには、PUT を使用します。

例

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



重要

Red Hat は、**ObjectCreate** イベント (例: **put**、**post**、**multipartUpload**、および **copy**) をサポートします。また、Red Hat は、**object_delete**、**s3_multi_object_delete** などの **ObjectRemove** イベントをサポートしています。

リクエストエンティティ

NotificationConfiguration

詳細

TopicConfiguration エンティティのリスト。

型

Container

必須

はい

TopicConfiguration

説明

イベントエンティティの **Id**、**Topic**、および **list**。

型

Container

必須

はい

id**詳細**

通知の名前。

型

String

必須

はい

Topic**説明**

トピック Amazon リソース名 (ARN)

**注記**

トピックは事前に作成する必要があります。

型

String

必須

はい

Event**詳細**

サポートされるイベントの一覧。複数のイベントエンティティを使用できます。省略すると、すべてのイベントが処理されます。

型

String

必須

いいえ

フィルター**詳細**

S3Key、**S3Metadata**、および **S3Tags** エンティティ。

型

Container

必須

いいえ

S3Key**詳細**

オブジェクトキーに基づくフィルターリングの **FilterRule** エンティティの一覧。リストには最大で3つのエンティティを含めることができます。たとえば、**Name** は **Prefix**、**suffix**、または **regex** になります。リスト内のフィルタールールはすべて、フィルターが一致するために一致している必要があります。

型

Container

必須

いいえ

S3Metadata

詳細

オブジェクトメタデータに基づくフィルターリングの **FilterRule** エンティティの一覧。リスト内のフィルタールールはすべて、オブジェクトで定義されたメタデータと一致する必要があります。ただし、フィルターにリストされていない他のメタデータエントリーがある場合には、オブジェクトは一致するままになります。

型

Container

必須

いいえ

S3Tags

詳細

オブジェクトタグに基づいてフィルターリングする **FilterRule** エンティティの一覧。リスト内のフィルタールールはすべて、オブジェクトで定義されたタグと一致する必要があります。ただし、フィルターに他のタグがリストされていない場合、オブジェクトは引き続き一致します。

型

Container

必須

いいえ

S3Key.FilterRule

詳細

Name エンティティおよび **Value** エンティティです。Name は、**prefix**、**suffix**、または **regex** です。**Value** は、キー接頭辞、キー接尾辞、またはキーに一致する正規表現を保持します。

型

Container

必須

はい

S3Metadata.FilterRule

詳細

Name エンティティおよび **Value** エンティティです。Name は、メタデータ属性の名前です (例: **x-amz-meta-xxx**)。この値は、この属性で想定される値になります。

型

Container

必須

はい

S3Tags.FilterRule

詳細

Name エンティティーおよび **Value** エンティティーです。Name はタグキーで、値はタグの値です。

型

Container

必須

はい

HTTP レスポンス**400**

ステータスコード

MalformedXML

詳細

XML は適していません。

400

ステータスコード

InvalidArgument

詳細

ID がないか、トピック ARN がないか無効であるか、イベントが無効です。

404

ステータスコード

NoSuchBucket

詳細

バケットが存在しません。

404

ステータスコード

NoSuchKey

詳細

トピックが存在しません。

3.4.3. S3 get bucket notifications

特定の通知を取得するか、またはバケットに設定されたすべての通知を一覧表示します。

構文

Get **/BUCKET?notification=NOTIFICATION_ID** HTTP/1.1

Host: cname.domain.com

Date: date

Authorization: AWS **ACCESS_KEY:HASH_OF_HEADER_AND_SECRET**

例

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

レスポンスの例

```
<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>
```



注記

notification サブリソースはバケット通知設定または空の **NotificationConfiguration** 要素を返します。呼び出し元はバケットの所有者である必要があります。

リクエストエンティティ

notification-id

詳細

通知の名前。ID が指定されていない場合は、すべての通知が一覧表示されます。

型

String

NotificationConfiguration

詳細

TopicConfiguration エンティティのリスト。

型

Container

必須

はい

TopicConfiguration

説明

イベントエンティティの **Id**、**Topic**、および **list**。

型

Container

必須

はい

id

詳細

通知の名前。

型

String

必須

はい

Topic

説明

トピック Amazon リソース名 (ARN)



注記

トピックは事前に作成する必要があります。

型

String

必須

はい

Event

詳細

処理されたイベント。複数のイベントエンティティが存在する可能性があります。

型

String

必須

はい

フィルター

詳細

指定の設定のフィルター。

型

Container

必須

いいえ

HTTP レスポンス**404**

ステータスコード

NoSuchBucket

詳細

バケットが存在しません。

404

ステータスコード

NoSuchKey

詳細

通知は、提供された場合に存在しません。

3.4.4. S3 delete bucket notifications

バケットから特定の通知またはすべての通知を削除します。

**注記**

通知の削除は、S3 通知 API の拡張機能です。バケットで定義された通知は、バケットの削除時に削除されます。不明な通知 (例: **double delete**) を削除しても、エラーとは見なされません。

特定の通知またはすべての通知を削除するには、DELETE を使用します。

構文

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

例

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

リクエストエンティティ**notification-id****詳細**

通知の名前。通知 ID が指定されていない場合は、バケットのすべての通知が削除されます。

型

String

HTTP レスポンス**404**

ステータスコード

NoSuchBucket**詳細**

バケットが存在しません。

3.4.5. バケットのホスト名へのアクセス

バケットにアクセスするモードは2つあります。最初のメソッドは推奨されるメソッドで、バケットを URI の最上位ディレクトリーとして識別します。

例

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

2 番目のメソッドは、仮想バケットのホスト名経由でバケットを識別します。

例

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

ヒント

2 番目の方法では高価なドメイン認定と DNS ワイルドカードが必要なため、Red Hat は最初の方法を推奨します。

3.4.6. S3 list buckets

GET / は、ユーザーがリクエストを行うユーザーが作成するバケットの一覧を返します。**GET /** は、認証ユーザーが作成したバケットのみを返します。匿名のリクエストを行うことはできません。

構文

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

```
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

レスポンスエンティティ**バケット****詳細**

バケットの一覧用のコンテナ。

型

Container

Bucket

詳細

バケット情報用のコンテナ

型

Container

名前

詳細

バケット名。

型

String

CreationDate

詳細

バケットが作成された時点の UTC 時間。

型

Date

ListAllMyBucketsResult

詳細

結果のコンテナ。

型

Container

Owner

詳細

バケット所有者の **ID** および **DisplayName** のコンテナ。

型

Container

ID

詳細

バケット所有者の ID。

型

String

DisplayName

詳細

バケットの所有者の表示名。

型

String

3.4.7. S3 return a list of bucket objects

バケットオブジェクトの一覧を返します。

構文

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

パラメーター

prefix

詳細

指定された接頭辞が含まれるオブジェクトのみを返します。

型

String

delimiter

詳細

接頭辞と他のオブジェクト名の上に挿入される区切り文字。

型

String

marker

詳細

返されるオブジェクトリストの開始インデックス。

型

String

max-keys

詳細

返すキーの最大数。デフォルトは 1000 です。

型

Integer

HTTP レスポンス

200

ステータスコード

OK

詳細

取得するバケット

GET /BUCKET は、以下のフィールドが含まれるバケットのコンテナを返します。

バケットレスポンスエンティティ

ListBucketResult

詳細

オブジェクト一覧のコンテナ。

型

エンティティ

名前

詳細

コンテンツが返されるバケットの名前。

型

String

接頭辞

詳細

オブジェクトキーの接頭辞。

型

String

Marker

詳細

返されるオブジェクトリストの開始インデックス。

型

String

MaxKeys

詳細

返されるキーの最大数。

型

Integer

デリミタ

詳細

設定されている場合は、同じ接頭辞を持つオブジェクトが **CommonPrefixes** リストに表示されます。

型

String

IsTruncated

詳細

true の場合、バケットの内容のサブセットのみが返されます。

型

Boolean

CommonPrefixes

詳細

複数のオブジェクトに同じ接頭辞が含まれる場合は、この一覧に表示されます。

型

Container

ListBucketResult にはオブジェクトが含まれ、各オブジェクトは **Contents** コンテナ内にあります。

オブジェクトレスポンスエンティティ

内容

詳細

オブジェクトのコンテナ。

型

Object

キー

詳細

オブジェクトのキー。

型

String

LastModified

詳細

オブジェクトの最終変更日および時間。

型

Date

ETag

詳細

オブジェクトの MD-5 ハッシュ。ETag はエンティティータグです。

型

String

サイズ

詳細

オブジェクトのサイズ。

型

Integer

StorageClass

詳細

常に **STANDARD** を返す必要があります。

型

String

3.4.8. S3 create a new bucket

新規バケットを作成します。バケットを作成するには、要求を認証するためにユーザー ID および有効な AWS アクセスキー ID が必要です。バケットを匿名ユーザーとして作成することはできません。

制約

通常、バケット名はドメイン名の制約に従う必要があります。

- バケット名は一意である必要があります。
- バケット名を IP アドレスとしてフォーマットすることはできません。
- バケット名の長さは 3～63 文字です。
- バケット名には、大文字やアンダースコアを含めることはできません。
- バケット名は小文字または数字で始まる必要があります。
- バケット名にはダッシュ (-) を含めることができます。
- バケット名は、一連の 1 つ以上のラベルである必要があります。隣接するラベルは単一のピリオド (.) で区切られます。バケット名には、小文字、数字、およびハイフンを含めることができます。各ラベルは、小文字または数字で開始および終了する必要があります。



注記

rgw_relaxed_s3_bucket_names が **true** に設定されている場合、上記の制約は緩和されます。バケット名は一意である必要があり、IP アドレスとしてフォーマットすることはできず、最大 255 文字の文字、数字、ピリオド、ダッシュ、およびアンダースコアを含めることができます。

構文

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

Authorization: AWS **ACCESS_KEY:HASH_OF_HEADER_AND_SECRET**

パラメーター

x-amz-acl

詳細

固定 ACL。

有効な値

private、**public-read**、**public-read-write**、**authenticated-read**

必須

いいえ

HTTP レスポンス

バケット名が一意で、制約内で未使用であると、操作は成功します。同じ名前のバケットがすでに存在し、ユーザーがバケット所有者である場合は、操作が成功します。バケット名が使用中の場合は、操作が失敗します。

409

ステータスコード

BucketAlreadyExists

詳細

バケットは、異なるユーザーの所有権に存在します。

3.4.9. S3 put bucket website

put bucket website API は、**website** サブリソースで指定されている Web サイトの設定を設定します。バケットを Web サイトとして設定するには、**website** のサブリソースをバケットに追加できます。



注記

put 操作には **S3:PutBucketWebsite** パーミッションが必要です。デフォルトでは、バケットの所有者のみがバケットに接続されている Web サイトを設定できます。

構文

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

例

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

関連情報

- この API 呼び出しの詳細は、[S3 API](#) を参照してください。

3.4.10. S3 get bucket website

get bucket website API は、**website** サブリソースで指定されている Web サイトの設定を取得します。



注記

Get 操作を実行するには、**S3:GetBucketWebsite** パーミッションが必要です。デフォルトでは、バケットの所有者のみがバケット Web 設定を読み取ることができます。

構文

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

例

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

関連情報

- この API 呼び出しの詳細は、[S3 API](#) を参照してください。

3.4.11. S3 delete bucket website

delete bucket website API は、バケットの Web サイト設定を削除します。

構文

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

例

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

関連情報

- この API 呼び出しの詳細は、[S3 API](#) を参照してください。

3.4.12. S3 delete a bucket

バケットを削除します。バケットの削除が正常に行われた後にバケット名を再利用できます。

構文

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

```
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

HTTP レスポンス

204

ステータスコード

コンテンツなし

詳細

バケットが削除されました。

3.4.13. S3 bucket lifecycle

バケットのライフサイクル設定を使用してオブジェクトを管理し、そのオブジェクトが有効期間中効果的に保存されるようにすることができます。Ceph Object Gateway の S3 API は、AWS バケットライフサイクルアクションのサブセットをサポートします。

- **Expiration:** これはバケット内のオブジェクトの有効期間を定義します。オブジェクトが存続する日数または有効期限がかかり、その時点で Ceph Object Gateway がオブジェクトを削除します。バケットがバージョン管理を有効にしない場合、Ceph Object Gateway はオブジェクトを永続的に削除します。バケットがバージョン管理を有効化する場合、Ceph Object Gateway は現行バージョンの削除マーカーを作成し、現行バージョンを削除します。
- **NoncurrentVersionExpiration:** これはバケット内の最新バージョン以外のオブジェクトバージョンのライフサイクルを定義します。この機能を使用するには、バケットがバージョン管理を有効にする必要があります。最新バージョン以外のオブジェクトが存続する日数を取ります。この時点では、Ceph Object Gateway が最新バージョン以外のオブジェクトを削除します。
- **AbortIncompleteMultipartUpload:** これは、非完全なマルチパートアップロードが中止されるまでの日数を定義します。

ライフサイクル設定には、**<Rule>** 要素を使用した1つ以上のルールが含まれます。

例

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

ライフサイクルルールは、ライフサイクルルールに指定する **<Filter>** 要素に基づいてバケットの全オブジェクトまたはサブセットに適用できます。フィルターは複数の方法を指定できます。

- キーの接頭辞
- オブジェクトタグ
- キー接頭辞と1つ以上のオブジェクトタグの両方

キーの接頭辞

ライフサイクルルールは、キー名の接頭辞に基づいてオブジェクトのサブセットに適用できます。たとえば、**<keypre/>** を指定すると、**keypre/** で始まるオブジェクトに適用されます。

```
<LifecycleConfiguration>
  <Rule>
    <Status>Enabled</Status>
    <Filter>
      <Prefix>keypre</Prefix>
    </Filter>
  </Rule>
</LifecycleConfiguration>
```

異なるキー接頭辞を持つオブジェクトに、異なるライフサイクルルールを適用することもできます。

```
<LifecycleConfiguration>
  <Rule>
```

```

    <Status>Enabled</Status>
    <Filter>
      <Prefix>keypre</Prefix>
    </Filter>
  </Rule>
  <Rule>
    <Status>Enabled</Status>
    <Filter>
      <Prefix>mypre</Prefix>
    </Filter>
  </Rule>
</LifecycleConfiguration>

```

オブジェクトタグ

ライフサイクルルールは、**<Key>** 要素および **<Value>** 要素を使用して、特定のタグを持つオブジェクトにのみ適用できます。

```

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

```

接頭辞および1つ以上のタグの両方

ライフサイクルルールでは、キーの接頭辞と1つ以上のタグの両方に基づいてフィルターを指定できます。これらは **<And>** 要素でラップする必要があります。フィルターには1つの接頭辞と、ゼロまたは複数のタグのみを使用できます。

```

<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>

```


関連情報

- バケットライフサイクルの取得の詳細については、Red Hat Ceph Storage 開発者ガイドの [S3 GET バケットライフサイクル](#) セクションを参照してください。
- バケットライフサイクルの作成の詳細については、Red Hat Ceph Storage 開発者ガイドの [S3 バケットライフサイクルの作成または置き換え](#) セクションを参照してください。
- バケットライフサイクルの削除の詳細については、Red Hat Ceph Storage 開発者ガイドの [S3 によるバケットライフサイクルの削除](#) セクションを参照してください。

3.4.14. S3 GET bucket lifecycle

バケットのライフサイクルを取得するには、**GET** を使用して宛先バケットを指定します。

構文

```
GET /BUCKET?lifecycle HTTP/1.1
Host: cname.domain.com
```

```
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

リクエストヘッダー

共通リクエストヘッダーの詳細については、付録 B の [S3 共通リクエストヘッダー](#) を参照してください。

応答

レスポンスには、バケットライフサイクルとその要素が含まれます。

3.4.15. S3 create or replace a bucket lifecycle

バケットライフサイクルを作成または置き換えるには、**PUT** を使用して宛先バケットとライフサイクル設定を指定します。Ceph Object Gateway は、S3 ライフサイクル機能のサブセットのみをサポートします。

構文

```
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>
```

リクエストヘッダー

content-md5

詳細

メッセージの base64 でエンコードされた MD-5 ハッシュ

有効な値

文字列。デフォルトや制約はありません。

必須

いいえ

関連情報

- Amazon S3 共通リクエストヘッダーの詳細については、[Red HatCephStorage 開発者ガイド](#) の付録 B の [S3 共通リクエストヘッダー](#) のセクションを参照してください。
- Amazon S3 バケットのライフサイクルの詳細については、[Red HatCephStorage 開発者ガイド](#) の [S3 バケットのライフサイクル](#) のセクションを参照してください。

3.4.16. S3 delete a bucket lifecycle

バケットライフサイクルを削除するには、**DELETE** を使用し、宛先バケットを指定します。

構文

DELETE /**BUCKET**?lifecycle HTTP/1.1

Host: cname.domain.com

Authorization: AWS **ACCESS_KEY:HASH_OF_HEADER_AND_SECRET**

リクエストヘッダー

リクエストには特別な要素が含まれません。

レスポンス

レスポンスは、一般的なレスポンスのステータスを返します。

関連情報

- Amazon S3 共通リクエストヘッダーの詳細については、[Red HatCephStorage 開発者ガイド](#) の付録 B の [S3 共通リクエストヘッダー](#) のセクションを参照してください。
- Amazon S3 共通応答ステータスコードの詳細については、[Red HatCephStorage 開発者ガイド](#) の付録 C の [S3 共通応答ステータスコード](#) のセクションを参照してください。

3.4.17. S3 get bucket location

バケットのゾングループを取得します。これ呼び出すには、ユーザーはバケット所有者である必要があります。PUT 要求時に **LocationConstraint** を指定して、バケットをゾングループに制限できます。

以下のように **location** サブリソースをバケットリソースに追加します。

構文

```
GET /BUCKET?location HTTP/1.1
Host: cname.domain.com
```

```
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

レスポンスエンティティ

LocationConstraint

説明

バケットが存在するゾーングループ (デフォルトゾーングループ用の空の文字列)

型

String

3.4.18. S3 get bucket versioning

バケットのバージョン状態を取得します。これ呼び出すには、ユーザーはバケット所有者である必要があります。

以下のように、**versioning** サブリソースをバケットリソースに追加します。

構文

```
GET /BUCKET?versioning HTTP/1.1
Host: cname.domain.com
```

```
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

3.4.19. S3 put bucket versioning

このサブリソースは、既存のバケットのバージョン管理状態を設定します。バージョン管理状態を設定するには、バケット所有者である必要があります。バージョン管理状態がバケットに設定されていないと、バージョンは管理されていません。GET バージョン管理リクエストを実行しても、バージョン管理状態の値は返されません。

バケットによるバージョン管理の状態を設定します。

Enabled: バケットのオブジェクトのバージョン管理を有効にします。バケットに追加したすべてのオブジェクトは、一意のバージョン ID を受信します。**Suspended:** バケットのオブジェクトのバージョン管理を無効にします。バケットに追加したすべてのオブジェクトは、バージョン ID の Null を受け取ります。

構文

```
PUT /BUCKET?versioning HTTP/1.1
```

例

```
PUT /testbucket?versioning HTTP/1.1
```

バケット要求のエンティティ

VersioningConfiguration

詳細

要求のコンテナ。

型

Container

状態

詳細

バケットのバージョン状態を設定します。有効な値: Suspended/Enabled

型

String

3.4.20. S3 get bucket access control lists

バケットのアクセス制御リストを取得します。ユーザーはバケットの所有者である必要があります。または、バケットで **READ_ACP** パーミッションが付与されている必要があります。

以下のように、**acl** サブリソースをバケット要求に追加します。

構文

```
GET /BUCKET?acl HTTP/1.1
```

```
Host: cname.domain.com
```

```
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

レスポンスエンティティ

AccessControlPolicy

詳細

応答のコンテナ。

型

Container

AccessControlList

詳細

ACL 情報用のコンテナ

型

Container

Owner

詳細

バケット所有者の **ID** および **DisplayName** のコンテナ。

型

Container

ID**詳細**

バケット所有者の ID。

型

String

DisplayName**詳細**

バケットの所有者の表示名。

型

String

Grant**詳細**

Grantee および **Permission** のコンテナ。

型

Container

Grantee**詳細**

パーミッションを付与されるユーザーの **DisplayName** および **ID** のコンテナ。

型

Container

パーミッション**詳細**

Grantee バケットに指定されるパーミッション。

型

String

3.4.21. S3 put bucket Access Control Lists

既存のバケットへのアクセス制御を設定します。ユーザーはバケットの所有者である必要があります。または、バケットの **WRITE_ACP** パーミッションが付与されている必要があります。

以下のように、**acl** サブリソースをバケット要求に追加します。

構文

```
PUT /BUCKET?acl HTTP/1.1
```

リクエストエンティティ

S3 list multipart uploads

AccessControlList

詳細

ACL 情報用のコンテナ

型

Container

Owner**詳細**

バケット所有者の **ID** および **DisplayName** のコンテナ。

型

Container

ID**詳細**

バケット所有者の ID。

型

String

DisplayName**詳細**

バケットの所有者の表示名。

型

String

Grant**詳細**

Grantee および **Permission** のコンテナ。

型

Container

Grantee**詳細**

パーミッションを付与されるユーザーの **DisplayName** および **ID** のコンテナ。

型

Container

パーミッション**詳細**

Grantee バケットに指定されるパーミッション。

型

String

3.4.22. S3 get bucket cors

バケットに設定された CORS 設定情報を取得します。ユーザーはバケットの所有者である必要があります。または、バケットで **READ_ACP** パーMISSIONが付与されている必要があります。

以下に示すように、**cors** サブリソースをバケット要求に追加します。

構文

```
GET /BUCKET?cors HTTP/1.1
```

```
Host: cname.domain.com
```

```
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

3.4.23. S3 put bucket cors

バケットの CORS 設定を設定します。ユーザーはバケットの所有者である必要があります。または、バケットで **READ_ACP** パーMISSIONが付与されている必要があります。

以下に示すように、**cors** サブリソースをバケット要求に追加します。

構文

```
PUT /BUCKET?cors HTTP/1.1
```

```
Host: cname.domain.com
```

```
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

3.4.24. S3 delete a bucket cors

バケットに設定された CORS 設定情報を削除します。ユーザーはバケットの所有者である必要があります。または、バケットで **READ_ACP** パーMISSIONが付与されている必要があります。

以下に示すように、**cors** サブリソースをバケット要求に追加します。

構文

```
DELETE /BUCKET?cors HTTP/1.1
```

```
Host: cname.domain.com
```

```
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

3.4.25. S3 list bucket object versions

バケット内のすべてのバージョンのオブジェクトに関するメタデータの一覧を返します。バケットへの READ アクセスが必要です。

以下のように **versions** サブリソースをバケット要求に追加します。

構文

```
GET /BUCKET?versions HTTP/1.1
```

```
Host: cname.domain.com
```

```
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

GET /BUCKET?versions のパラメーターを指定できますが、いずれも不要です。

パラメーター

prefix

説明

指定の接頭辞が含まれるキーが含まれる進行中のアップロードを返します。

型

String

delimiter

詳細

接頭辞と他のオブジェクト名の間に挿入される区切り文字。

型

String

key-marker

詳細

アップロード一覧の最初のマーカ。

型

String

max-keys

詳細

進行中のアップロードの最大数。デフォルトは1000 です。

型

Integer

version-id-marker

詳細

リストを開始するオブジェクトバージョンを指定します。

型

String

レスポンスエンティティ

KeyMarker

詳細

key-marker 要求パラメーターによって指定されるキーマーカ (存在する場合)。

型

String

NextKeyMarker

詳細

IsTruncated が **true** の場合に後続のリクエストで使用するキーマーカ。

型

String

NextUploadIdMarker

詳細

IsTruncated が **true** の場合に後続のリクエストで使用するアップロード ID マーカ。

型

String

IsTruncated

詳細

true の場合は、バケットのアップロードコンテンツのサブセットのみが返されます。

型

Boolean

サイズ

詳細

アップロードした部分のサイズ。

型

Integer

DisplayName

説明

所有者の表示名。

型

String

ID

説明

所有者の ID。

型

String

Owner

詳細

オブジェクトを所有するユーザーの **ID** および **DisplayName** のコンテナ。

型

Container

StorageClass

詳細

作成されるオブジェクトを保存するために使用されるメソッド。 **STANDARD** または **REDUCED_REDUNDANCY**

型

String

バージョン**詳細**

バージョン情報のコンテナ

型

Container

versionId**詳細**

オブジェクトのバージョン ID。

型

String

versionIdMarker**詳細**

省略されたレスポンスのキーの最後のバージョン。

型

String

3.4.26. S3 head bucket

バケットで HEAD を呼び出して、存在する場合は、呼び出し元にアクセス権限があるかどうかを判断します。バケットが存在し、呼び出し元にパーミッションがある場合は **200 OK** を返します。バケットが存在しない場合は **404 Not Found**、バケットが存在しますが呼び出し元にはアクセスパーミッションがない場合は **403 Forbidden** を返します。

構文HEAD /**BUCKET** HTTP/1.1

Host: cname.domain.com

Date: date

Authorization: AWS **ACCESS_KEY**:**HASH_OF_HEADER_AND_SECRET**

3.4.27. S3 list multipart uploads

GET /?uploads は、現在の進行中のマルチパートアップロードの一覧を返します。つまり、アプリケーションは複数パートごとのアップロードを開始しますが、サービスがすべてのアップロードを完了しているわけではありません。

構文GET /**BUCKET**?uploads HTTP/1.1

GET /BUCKET?uploads のパラメーターを指定できますが、いずれも不要です。

パラメーター

prefix

説明

指定の接頭辞が含まれるキーが含まれる進行中のアップロードを返します。

型

String

delimiter

詳細

接頭辞と他のオブジェクト名の上に挿入される区切り文字。

型

String

key-marker

詳細

アップロード一覧の最初のマーカー。

型

String

max-keys

詳細

進行中のアップロードの最大数。デフォルトは 1000 です。

型

Integer

max-uploads

詳細

マルチパートアップロードの最大数。範囲は 1-1000 です。デフォルトは 1000 です。

型

Integer

version-id-marker

詳細

key-marker が指定されていない場合は無視されます。**ID** またはそれに続く辞書順序でリストされる最初のアップロードの **ID** を指定します。

型

String

レスポンスエンティティ

ListMultipartUploadsResult

詳細

結果のコンテナ

型

Container

ListMultipartUploadsResult.Prefix**説明**

prefix 要求パラメーターで指定される接頭辞 (存在する場合)。

型

String

Bucket**詳細**

バケットのコンテンツを受け取るバケット。

型

String

KeyMarker**詳細**

key-marker 要求パラメーターによって指定されるキーマーカ (存在する場合)。

型

String

UploadIdMarker**詳細**

upload-id-marker 要求パラメーターによって指定されるマーカ (存在する場合)。

型

String

NextKeyMarker**詳細**

IsTruncated が **true** の場合に後続のリクエストで使用するキーマーカ。

型

String

NextUploadIdMarker**詳細**

IsTruncated が **true** の場合に後続のリクエストで使用するアップロード ID マーカ。

型

String

MaxUploads**詳細**

max-uploads リクエストパラメーターで指定される最大アップロード数。

型

Integer

デリミタ

詳細

設定されている場合は、同じ接頭辞を持つオブジェクトが **CommonPrefixes** リストに表示されます。

型

String

IsTruncated

詳細

true の場合は、バケットのアップロードコンテンツのサブセットのみが返されます。

型

Boolean

Upload

詳細

Key、**UploadId**、**InitiatorOwner**、**StorageClass**、および **Initiated** 要素のコンテナ。

型

Container

キー

詳細

マルチパートアップロードが完了した後のオブジェクトのキー。

型

String

UploadId

詳細

マルチパートアップロードを識別する **ID**。

型

String

イニシエーター

詳細

アップロードを開始したユーザーの **ID** と **DisplayName** が含まれます。

型

Container

DisplayName

詳細

イニシエーターの表示名。

型

String

ID

詳細

イニシエーターの ID。

型

String

Owner**詳細**

アップロードしたオブジェクトを所有するユーザーの **ID** および **DisplayName** のコンテナ。

型

Container

StorageClass**詳細**

作成されるオブジェクトを保存するために使用されるメソッド。 **STANDARD** または **REDUCED_REDUNDANCY**

型

String

Initiated**詳細**

ユーザーがアップロードを開始した日時。

型

Date

CommonPrefixes**詳細**

複数のオブジェクトに同じ接頭辞が含まれる場合は、この一覧に表示されます。

型

Container

CommonPrefixes.Prefix**詳細**

prefix リクエストパラメーターで定義されている接頭辞の後にキーのサブ文字列。

型

String

3.4.28. S3 bucket policies

Ceph Object Gateway は、バケットに適用される Amazon S3 ポリシー言語のサブセットをサポートします。

作成および削除

Ceph Object Gateway は、CLI ツール **radosgw-admin** を使用するのではなく、標準の S3 操作を使用して S3 バケットポリシーを管理します。

管理者は、**s3cmd** コマンドを使用してポリシーを設定または削除できます。

例

```
$ 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
```

制限

Ceph Object Gateway がサポートするのは以下の S3 アクションだけです。

- **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**



注記

Ceph Object Gateway は、ユーザー、グループ、またはロールへのポリシー設定をサポートしません。

Ceph Object Gateway は、Amazon の 12 桁のアカウント ID の代わりに RGW の tenant 識別子を使用します。Amazon Web Service (AWS) S3 と Ceph Object Gateway S3 との間でポリシーを使用する場合、Ceph Object Gateway は、ユーザーの作成時に Amazon アカウント ID をテナント ID として使用する必要があります。

AWS S3 では、すべてのテナントが単一の名前空間を共有します。対照的に、Ceph Object Gateway はすべてのテナントにバケットの独自の名前空間を提供します。現在、別のテナントに属するバケットにアクセスしようとしている Ceph Object Gateway クライアントは、S3 リクエストの **tenant:bucket** としてそれを処理する必要があります。

AWS では、バケットポリシーは別のアカウントへのアクセスを許可し、そのアカウントの所有者はユーザーパーミッションを持つ個々のユーザーにアクセス権限を付与できます。Ceph Object Gateway はユーザー、ロール、およびグループのパーミッションをサポートしていません。そのため、アカウントの所有者は個々のユーザーに直接アクセスを付与する必要があります。



重要

アカウント全体のアクセスをバケットに付与すると、そのアカウントのすべてのユーザーにアクセス権限が付与されます。

バケットポリシーは文字列の補正を **サポートしません**。

Ceph Object Gateway では、以下の条件キーがサポートされます。

- **aws:CurrentTime**
- **aws:EpochTime**
- **aws:PrincipalType**
- **aws:Referer**
- **aws:SecureTransport**
- **aws:SourceIp**
- **aws:UserAgent**
- **aws:username**

Ceph Object Gateway **のみ** は、**ListBucket** アクションの以下の条件キーをサポートします。

- **s3:prefix**
- **s3:delimiter**
- **s3:max-keys**

Swift への影響

Ceph Object Gateway は、Swift API にバケットポリシーを設定する機能はありません。ただし、S3 API で設定されているバケットポリシーは Swift と S3 のいずれの操作も管理します。

Ceph Object Gateway は、ポリシーで指定されたプリンシパルに対して Swift の認証情報と一致します。

3.4.29. S3 get the request payment configuration on a bucket

requestPayment サブリソースを使用してバケットの要求支払い設定を返します。ユーザーはバケットの所有者である必要があります。または、バケットで **READ_ACP** パーミッションが付与されている必要があります。

以下のように **requestPayment** サブリソースをバケット要求に追加します。

構文

```
GET /BUCKET?requestPayment HTTP/1.1
Host: cname.domain.com
```

```
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

3.4.30. S3 set the request payment configuration on a bucket

requestPayment サブリソースを使用してバケットの要求支払い設定を設定します。デフォルトでは、バケットの所有者はバケットからのダウンロードに対して支払います。この設定パラメーターにより、バケットの所有者は、ダウンロードを要求するすべてのユーザーが要求に対して要求およびバケットからダウンロードに対して課金されることを指定できます。

以下のように **requestPayment** サブリソースをバケット要求に追加します。

構文

```
PUT /BUCKET?requestPayment HTTP/1.1
Host: cname.domain.com
```

リクエストエンティティ

Payer

詳細

ダウンロードおよびリクエストの費用の課金を指定します。

型

Enum

RequestPaymentConfiguration

詳細

Payer のコンテナ。

型

Container

3.4.31. マルチテナントバケット操作

クライアントアプリケーションがバケットにアクセスする場合は、常に特定ユーザーの認証情報で動作します。Red Hat Ceph Storage クラスターでは、すべてのユーザーがテナントに属します。そのため、テナントが明示的に指定されていない場合は、すべてのバケット操作のコンテキストに暗黙的なテナントがあります。したがって、マルチテナンシーは、参照されるバケットと参照ユーザーが同じテナントに属する限り、以前のリリースと完全に後方互換性があります。

明示的なテナントの指定に使用される拡張機能は、使用されるプロトコルおよび認証システムによって異なります。

以下の例では、コロンの文字はテナントとバケットを分離します。そのため、URL のサンプルは以下のようになります。

```
https://rgw.domain.com/tenant:bucket
```

一方、単純な Python の例は、バケットメソッド自体でテナントとバケットを分離します。

例

```
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")
```



注記

ホスト名に、コロンや、バケット名では有効ではない他の区切り文字を含めることができないため、マルチテナンシーを使用して S3 形式のサブドメインを使用することはできません。期間を使用するとあいまいな構文が作成されます。そのため、**bucket-in-URL-path** 形式をマルチテナンシーと併用する必要があります。

関連情報

- 詳細は、Red Hat Ceph Storage Object Gateway ガイドの ユーザー管理 の [マルチテナンシー](#) セクションを参照してください。

3.4.32. S3 ブロックパブリックアクセス

S3 のパブリックアクセスのブロック機能を使用して、Amazon S3 リソースへのパブリックアクセスの管理に役立つアクセスポイント、バケット、アカウントを設定できます。

この機能を使用すると、バケットポリシー、アクセスポイントポリシー、およびオブジェクトの権限をオーバーライドしてパブリックアクセスを許可できます。デフォルトでは、新しいバケット、アクセスポイント、およびオブジェクトはパブリックアクセスを許可しません。

Ceph Object Gateway の S3 API は、AWS パブリックアクセス設定のサブセットをサポートしています。

- **BlockPublicPolicy**: ユーザーがアクセスポイントとバケットのポリシーを管理できるようにする設定を定義します。この設定では、ユーザーがバケットまたはバケットに含まれるオブジェクトをパブリックに共有することはできません。既存のアクセスポイントとバケットのポリシーは、この設定を有効にしても影響を受けません。このオプションを **TRUE** に設定すると、S3 は次のようになります。
 - PUT Bucket ポリシーへの呼び出しを拒否します。
 - バケットの同じアカウントのすべてのアクセスポイントに対する PUT アクセスポイントポリシーへの呼び出しを拒否するには。



重要

この設定を **account** レベルで適用すると、ユーザーは特定のバケットのパブリックアクセスのブロック設定を変更できなくなります。



注記

TRUE 設定は、指定されたポリシーでパブリックアクセスが許可されている場合にのみ機能します。

- **RestrictPublicBuckets**: これは、パブリックポリシーを使用してバケットまたはアクセスポイントへのアクセスを制限する設定を定義します。この制限は、AWS サービスプリンシパルと、

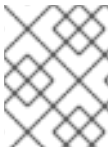
バケット所有者のアカウントおよびアクセスポイント所有者のアカウント内の承認されたユーザーにのみ適用されます。これにより、指定された場合を除き、アクセスポイントまたはバケットへのクロスアカウントアクセスがブロックされますが、アカウント内のユーザーは引き続きアクセスポイントまたはバケットを管理できます。この設定を有効にしても、既存のアクセスポイントまたはバケットポリシーには影響しません。これは、Amazon S3 が、特定のアカウントへの非公開委任を含む、パブリックアクセスポイントまたはバケットポリシーに由来するパブリックアクセスおよびクロスアカウントアクセスをブロックすることのみを定義します。



注記

アクセス制御リスト (ACL) は現在、Red Hat Ceph Storage ではサポートされていません。

特に定義されていない限り、バケットポリシーはパブリックであるとみなされます。パブリックアクセスをブロックするには、バケットポリシーで次の1つ以上の固定値へのアクセスのみを許可する必要があります。



注記

固定値には、ワイルドカード (*) または AWS Identity and Access Management ポリシー変数は含まれません。

- AWS プリンシパル、ユーザー、ロール、またはサービスプリンシパル
- **aws:SourceIp** を使用したクラスレスドメイン間ルーティング (CIDR) のセット
- **aws:SourceArn**
- **aws:SourceVpc**
- **aws:SourceVpce**
- **aws:SourceOwner**
- **aws:SourceAccount**
- **s3:x-amz-server-side-encryption-aws-kms-key-id**
- **aws:userid**、パターン **AROLEID:*** の外
- **s3:DataAccessPointArn**



注記

バケットポリシーで使用する場合、アカウント ID が固定されている限り、ポリシーを公開せずに、この値にアクセスポイント名のワイルドカードを含めることができます。

- **s3:DataAccessPointAccount**

次のポリシー例はパブリックとみなされます。

例

-

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow",
  "Condition": { "StringLike": {"aws:SourceVpc": "vpc-*"} }
}
```

ポリシーを非公開にするには、固定値を持つ条件キーのいずれかを含めます。

例

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow",
  "Condition": { "StringEquals": {"aws:SourceVpc": "vpc-91237329"} }
}
```

関連情報

- PublicAccessBlock の取得の詳細は、Red Hat Ceph Storage 開発者ガイドの [S3 GET `PublicAccessBlock`](#) セクションを参照してください。
- PublicAccessBlock の作成または変更の詳細は、Red Hat Ceph Storage 開発者ガイドの [S3 PUT `PublicAccessBlock`](#) セクションを参照してください。
- PublicAccessBlock の削除の詳細は、Red Hat Ceph Storage 開発者ガイドの [S3 Delete `PublicAccessBlock`](#) セクションを参照してください。
- バケットポリシーの詳細は、Red Hat Ceph Storage 開発者ガイドの [S3 バケットポリシー](#) セクションを参照してください。
- Amazon Simple Storage Service (S3) ドキュメントの [Amazon S3 ストレージへのパブリックアクセスのブロック](#) セクションを参照してください。

3.4.33. S3 GET PublicAccessBlock

S3 ブロックパブリックアクセス機能を設定するには、**GET** を使用して宛先 AWS アカウントを指定します。

構文

```
GET /v20180820/configuration/publicAccessBlock HTTP/1.1
Host: cname.domain.com
x-amz-account-id: _ACCOUNTID_
```

リクエストヘッダー

一般的なリクエストヘッダーの詳細は、付録 B の [S3 の一般的なリクエストヘッダー](#) を参照してください。

応答

応答は HTTP 200 応答であり、XML 形式で返されます。

3.4.34. S3 PUT PublicAccessBlock

これを使用して、S3 バケットの **PublicAccessBlock** 設定を作成または変更します。

この操作を使用するには、**s3:PutBucketPublicAccessBlock** 権限が必要です。



重要

PublicAccessBlock 設定がバケットとアカウント間で異なる場合、Amazon S3 はバケットレベルとアカウントレベルの設定の最も制限的な組み合わせを使用します。

構文

```
PUT /?publicAccessBlock HTTP/1.1
Host: Bucket.s3.amazonaws.com
Content-MD5: ContentMD5
x-amz-sdk-checksum-algorithm: ChecksumAlgorithm
x-amz-expected-bucket-owner: ExpectedBucketOwner
<?xml version="1.0" encoding="UTF-8"?>
<PublicAccessBlockConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <BlockPublicAcls>boolean</BlockPublicAcls>
  <IgnorePublicAcls>boolean</IgnorePublicAcls>
  <BlockPublicPolicy>boolean</BlockPublicPolicy>
  <RestrictPublicBuckets>boolean</RestrictPublicBuckets>
</PublicAccessBlockConfiguration>
```

リクエストヘッダー

一般的なリクエストヘッダーの詳細は、付録 B の [S3 の一般的なリクエストヘッダー](#) を参照してください。

応答

応答は HTTP 200 応答であり、空の HTTP 本文が返されます。

3.4.35. S3 delete PublicAccessBlock

これを使用して、S3 バケットの **PublicAccessBlock** 設定を削除します。

構文

```
DELETE /v20180820/configuration/publicAccessBlock HTTP/1.1
Host: s3-control.amazonaws.com
x-amz-account-id: AccountId
```

リクエストヘッダー

一般的なリクエストヘッダーの詳細は、付録 B の [S3 の一般的なリクエストヘッダー](#) を参照してください。

応答

応答は HTTP 200 応答であり、空の HTTP 本文が返されます。

3.5. S3 オブジェクト操作

開発者は、Ceph Object Gateway 経由で Amazon S3 アプリケーションプログラミングインターフェイス (API) を使用してオブジェクト操作を行うことができます。

以下の表は、関数のサポートステータスとともに、オブジェクトの Amazon S3 の機能操作を示しています。

表3.3 オブジェクト操作

機能	状態
Get Object	サポート対象
Get Object Information	サポート対象
Put Object Lock	サポート対象
Get Object Lock	サポート対象
Put Object Legal Hold	サポート対象
Get Object Legal Hold	サポート対象
Put Object Retention	サポート対象
Get Object Retention	サポート対象
Put Object Tagging	サポート対象
Get Object Tagging	サポート対象
Delete Object Tagging	サポート対象
Put Object	サポート対象
Delete Object	サポート対象
Delete Multiple Objects	サポート対象
Get Object ACLs	サポート対象
Put Object ACLs	サポート対象
Copy Object	サポート対象
Post Object	サポート対象
Options Object	サポート対象

機能	状態
Initiate Multipart Upload	サポート対象
Add a Part to a Multipart Upload	サポート対象
List Parts of a Multipart Upload	サポート対象
Assemble Multipart Upload	サポート対象
Copy Multipart Upload	サポート対象
Abort Multipart Upload	サポート対象
マルチテナンシー	サポート対象

3.5.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。
- RESTful クライアント。

3.5.2. S3 get an object from a bucket

バケットからオブジェクトを取得します。

構文

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

versionId サブリソースを追加して、オブジェクトの特定のバージョンを取得します。

構文

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

リクエストヘッダー

range

詳細

取得するオブジェクトの範囲。

有効な値

範囲: bytes=beginbyte-endbyte

必須

いいえ

if-modified-since**詳細**

タイムスタンプ以降に変更した場合にのみ取得します。

有効な値

Timestamp

必須

いいえ

if-match**詳細**

オブジェクトの ETag が ETag と一致する場合にのみ取得します。

有効な値

エンティティータグ

必須

いいえ

if-none-match**詳細**

オブジェクトの ETag が ETag と一致する場合にのみ取得します。

有効な値

エンティティータグ

必須

いいえ

レスポンスヘッダー**Content-Range****詳細**

データ範囲 (範囲ヘッダーフィールドがリクエストに指定された場合のみを返します)。

x-amz-version-id**詳細**

バージョン ID または Null を返します。

3.5.3. S3 get information on an object

オブジェクトに関する情報を返します。この要求は Get Object 要求と同じヘッダー情報を返しますが、オブジェクトデータペイロードではなくメタデータのみが含まれます。

オブジェクトの現行バージョンを取得します。

構文

HEAD /**BUCKET**/**OBJECT** HTTP/1.1

versionId サブリソースを追加して、特定バージョンの情報を取得します。

構文

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

リクエストヘッダー

range

詳細

取得するオブジェクトの範囲。

有効な値

範囲: bytes=beginbyte-endbyte

必須

いいえ

if-modified-since

詳細

タイムスタンプ以降に変更した場合にのみ取得します。

有効な値

Timestamp

必須

いいえ

if-match

詳細

オブジェクトの ETag が ETag と一致する場合にのみ取得します。

有効な値

エンティティータグ

必須

いいえ

if-none-match

詳細

オブジェクトの ETag が ETag と一致する場合にのみ取得します。

有効な値

エンティティータグ

必須

いいえ

レスポンスヘッダー

x-amz-version-id

詳細

バージョン ID または Null を返します。

3.5.4. S3 put object lock

put object lock API は、選択したバケットにロック設定を配置します。オブジェクトロックを使用すると、**write-once-read-many**(WORM) モデルを使用してオブジェクトを格納できます。オブジェクトロックは、オブジェクトが一定期間または無期限に削除または上書きされないようにします。Object Lock 設定に指定されるルールは、デフォルトで選択されるバケットにあるすべての新規オブジェクトに適用されます。



重要

バケットを作成するときにオブジェクトロックを有効にします。有効にしないと、操作が失敗します。

構文

```
PUT /BUCKET?object-lock HTTP/1.1
```

例

```
PUT /testbucket?object-lock HTTP/1.1
```

リクエストエンティティ

ObjectLockConfiguration

詳細

要求のコンテナ。

型

Container

必須

はい

ObjectLockEnabled

詳細

このバケットにオブジェクトロック設定が有効になっているかどうかを示します。

型

String

必須

はい

ルール

詳細

指定されたバケットの位置にあるオブジェクトロックルール。

型

Container

必須

いいえ

DefaultRetention

詳細

指定されたバケットに設定される新規オブジェクトに適用されるデフォルトの保持期間。

型

Container

必須

いいえ

Mode

詳細

デフォルトのオブジェクトのロック保持モード。有効な値は GOVERNANCE/COMPLIANCE です。

型

Container

必須

はい

Days

詳細

デフォルトの保持期間に指定される日数。

型

整数

必須

いいえ

Years

詳細

デフォルトの保持期間に指定されるの年数。

型

整数

必須

いいえ

HTTP レスポンス

400

ステータスコード

MalformedXML

詳細

XML は適していません。

409

ステータスコード

InvalidBucketState

詳細

バケットオブジェクトのロックが有効になっていません。

関連情報

- この API 呼び出しの詳細は、[S3 API](#) を参照してください。

3.5.5. S3 get object lock

get object lock API は、バケットのロック設定を取得します。

構文

```
GET /BUCKET?object-lock HTTP/1.1
```

例

```
GET /testbucket?object-lock HTTP/1.1
```

レスポンスエンティティ

ObjectLockConfiguration

詳細

要求のコンテナ。

型

Container

必須

はい

ObjectLockEnabled

詳細

このバケットにオブジェクトロック設定が有効になっているかどうかを示します。

型

String

必須

はい

****ルール****

説明

指定されたバケットの位置に、オブジェクトロックルールがあります。

型

Container

必須

いいえ

DefaultRetention

詳細

指定されたバケットに設定される新規オブジェクトに適用されるデフォルトの保持期間。

型

Container

必須

いいえ

Mode

詳細

デフォルトのオブジェクトのロック保持モード。有効な値は GOVERNANCE/COMPLIANCE です。

型

Container

必須

はい

Days

詳細

デフォルトの保持期間に指定される日数。

型

整数

必須

いいえ

Years

詳細

デフォルトの保持期間に指定されるの年数。

型

整数

必須

いいえ

関連情報

- この API 呼び出しの詳細は、[S3 API](#) を参照してください。

3.5.6. S3 put object legal hold

put object legal hold API は、選択したオブジェクトに有効な保持設定を適用します。法的な保持機能で、オブジェクトバージョンを上書きしたり、削除したりすることはできません。法的保持には関連付けられた保持期間がなく、明示的に削除されるまでそのまま維持されます。

構文

```
PUT /BUCKET/OBJECT?legal-hold&versionId= HTTP/1.1
```

例

```
PUT /testbucket/testobject?legal-hold&versionId= HTTP/1.1
```

versionId サブリソースは、オブジェクトの特定のバージョンを取得します。

リクエストエンティティ

LegalHold

詳細

要求のコンテナ。

型

Container

必須

はい

状態

詳細

指定されたオブジェクトに正当な保留があるかどうかを示します。有効な値は ON/OFF です。

型

String

必須

はい

関連情報

- この API 呼び出しの詳細は、[S3 API](#) を参照してください。

3.5.7. S3 get object legal hold

get object legal hold API は、オブジェクトの現在の法的保持ステータスを取得します。

構文

```
GET /BUCKET/OBJECT?legal-hold&versionId= HTTP/1.1
```

例

```
GET /testbucket/testobject?legal-hold&versionId= HTTP/1.1
```

versionId サブリソースは、オブジェクトの特定のバージョンを取得します。

レスポンスエンティティ

LegalHold

詳細

要求のコンテナ。

型

Container

必須

はい

状態

詳細

指定されたオブジェクトに正当な保留があるかどうかを示します。有効な値は ON/OFF です。

型

String

必須

はい

関連情報

- この API 呼び出しの詳細は、[S3 API](#) を参照してください。

3.5.8. S3 put object retention

put object retention API は、オブジェクトの保持設定をオブジェクトに配置します。保持期間は、オブジェクトのバージョンを一定時間保護します。ガバナンスモードとコンプライアンスモードの2つのモードがあります。これら2つの保持モードは、オブジェクトに対して異なる保護レベルを適用します。



注記

この期間中、オブジェクトは **write-once-read-many**(WORM 保護) で、上書きまたは削除することはできません。

構文

```
PUT /BUCKET/OBJECT?retention&versionId= HTTP/1.1
```

例

```
PUT /testbucket/testobject?retention&versionId= HTTP/1.1
```

versionId サブリソースは、オブジェクトの特定のバージョンを取得します。

リクエストエンティティ

Retention

詳細

要求のコンテナ。

型

Container

必須

はい

Mode**詳細**

指定されたオブジェクトの保持モード。有効な値: GOVERNANCE/COMPLIANCE

型

String

必須

はい

RetainUntilDate**詳細**

保持日。形式: 2020-01-05T00:00:00.000Z

型

Timestamp

必須

はい

関連情報

- この API 呼び出しの詳細は、[S3 API](#) を参照してください。

3.5.9. S3 get object retention

get object retention API は、オブジェクト上でオブジェクトの保持設定を取得します。

構文

```
GET /BUCKET/OBJECT?retention&versionId= HTTP/1.1
```

例

```
GET /testbucket/testobject?retention&versionId= HTTP/1.1
```

versionId サブリソースは、オブジェクトの特定のバージョンを取得します。

レスポンスエンティティ**Retention****詳細**

要求のコンテナ。

型

Container

必須

はい

Mode**詳細**

指定されたオブジェクトの保持モード。有効な値: GOVERNANCE/COMPLIANCE

型

String

必須

はい

RetainUntilDate**詳細**

保持日。形式: 2020-01-05T00:00:00.000Z

型

Timestamp

必須

はい

関連情報

- この API 呼び出しの詳細は、[S3 API](#) を参照してください。

3.5.10. S3 put object tagging

put object tagging API は、タグをオブジェクトに関連付けます。タグはキーと値のペアです。他のバージョンのタグを配置するには、**versionId** クエリーパラメーターを使用します。**s3:PutObjectTagging** アクションを実行するパーミッションが必要です。デフォルトでは、バケットの所有者はこのパーミッションを持ち、このパーミッションを他のユーザーに付与できます。

構文

```
PUT /BUCKET/OBJECT?tagging&versionId= HTTP/1.1
```

例

```
PUT /testbucket/testobject?tagging&versionId= HTTP/1.1
```

リクエストエンティティ**タグ付け****詳細**

要求のコンテナ。

型

Container

必須

はい

TagSet

詳細

タグのセットのコレクションです。

型

String

必須

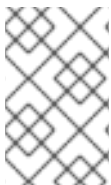
はい

関連情報

- この API 呼び出しの詳細は、[S3 API](#) を参照してください。

3.5.11. S3 get object tagging

get object tagging API は、オブジェクトのタグを返します。デフォルトでは、**GET** 操作はオブジェクトの現行バージョンについての情報を返します。



注記

バージョン付けされたバケットの場合は、バケットに複数のバージョンのオブジェクトを使用できます。他のバージョンのタグを取得するには、要求に **versionId** クエリーパラメーターを追加します。

構文

```
GET /BUCKET/OBJECT?tagging&versionId= HTTP/1.1
```

例

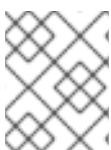
```
GET /testbucket/testobject?tagging&versionId= HTTP/1.1
```

関連情報

- この API 呼び出しの詳細は、[S3 API](#) を参照してください。

3.5.12. S3 delete object tagging

delete object tagging API は、指定されたオブジェクトから設定されたタグ全体を削除します。この操作を使用するには、**s3:DeleteObjectTagging** アクションを実行するパーミッションが必要です。



注記

特定のオブジェクトバージョンのタグを削除するには、リクエストに **versionId** クエリーパラメーターを追加します。

構文

■

DELETE **/BUCKET/OBJECT**?tagging&versionId= HTTP/1.1

例

DELETE /testbucket/testobject?tagging&versionId= HTTP/1.1

関連情報

- この API 呼び出しの詳細は、[S3 API](#) を参照してください。

3.5.13. S3 add an object to a bucket

オブジェクトをバケットに追加します。この操作を実行するには、バケットに書き込みパーミッションが必要です。

構文

PUT **/BUCKET/OBJECT** HTTP/1.1

リクエストヘッダー

content-md5

詳細

メッセージの base64 でエンコードされた MD-5 ハッシュ

有効な値

文字列。デフォルトや制約はありません。

必須

いいえ

content-type

詳細

標準の MIME タイプ。

有効な値

MIME タイプ。デフォルト: **binary/octet-stream**

必須

いいえ

x-amz-meta-<...>*

詳細

ユーザーのメタデータ。オブジェクトとともに保存されます。

有効な値

8kb までの文字列。デフォルトはありません。

必須

いいえ

x-amz-acl

詳細

固定 ACL。

有効な値

private、**public-read**、**public-read-write**、**authenticated-read**

必須

いいえ

レスポンスヘッダー**x-amz-version-id****詳細**

バージョン ID または Null を返します。

3.5.14. S3 delete an object

オブジェクトを削除します。含まれるバケットに WRITE パーミッションを設定する必要があります。

オブジェクトを削除します。オブジェクトのバージョン管理が有効な場合、マーカーが作成されます。

構文

DELETE /BUCKET/OBJECT HTTP/1.1

バージョン管理が有効な場合にオブジェクトを削除するには、**versionId** サブリソースおよび削除するオブジェクトのバージョンを指定する必要があります。

DELETE /BUCKET/OBJECT?versionId=VERSION_ID HTTP/1.1

3.5.15. S3 delete multiple objects

この API 呼び出しは、バケットから複数のオブジェクトを削除します。

構文

POST /BUCKET/OBJECT?delete HTTP/1.1

3.5.16. S3 get an object's Access Control List (ACL)

オブジェクトの現行バージョンの ACL を返します。

構文

GET /BUCKET/OBJECT?acl HTTP/1.1

versionId サブリソースを追加して、特定バージョンの ACL を取得します。

構文

GET /**BUCKET/OBJECT**?versionId=**VERSION_ID**&acl HTTP/1.1

レスポンスヘッダー

x-amz-version-id

詳細

バージョン ID または Null を返します。

レスポンスエンティティ

AccessControlPolicy

詳細

応答のコンテナ。

型

Container

AccessControlList

詳細

ACL 情報用のコンテナ

型

Container

Owner

詳細

バケット所有者の **ID** および **DisplayName** のコンテナ。

型

Container

ID

詳細

バケット所有者の ID。

型

String

DisplayName

詳細

バケットの所有者の表示名。

型

String

Grant

詳細

Grantee および **Permission** のコンテナ。

型

Container

Grantee

詳細

パーミッションを付与されるユーザーの **DisplayName** および **ID** のコンテナ。

型

Container

パーミッション

詳細

Grantee バケットに指定されるパーミッション。

型

String

3.5.17. S3 set an object's Access Control List (ACL)

オブジェクトの現行バージョンのオブジェクト ACL を設定します。

構文

```
PUT /BUCKET/OBJECT?acl
```

リクエストエンティティ

AccessControlPolicy

詳細

応答のコンテナ。

型

Container

AccessControlList

詳細

ACL 情報用のコンテナ

型

Container

Owner

詳細

バケット所有者の **ID** および **DisplayName** のコンテナ。

型

Container

ID

詳細

バケット所有者の ID。

型

String

DisplayName**詳細**

バケットの所有者の表示名。

型

String

Grant**詳細****Grantee** および **Permission** のコンテナ。**型**

Container

Grantee**詳細**パーミッションを付与されるユーザーの **DisplayName** および **ID** のコンテナ。**型**

Container

パーミッション**詳細****Grantee** バケットに指定されるパーミッション。**型**

String

3.5.18. S3 copy an object

オブジェクトをコピーするには、**PUT** を使用して宛先バケットとオブジェクト名を指定します。

構文

```
PUT /DEST_BUCKET/DEST_OBJECT HTTP/1.1
x-amz-copy-source: SOURCE_BUCKET/SOURCE_OBJECT
```

リクエストヘッダー**x-amz-copy-source****詳細**

ソースバケット名 + オブジェクト名。

有効な値**BUCKET/OBJECT****必須**

はい

x-amz-acl**詳細**

固定 ACL。

有効な値

private、**public-read**、**public-read-write**、**authenticated-read**

必須

いいえ

x-amz-copy-if-modified-since**詳細**

タイムスタンプ以降に変更された場合のみコピーします。

有効な値

Timestamp

必須

いいえ

x-amz-copy-if-unmodified-since**詳細**

タイムスタンプ以降変更されていない場合にのみコピーします。

有効な値

Timestamp

必須

いいえ

x-amz-copy-if-match**詳細**

オブジェクトの ETag が ETag と一致する場合に限りコピーします。

有効な値

エンティティータグ

必須

いいえ

x-amz-copy-if-none-match**詳細**

オブジェクトの ETag が ETag と一致する場合に限りコピーします。

有効な値

エンティティータグ

必須

いいえ

レスポンスエンティティ

CopyObjectResult

詳細

レスポンス要素のコンテナ。

型

Container

LastModified**詳細**

ソースオブジェクトを最後に変更した日付。

型

Date

Etag**詳細**

新規オブジェクトの ETag。

型

String

3.5.19. S3 add an object to a bucket using HTML forms

HTML フォームを使用してオブジェクトをバケットに追加します。この操作を実行するには、バケットに書き込みパーミッションが必要です。

構文

```
POST /BUCKET/OBJECT HTTP/1.1
```

3.5.20. S3 determine options for a request

特定の送信元、HTTP メソッド、およびヘッダーを使用して実際のリクエストを送信できるかどうかを判断するための事前要求です。

構文

```
OPTIONS /OBJECT HTTP/1.1
```

3.5.21. S3 initiate a multipart upload

複数パートからなるアップロードプロセスを開始します。追加部分の追加、パーツの一覧表示、および複数パートアップロードの完了または破棄時に指定できる **UploadId** を返します。

構文

```
POST /BUCKET/OBJECT?uploads
```

リクエストヘッダー

content-md5

詳細

メッセージの base64 でエンコードされた MD-5 ハッシュ

有効な値

文字列。デフォルトや制約はありません。

必須

いいえ

content-type**詳細**

標準の MIME タイプ。

有効な値

MIME タイプ。デフォルト: **binary/octet-stream**

必須

いいえ

x-amz-meta-<...>**詳細**

ユーザーのメタデータ。オブジェクトとともに保存されます。

有効な値

8kb までの文字列。デフォルトはありません。

必須

いいえ

x-amz-acl**詳細**

固定 ACL。

有効な値

private、**public-read**、**public-read-write**、**authenticated-read**

必須

いいえ

レスポンスエンティティ**InitiatedMultipartUploadsResult****詳細**

結果のコンテナ

型

Container

Bucket**詳細**

オブジェクトの内容を受け取るバケット。

型

String

キー

詳細

key リクエストパラメーターで指定されるキー (存在する場合)。

型

String

UploadId

詳細

upload-id 要求パラメーターで指定される ID で、マルチパートアップロードを特定します (存在する場合)。

型

String

3.5.22. S3 add a part to a multipart upload

マルチパートアップロードに部分を追加します。

複数パートのアップロードに部分を追加するために **uploadId** サブリソースとアップロード ID を指定します。

構文

```
PUT /BUCKET/OBJECT?partNumber=&uploadId=UPLOAD_ID HTTP/1.1
```

以下の HTTP レスポンスが返されます。

HTTP レスポンス

404

ステータスコード

NoSuchUpload

詳細

指定した upload-id がこのオブジェクトで開始されたアップロードと一致しません。

3.5.23. S3 list the parts of a multipart upload

マルチパートアップロードの一部を一覧表示するために **uploadId** サブリソースとアップロード ID を指定します。

構文

```
GET /BUCKET/OBJECT?uploadId=UPLOAD_ID HTTP/1.1
```

レスポンスエンティティ

InitiatedMultipartUploadsResult

詳細

結果のコンテナ

型

Container

Bucket

詳細

オブジェクトの内容を受け取るバケット。

型

String

キー

詳細

key リクエストパラメーターで指定されるキー (存在する場合)。

型

String

UploadId

詳細

upload-id 要求パラメーターで指定される ID で、マルチパートアップロードを特定します (存在する場合)。

型

String

イニシエーター

詳細

アップロードを開始したユーザーの **ID** と **DisplayName** が含まれます。

型

Container

ID

詳細

イニシエーターの ID。

型

String

DisplayName

詳細

イニシエーターの表示名。

型

String

Owner

詳細

アップロードしたオブジェクトを所有するユーザーの **ID** および **DisplayName** のコンテナ。

型

Container

StorageClass

詳細

作成されるオブジェクトを保存するために使用されるメソッド。**STANDARD** または **REDUCED_REDUNDANCY**

型

String

PartNumberMarker

詳細

IsTruncated が **true** の場合に後続のリクエストで使用する部分マーカー。一覧の先頭に指定します。

型

String

NextPartNumberMarker

詳細

IsTruncated が **true** の場合は、後続のリクエストで使用する次の部分マーカー。リストの末尾。

型

String

IsTruncated

詳細

true の場合は、オブジェクトのアップロードコンテンツのサブセットのみが返されます。

型

Boolean

部分

詳細

Key、**Part**、**InitiatorOwner**、**StorageClass**、および **Initiated** 要素のコンテナ。

型

Container

PartNumber

詳細

Key、**Part**、**InitiatorOwner**、**StorageClass**、および **Initiated** 要素のコンテナ。

型

Integer

ETag

詳細

コンポーネントのエンティティータグです。

型

String

サイズ**詳細**

アップロードした部分のサイズ。

型

Integer

3.5.24. S3 assemble the uploaded parts

アップロードした部分を組み立て、新規オブジェクトを作成します。これにより、複数パートのアップロードが実行されます。

複数パートからなるアップロードを完了するには、**uploadId** サブリソースとアップロード ID を指定します。

構文

POST /**BUCKET/OBJECT**?uploadId=**UPLOAD_ID** HTTP/1.1

リクエストエンティティ**CompleteMultipartUpload****詳細**

1つ以上の部分で設定されるコンテナ。

型

Container

必須

はい

部分**詳細**

PartNumber および **ETag** のコンテナ。

型

Container

必須

はい

PartNumber**詳細**

部分の識別子。

型

整数

必須

はい

ETag

詳細

コンポーネントのエンティティータグです。

型

String

必須

はい

レスポンスエンティティ

CompleteMultipartUploadResult

詳細

応答のコンテナ。

型

Container

場所

詳細

新規オブジェクトのリソース識別子 (パス)。

型

URI

bucket

詳細

新規オブジェクトが含まれるバケットの名前。

型

String

キー

詳細

オブジェクトのキー。

型

String

ETag

詳細

新規オブジェクトのエンティティータグ。

型

String

3.5.25. S3 copy a multipart upload

既存のオブジェクトからデータをデータソースとしてコピーして、パーツをアップロードします。

複数パートからなるアップロードコピーを実行するには、**uploadId** サブリソースとアップロード ID を指定します。

構文

```
PUT /BUCKET/OBJECT?partNumber=PartNumber&uploadId=UPLOAD_ID HTTP/1.1
Host: cname.domain.com
```

```
Authorization: AWS ACCESS_KEY:HASH_OF_HEADER_AND_SECRET
```

リクエストヘッダー

x-amz-copy-source

詳細

ソースバケット名およびオブジェクト名。

有効な値

BUCKET/OBJECT

必須

はい

x-amz-copy-source-range

詳細

ソースオブジェクトからコピーするバイトの範囲。

有効な値

範囲: **bytes=first-last** (ここで、最初および最後は、コピーするゼロベースのバイトオフセットです)たとえば、**bytes=0-9** は、ソースの最初の 10 バイトをコピーすることを示しています。

必須

いいえ

レスポンスエンティティ

CopyPartResult

詳細

すべてのレスポンス要素のコンテナ。

型

Container

ETag

詳細

新しい部分の ETag を返します。

型

String

LastModified**詳細**

最後に変更した日付を返します。

型

String

関連情報

- この機能の詳細は、[Amazon S3 のサイト](#) を参照してください。

3.5.26. S3 abort a multipart upload

複数パートアップロードを中止します。

マルチパートによるアップロードを中止するために **uploadId** サブリソースとアップロード ID を指定します。

構文

```
DELETE /BUCKET/OBJECT?uploadId=UPLOAD_ID HTTP/1.1
```

3.5.27. S3 Hadoop interoperability

HDFS (Hadoop Distributed File System) のアクセスを必要とするデータ解析アプリケーションは、Hadoop 用の Apache S3A コネクタを使用して Ceph Object Gateway にアクセスできます。S3A コネクタは、データが Ceph Object Gateway に保存される一方で、HDFS ファイルシステムがアプリケーションへのセマンティクスを読み取りおよび書き込みする S3 互換のオブジェクトストレージを HDFS ファイルシステムとして表示するオープンソースツールです。

Ceph Object Gateway は、Hadoop 2.7.3 に同梱される S3A コネクタと完全に互換性があります。

3.5.28. 関連情報

- マルチテナンシーの詳細は、[Red Hat Ceph Storage Object Gateway ガイド](#) を参照してください。

3.6. S3 選択操作 (テクノロジープレビュー)

開発者は、Spark-SQL などの高レベルの分析アプリケーションに S3 select API を使用して、レイテンシーとスループットを向上させることができます。たとえば、複数のギガバイトのデータを持つ CSV S3 オブジェクトの場合、ユーザーは以下のクエリーを使用して別の列でフィルターされる単一の列を抽出できます。

例

```
select customerid from s3Object where age>30 and age<65;
```

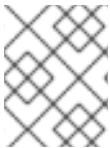
現時点で、S3 オブジェクトはデータのフィルターリングおよび抽出の前に、Ceph Object Gateway 経由で Ceph OSD からデータを取得する必要があります。オブジェクトのサイズが大きく、クエリーが具体的な場合に、パフォーマンスが向上します。

3.6.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。
- RESTful クライアント。
- ユーザーアクセスで作成された S3 ユーザー。

3.6.2. S3 select content from an object

select object content API は、構造化されたクエリー言語 (SQL) でオブジェクトの内容をフィルターします。リクエストでは、オブジェクトのコンマ区切りの値 (CSV) であるデータのシリアル化形式を指定して、指定のコンテンツを取得する必要があります。Amazon Web Services (AWS) のコマンドラインインターフェイス (CLI) 選択オブジェクトコンテンツは CSV 形式を使用してオブジェクトデータをレコードに解析し、クエリーで指定されたレコードのみを返します。



注記

応答のデータシリアル化形式を指定する必要があります。この操作には **s3:GetObject** パーミッションが必要です。

構文

```
POST /BUCKET/KEY?select&select-type=2 HTTP/1.1\r\n
```

例

```
POST /testbucket/sample1csv?select&select-type=2 HTTP/1.1\r\n
```

要求エンティティ

Bucket

詳細

オブジェクトコンテンツを選択するバケット。

型

String

必須

はい

キー

詳細

オブジェクトキー。

長さに関する制約

最小長は1です。

型

String

必須

はい

SelectObjectContentRequest

詳細

select オブジェクトコンテンツ要求パラメーターのルートレベルタグ。

型

String

必須

はい

式

詳細

オブジェクトのクエリーに使用される式。

型

String

必須

はい

ExpressionType

詳細

SQL など、提供された式のタイプ。

型

String

有効な値

SQL

必須

はい

InputSerialization

詳細

クエリーされるオブジェクトに含まれるデータの形式を記述します。

型

String

必須

はい

OutputSerialization

詳細

コンマセパレーターおよび改行で返されるデータの形式。

型

String

必須

はい

応答エンティティ

アクションに成功すると、サービスは **HTTP 200** 応答を返します。データは、サービスによって XML 形式で返されます。

Payload

詳細

ペイロードパラメーターのルートレベルタグ。

型

String

必須

はい

Records

詳細

レコードイベント。

型

base64 でエンコードされたバイナリーデータオブジェクト

必須

いいえ

Stats

詳細

stats イベント。

型

ロング

必須

いいえ

Ceph Object Gateway は以下の応答をサポートします。

例

```
{:event-type,records} {:content-type,application/octet-stream} :message-type,event}
```

構文

```
aws --endpoint-url http://localhost:80 s3api select-object-content
--bucket BUCKET_NAME
--expression-type 'SQL'
--input-serialization
'{"CSV": {"FieldDelimiter": ",", "QuoteCharacter": "\"", "RecordDelimiter": "\n",
"QuoteEscapeCharacter": "\\\" ", "FileHeaderInfo": "USE" }, "CompressionType": "NONE"}'
--output-serialization '{"CSV": {}}'
--key OBJECT_NAME
--expression "select count(0) from stdin where int(_1)<10;" output.csv
```

例

```
aws --endpoint-url http://localhost:80 s3api select-object-content
--bucket testbucket
--expression-type 'SQL'
--input-serialization
'{"CSV": {"FieldDelimiter": ",", "QuoteCharacter": "\"", "RecordDelimiter": "\n",
"QuoteEscapeCharacter": "\\\" ", "FileHeaderInfo": "USE" }, "CompressionType": "NONE"}'
--output-serialization '{"CSV": {}}'
--key testobject
--expression "select count(0) from stdin where int(_1)<10;" output.csv
```

サポートされる機能

現時点で、AWS s3 select コマンドの一部のみがサポートされます。

機能	詳細	詳細	例
算定演算子	$\wedge * \% / + - ()$		select (int(_1)+int(_2))*int(_9) from stdin;
算定演算子	% modulo		select count(*) from stdin where cast(_1 as int)%2 == 0;
算定演算子	\wedge power-of		select cast(2^10 as int) from stdin;
演算子の比較	$> < >= <= == !=$		select _1,_2 from stdin where (int(_1)+int(_3))>int(_5);
論理演算子	AND または NOT		select count(*) from stdin where not (int(1)>123 and int(_5)<200);
論理演算子	is null	式の null 表示の場合は true/false を返します。	
論理演算子および NULL	is not null	式の null 表示の場合は true/false を返します。	
論理演算子および NULL	不明な状態	null 処理を確認し、NULL で論理操作の結果を確認します。クエリーは 0 を返します。	select count(*) from stdin where null and (3>2);
NULL を使用した算術演算子	不明な状態	null 処理を確認し、NULL でバイナリー操作の結果を確認します。クエリーは 0 を返します。	select count(*) from stdin where (null+1) and (3>2);
NULL との比較	不明な状態	null 処理を確認し、比較操作の結果を NULL で確認します。クエリーは 0 を返します。	select count(*) from stdin where (null*1.5) != 3;
列がない	不明な状態		select count(*) from stdin where _1 is null;

機能	詳細	詳細	例
投影列	if、then、または else と同様です。	ケースの選択	when (1+1==(2+1)*3) then 'case_1' when 4*3==(12 then 'case_2' else 'case_else' end, age*2 from stdin;
論理演算子		coalesce は、最初の null 以外の引数を返します。	select coalesce(nullif(5,5),nullif(1,1.0),age+12) from stdin;
論理演算子		nullif の場合は、両方の引数が等しい場合は null を返し、それ以外の場合は最初の引数 nullif(1,1)=NULL nullif(null,1)=NULL nullif(2,1)=2 を返します。	select nullif(cast(_1 as int),cast(_2 as int)) from stdin;
論理演算子		{expression} in (.. {expression} ..)	select count(*) from s3object where 'ben' in (trim(_5),substring(_1,char_length(_1)-3,3),last_name);
論理演算子		{expression} between {expression} and {expression}	select count(*) from stdin where substring(_3,char_length(_3),1) between "x" and trim(_1) and substring(_3,char_length(_3)-1,1) == ":";
論理演算子		{expression} like {match-pattern}	select count() from stdin where first_name like '%de_'; select count() from stdin where _1 like "%a[r-s];
キャスト演算子			select cast(123 as int)%2 from stdin;
キャスト演算子			select cast(123.456 as float)%2 from stdin;
キャスト演算子			select cast('ABC0-9' as string),cast(substr('ab12cd',3,2) as int)*4 from stdin;
キャスト演算子			select cast(substring('publish on 2007-01-01',12,10) as timestamp) from stdin;

機能	詳細	詳細	例
AWS 以外の キャスト演算子			select int(_1),int(1.2 + 3.4) from stdin;
AWS 以外の キャスト演算子			select float(1.2) from stdin;
AWS 以外の キャスト演算子			select timestamp('1999:10:10- 12:23:44') from stdin;
集約機能	sum		select sum(int(_1)) from stdin;
集約機能	avg		select avg(cast(_1 a float) + cast(_2 as int)) from stdin;
集約機能	min		select avg(cast(_1 a float) + cast(_2 as int)) from stdin;
集約機能	max		select max(float(_1)),min(int(_5)) from stdin;
集約機能	count		select count(*) from stdin where (int(1)+int(_3))>int(_5);
タイムスタンプ 関数	extract		select count(*) from stdin where extract('year',timestamp(_2)) > 1950 and extract('year',timestamp(_1)) < 1960;
タイムスタンプ 関数	dateadd		select count(0) from stdin where datediff('year',timestamp(_1) ,dateadd('day',366,timestam p(_1))) == 1;
タイムスタンプ 関数	datediff		select count(0) from stdin where datediff('month',timestamp(_1),timestamp(_2))) == 2;

機能	詳細	詳細	例
タイムスタンプ関数	utcnow		select count(0) from stdin where datediff('hours',utcnow(),dateadd('day',1,utcnow())) == 24
文字列関数	substring		select count(0) from stdin where int(substring(_1,1,4))>1950 and int(substring(_1,1,4))<1960;
文字列関数	trim		select trim(' foobar ') from stdin;
文字列関数	trim		select trim(trailing from ' foobar ') from stdin;
文字列関数	trim		select trim(leading from ' foobar ') from stdin;
文字列関数	trim		select trim(both '12' from '1112211foobar22211122') from stdin;
文字列関数	lower または upper		select trim(both '12' from '1112211foobar22211122') from stdin;
文字列関数	char_length, character_length		select count(*) from stdin where char_length(_3)==3;
複雑なクエリー			select sum(cast(_1 as int)),max(cast(_3 as int)), substring('abcdefghijklm', (2-1)*3+sum(cast(_1 as int))/sum(cast(_1 as int))+1, (count() + count(0))/count(0)) from stdin;
エイリアスのサポート			select int(_1) as a1, int(_2) as a2 , (a1+a2) as a3 from stdin where a3>100 and a3<300;

関連情報

- 詳細は、[Amazon's S3 Select Object Content API](#) を参照してください。

3.6.3. S3 supported select functions

S3 select は、.Timestamp の機能をサポートします。

timestamp(string)

詳細

文字列をタイムスタンプの基本タイプに変換します。

サポート対象

現在、yyyy:mm:dd hh:mi:dd に変換します。

extract(date-part,timestamp)

詳細

入力タイムスタンプからの date-part の抽出に従って整数を返します。

サポート対象

date-part: year,month,week,day.

dateadd(date-part ,integer,timestamp)

詳細

入力されたタイムスタンプと date-part の結果に基づいて計算されたタイムスタンプを返します。

サポート対象

date-part : year,month,day.

datediff(date-part,timestamp,timestamp)

詳細

整数を返します。これは、date-part に応じた 2 つのタイムスタンプの差の計算結果です。

サポート対象

date-part : year,month,day,hours.

utcnow()

詳細

現在の時刻のタイムスタンプを返します。

集約

count()

詳細

(条件がある場合) 条件と一致する行数に基づいて整数を返します。

sum(expression)

詳細

(条件がある場合) 条件と一致する各行の式の概要を返します。

avg(expression)

詳細

(条件がある場合) 条件に一致する各行の平均式を返します。

max(expression)

詳細

(条件がある場合) 条件に一致するすべての式について最大結果を返します。

min(expression)

詳細

(条件がある場合) 条件に一致するすべての式の最小結果を返します。

String

substring(string,from,to)

詳細

from および input をもとに、入力文字列から抽出した文字列を返します。

Char_length

詳細

文字列の文字数を返します。Character_length も同じです。

Trim

詳細

ターゲット文字列から先頭または末尾の文字をトリミングします。デフォルトは空白です。

Upper\lower

詳細

文字を大文字または小文字に変換します。

NULL

NULL 値が見つからないか、または不明な値で、**NULL** が任意の演算に値を生成できません。同じことが算術比較にも当てはまります。**NULL** との比較は不明である **NULL** です。

表3.4 NULL ユースケース

A is NULL	Result(NULL=UNKNOWN)
Not A	NULL
A または alse	NULL
A or True	True
A or A	NULL

A is NULL	Result(NULL=UNKNOWN)
A and False	False
A and True	NULL
A and A	NULL

関連情報

- 詳細は、[Amazon's S3 Select Object Content API](#) を参照してください。

3.6.4. S3 alias programming construct

エイリアスプログラミング構築は、多くの列または複雑なクエリーを含むオブジェクトを持つプログラミングを容易にするため、s3 select 言語に不可欠な部分です。エイリアス構造を含むステートメントを解析すると、エイリアスを適切な投影列への参照に置き換え、クエリーの実行時に参照が他の式として評価されます。エイリアスは結果キャッシュを維持します。つまり、エイリアスが複数回使用された場合は、キャッシュからの結果が使用されるため、同じ式は評価されず、同じ結果が返されます。現在、Red Hat は列エイリアスをサポートしています。

例

```
select int(_1) as a1, int(_2) as a2 , (a1+a2) as a3 from s3object where a3>100 and a3<300;")
```

3.6.5. S3 CSV parsing explained

入力シリアルライゼーションを使用して CSV 定義をデフォルト値で定義できます。

- 行区切り文字には `{\n}` を使用します。
- 引用には `{"}` を使用します。
- エスケープ文字には `{\}` を使用します。

csv-header-info の解析は、これはスキーマを含む入力オブジェクトの最初の行になります。現在、シリアル化および圧縮タイプの出力はサポートされていません。S3 select エンジンには、S3-objects を解析する CSV パーサーがあります。

- 各行は、行区切り文字で終わります。
- フィールド区切り文字は、隣接する列を区切ります。
- 連続するフィールドの区切り文字は **NULL** 列を定義します。
- 引用符は、フィールド区切り文字をオーバーライドします。フィールド区切り文字であるは、引用符の間の任意の文字です。
- エスケープ文字は、行区切り文字以外の特殊文字を無効にします。

以下は、CSV 解析ルールの例です。

表3.5 CSV の解析

機能	詳細	入力 (トークン)
NULL	連続するフィールド区切り文字	<code>„1,,2, => {null}{null}{1}{null}{2}{null}</code>
QUOTE	引用符は、フィールドの区切り文字を上書きします。	<code>11,22,"a,b,c,d",last => {11}{22}{“a,b,c,d”}{last}</code>
Escape	エスケープ文字はメタ文字をオーバーライドします。	オブジェクトの所有者の ID および DisplayName のコンテナ。
row delimiter	クローズされた引用符はありません。行区切り文字は終了行になります。	<code>11,22,a="str,44,55,66 => {11}{22}{a="str,44,55,66}</code>
csv header info	FileHeaderInfo タグ	USE の値は、最初の行の各トークンが column-name であることを示します。IGNORE 値は最初の行をスキップすることを意味します。

関連情報

- 詳細は、[Amazon's S3 Select Object Content API](#) を参照してください。

3.7. 関連情報

- Amazon S3 の一般的なリクエストヘッダーは、[付録B S3 の一般的なリクエストヘッダー](#) を参照してください。
- Amazon S3 の一般的なレスポンスステータスコードは、[付録C S3 の一般的なレスポンスステータスコード](#) を参照してください。
- サポートされないヘッダーフィールドは [付録D S3 サポートされないヘッダーフィールド](#) を参照してください。

第4章 CEPH OBJECT GATEWAY および SWIFT API

開発者は、Swift API データアクセスモデルと互換性のある RESTful アプリケーションプログラミング インターフェイス (API) を使用できます。Ceph Object Gateway を使用して、Red Hat Ceph Storage クラスタに保存されているバケットおよびオブジェクトを管理できます。

以下の表は、現在の Swift 機能機能のサポート状況を示しています。

表4.1 機能

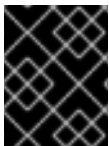
機能	状態	備考
認証	サポート対象	
アカウントメタデータの取得	サポート対象	カスタムメタデータなし
Swift ACL	サポート対象	Swift ACL のサブセットに対応
コンテナの一覧表示	サポート対象	
コンテナのオブジェクト一覧の表示	サポート対象	
コンテナの作成	サポート対象	
コンテナの削除	サポート対象	
コンテナメタデータの取得	サポート対象	
コンテナメタデータの追加/更新	サポート対象	
コンテナメタデータの削除	サポート対象	
オブジェクトの取得	サポート対象	
オブジェクトの作成/更新	サポート対象	
大規模オブジェクトの作成	サポート対象	
オブジェクトの削除	サポート対象	
オブジェクトのコピー	サポート対象	
オブジェクトメタデータの取得	サポート対象	
オブジェクトメタデータの追加/更新	サポート対象	
一時 URL 操作	サポート対象	

機能	状態	備考
CORS	サポート対象外	
オブジェクトの期限設定	サポート対象	
オブジェクトのバージョン管理	サポート対象外	
静的な Web サイト	サポート対象外	

4.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。
- RESTful クライアント。

4.2. SWIFT API の制限



重要

以下の制限事項を使用してください。お使いのハードウェアの選択には影響があるため、この要件を Red Hat アカウントチームと常に相談してください。

- Swift API を使用する場合の最大オブジェクトサイズ: 5GB
- Swift API を使用する場合のメタデータの最大サイズ: オブジェクトに適用できるユーザーメタデータの合計サイズに定義された制限はありませんが、単一の HTTP 要求は 16,000 バイトに制限されます。

4.3. SWIFT ユーザーの作成

Swift インターフェイスをテストするには、Swift サブユーザーを作成します。Swift ユーザーは、2つの手順で作成します。最初のステップでは、ユーザーを作成します。次のステップでは、秘密鍵を作成します。



注記

マルチサイトのデプロイメントでは、マスターゾーングループのマスターゾーンにあるホストでユーザーを作成します。

前提条件

- Ceph Object Gateway のインストール
- Ceph Object Gateway ノードへのルートレベルのアクセス

手順

- Swift ユーザーを作成します。

構文

```
radosgw-admin subuser create --uid=NAME --subuser=NAME:swift --access=full
```

NAME を Swift ユーザー名に置き換えます。以下に例を示します。

例

```
[root@host01 ~]# radosgw-admin subuser create --uid=testuser --subuser=testuser:swift --
access=full
{
  "user_id": "testuser",
  "display_name": "First User",
  "email": "",
  "suspended": 0,
  "max_buckets": 1000,
  "auid": 0,
  "subusers": [
    {
      "id": "testuser:swift",
      "permissions": "full-control"
    }
  ],
  "keys": [
    {
      "user": "testuser",
      "access_key": "O8JDE41XMI74O185EHKD",
      "secret_key": "i4Au2yxG5wtr1JK01mI8kjJPM93HNAoVWOSTdJd6"
    }
  ],
  "swift_keys": [
    {
      "user": "testuser:swift",
      "secret_key": "13TLtdEW7bCqgttQgPzxFxziu0AgabtOc6vM8DLA"
    }
  ],
  "caps": [],
  "op_mask": "read, write, delete",
  "default_placement": "",
  "placement_tags": [],
  "bucket_quota": {
    "enabled": false,
    "check_on_raw": false,
    "max_size": -1,
    "max_size_kb": 0,
    "max_objects": -1
  },
  "user_quota": {
    "enabled": false,
    "check_on_raw": false,
    "max_size": -1,
    "max_size_kb": 0,
    "max_objects": -1
  },
}
```



```

    "temp_url_keys": [],
    "type": "rgw"
  }

```

2. シークレットキーを作成します。

構文

```
radosgw-admin key create --subuser=NAME:swift --key-type=swift --gen-secret
```

NAME を Swift ユーザー名に置き換えます。以下に例を示します。

例

```

[root@host01 ~]# radosgw-admin key create --subuser=testuser:swift --key-type=swift --gen-secret
{
  "user_id": "testuser",
  "display_name": "First User",
  "email": "",
  "suspended": 0,
  "max_buckets": 1000,
  "auid": 0,
  "subusers": [
    {
      "id": "testuser:swift",
      "permissions": "full-control"
    }
  ],
  "keys": [
    {
      "user": "testuser",
      "access_key": "O8JDE41XMI74O185EHKD",
      "secret_key": "i4Au2yxG5wtr1JK01ml8kjJPM93HNAoVWOSTdJd6"
    }
  ],
  "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"
}

```

4.4. ユーザーの SWIFT 認証

ユーザーを認証するには、**X-Auth-User** および **X-Auth-Key** を含むリクエストを作成します。

構文

```

GET /auth HTTP/1.1
Host: swift.example.com
X-Auth-User: johndoe
X-Auth-Key: R7UUOLFDI2ZI9PRCQ53K

```

レスポンスの例

```

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: UOICCC8TahFKIWuv9DB09TWHF0nDjpPEIha0kAa
Content-Length: 0
Content-Type: text/plain; charset=UTF-8

```



注記

認証中に **X-Storage-Url** 値を使用して **GET** リクエストを実行すると、Ceph の Swift 互換サービスに関するデータを取得できます。

関連情報

- Swift リクエストヘッダーは [Red Hat Ceph Storage 開発者ガイド](#) を参照してください。
- Swift レスポンスヘッダーは [Red Hat Ceph Storage 開発者ガイド](#) を参照してください。

4.5. SWIFT コンテナー操作

開発者は、Ceph Object Gateway 経由で Swift アプリケーションのプログラミングインターフェイス (API) を使用してコンテナーの操作を行うことができます。コンテナーを一覧表示、作成、更新、および削除できます。コンテナーのメタデータを追加または更新できます。

4.5.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。
- RESTful クライアント。

4.5.2. Swift コンテナ操作

コンテナは、データオブジェクトを格納するメカニズムです。アカウントには多くのコンテナを持たせることができますが、コンテナ名は一意でなければなりません。この API により、クライアントはコンテナの作成、アクセス制御およびメタデータの設定、コンテナのコンテンツの取得、およびコンテナの削除を行うことができます。この API は特定のユーザーのアカウントの情報に関連するリクエストを行うため、コンテナのアクセス制御が意図的に公開されていない限り、つまり匿名のリクエストを許可しない限り、この API のすべてのリクエストを認証する必要があります。



注記

Amazon S3 API はバケットという用語を使用してデータコンテナを記述します。Swift API 内のバケットを参照すると、バケットという用語はコンテナという用語と同じものになります。

オブジェクトストレージの1つは、階層パスやディレクトリーをサポートしないことです。代わりに、各コンテナにオブジェクトがある1つ以上のコンテナで設定される1つのレベルをサポートします。RADOS Gateway の Swift 互換 API は、疑似階層コンテナの概念をサポートします。これは、オブジェクトの命名を使用してコンテナをエミュレートする手段で、ストレージシステムで実際には実装されません。たとえば、photos/buildings/empire-state.jpg のように、疑似階層名でオブジェクトに名前を付けることができますが、コンテナ名にスラッシュ (/) 文字を含めることはできません。



重要

バージョン付けされた Swift コンテナに大規模なオブジェクトをアップロードする場合は、**python-swiftclient** ユーティリティで **--leave-segments** オプションを使用します。**--leave-segments** を使用しないと、マニフェストファイルが上書きされます。したがって、既存のオブジェクトは上書きされ、データが失われることになります。

4.5.3. Swift でコンテナのアクセス制御リスト (ACL) の更新

ユーザーがコンテナを作成すると、ユーザーはデフォルトでコンテナへの読み取り/書き込みアクセスを持ちます。その他のユーザーがコンテナのコンテンツを読み取りしたり、コンテナに書き込むことを許可するには、ユーザーを明示的に有効にする必要があります。**X-Container-Read** または **X-Container-Write** に * を指定することもできます。これにより、すべてのユーザーがコンテナから読み取るか、またはコンテナへの書き込みが可能になります。* を設定すると、コンテナが公開されます。これにより、匿名ユーザーがコンテナから読み込むか、またはコンテナに書き込むことができます。

構文

```
POST /API_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
```

リクエストヘッダー

X-Container-Read

詳細

コンテナの読み取りパーミッションを持つユーザー ID。

型

ユーザー ID のコンマ区切りの文字列値。

必須

いいえ

X-Container-Write

詳細

コンテナの書き込みパーミッションを持つユーザー ID。

型

ユーザー ID のコンマ区切りの文字列値。

必須

いいえ

4.5.4. Swift 一覧コンテナ

API バージョンを指定し、アカウントは特定のユーザーアカウントのコンテナの一覧を返す **GET** リクエスト。リクエストは特定のユーザーのコンテナを返すため、リクエストには認証トークンが必要です。リクエストは匿名で行われません。

構文

```
GET /API_VERSION/ACCOUNT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

リクエストパラメーター

limit

詳細

結果の数を指定の値に制限します。

型

Integer

有効な値

該当なし

必須

はい

format

詳細

結果の数を指定の値に制限します。

型

Integer

有効な値

json または **xml**

必須

いいえ

marker**詳細**

マーカー値よりも大きな結果の一覧を返します。

型

String

有効な値

該当なし

必須

いいえ

応答にはコンテナの一覧が含まれるか、または **204** 応答コードで返されます。

レスポンスエンティティ**アカウント****詳細**

アカウント情報の一覧。

型

Container

コンテナ**詳細**

コンテナの一覧。

型

Container

name**詳細**

コンテナの名前。

型

String

bytes**詳細**

コンテナのサイズ。

型

Integer

4.5.5. Swift でコンテナオブジェクトの一覧表示

コンテナ内のオブジェクトを一覧表示するには、API バージョン、アカウント、およびコンテナの名前を使用して **GET** リクエストを行います。クエリーパラメーターを指定して完全なリストをフィルターリングしたり、パラメーターを除外してコンテナに保存されている最初の 10,000 オブジェクト名の一覧を返すこともできます。

構文

```
GET /API_VERSION/TENANT:CONTAINER HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

リクエストパラメーター

format

詳細

結果の数を指定の値に制限します。

型

Integer

有効な値

json または **xml**

必須

いいえ

prefix

詳細

結果を、指定した接頭辞で始まるオブジェクトに制限します。

型

String

有効な値

該当なし

必須

いいえ

marker

詳細

マーカー値よりも大きな結果の一覧を返します。

型

String

有効な値

該当なし

必須

いいえ

limit

詳細

結果の数を指定の値に制限します。

型

Integer

有効な値

0 - 10,000

必須

いいえ

delimiter

詳細

接頭辞と他のオブジェクト名の上に挿入される区切り文字。

型

String

有効な値

該当なし

必須

いいえ

path

詳細

オブジェクトの擬似階層パス。

型

String

有効な値

該当なし

必須

いいえ

レスポンスエンティティ

コンテナ

詳細

コンテナ

型

Container

object

詳細

コンテナ内のオブジェクト。

型

Container

name

詳細

コンテナ内のオブジェクトの名前。

型

String

ハッシュ

詳細

オブジェクトのコンテンツのハッシュコード。

型

String

last_modified

詳細

オブジェクトの内容を最後に変更した時間。

型

Date

content_type

詳細

オブジェクト内のコンテンツのタイプ。

型

String

4.5.6. Swift でコンテナの作成

新規コンテナを作成するには、API バージョン、アカウント、および新規コンテナの名前で **PUT** 要求を行います。コンテナ名は一意である必要があります。スラッシュ (/) を含めることはできず、256 バイト未満でなければなりません。リクエストには、アクセス制御ヘッダーおよびメタデータヘッダーを含めることができます。一連の配置プールのキーを特定するストレージポリシーを含めることもできます。たとえば、**radosgw-admin zone get** を実行すると、**placement_pools** で利用可能なキーの一覧を確認します。ストレージポリシーを使用すると、SSD ベースのストレージなど、コンテナの特別なプールセットを指定できます。操作には、べき等性があります。既存のコンテナを作成するように要求すると、HTTP 202 戻りコードが返されますが、別のコンテナは作成されません。

構文

```
PUT /API_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
```

ヘッダー

X-Container-Read

詳細

コンテナの読み取りパーミッションを持つユーザー ID。

型

ユーザー ID のコンマ区切りの文字列値。

必須

いいえ

X-Container-Write

詳細

コンテナの書き込みパーミッションを持つユーザー ID。

型

ユーザー ID のコンマ区切りの文字列値。

必須

いいえ

X-Container-Meta-KEY

詳細

任意の文字列の値を取得するユーザー定義のメタデータキー。

型

String

必須

いいえ

X-Storage-Policy

詳細

Ceph Object Gateway の **placement_pools** 下にあるストレージポリシーを識別するキー。**radosgw-admin zone get** を実行し、利用可能なキーを取得します。

型

String

必須

いいえ

同じ名前のコンテナがすでに存在し、ユーザーがコンテナ所有者である場合、操作は成功します。そうでないと、操作は失敗します。

HTTP レスポンス

409

ステータスコード

BucketAlreadyExists

詳細

コンテナは、別のユーザーの所有権にすでに存在します。

4.5.7. Swift コンテナの削除

コンテナを削除するには、API バージョン、アカウント、およびコンテナの名前を使用して **DELETE** 要求を行います。コンテナは空である必要があります。コンテナが空であるかを確認する場合は、コンテナに対して **HEAD** リクエストを実行します。コンテナが正常に削除されると、コンテナ名を再利用できます。

構文

```
DELETE /API_VERSION/ACCOUNT/TENANT:CONTAINER HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

HTTP レスポンス

204

ステータスコード

NoContent

詳細

コンテナが削除されました。

4.5.8. Swift がコンテナのメタデータを追加または更新

コンテナにメタデータを追加するには、API バージョン、アカウント、およびコンテナ名で **POST** 要求を行います。メタデータを追加または更新するには、コンテナに対する書き込み権限が必要です。

構文

```
POST /API_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
```

リクエストヘッダー

X-Container-Meta-KEY

詳細

任意の文字列の値を取得するユーザー定義のメタデータキー。

型

String

必須

いいえ

4.6. SWIFT オブジェクト操作

開発者は、Ceph Object Gateway 経由で Swift アプリケーションのプログラミングインターフェイス (API) を使用してオブジェクト操作を行うことができます。オブジェクトを一覧表示、作成、更新、および削除することができます。オブジェクトのメタデータを追加または更新することもできます。

4.6.1. 前提条件

- 稼働中の Red Hat Ceph Storage クラスタがある。

- RESTful クライアント。

4.6.2. Swift オブジェクト操作

オブジェクトは、データおよびメタデータを保存するコンテナです。コンテナには多くのオブジェクトがありますが、オブジェクト名は一意である必要があります。この API により、クライアントはオブジェクトの作成、アクセス制御およびメタデータの設定、オブジェクトのデータおよびメタデータの取得、およびオブジェクトの削除を行うことができます。この API は特定のユーザーのアカウントの情報に関連するリクエストを行うため、この API のすべてのリクエストを認証する必要があります。コンテナまたはオブジェクトのアクセス制御が意図的に公開されていない限り、つまり匿名の要求を許可している場合を除きます。

4.6.3. Swift がオブジェクトを取得

オブジェクトを取得するには、API バージョン、アカウント、コンテナ、およびオブジェクト名を使用して **GET** リクエストを行います。コンテナ内のオブジェクトを取得するには、コンテナの読み取り権限が必要です。

構文

```
GET /API_VERSION/ACCOUNT/TENANT:CONTAINER/OBJECT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

リクエストヘッダー

range

詳細

オブジェクトの内容のサブセットを取得するには、バイト範囲を指定します。

型

Date

必須

いいえ

If-Modified-Since

詳細

ソースオブジェクトの **last_modified** 属性の日時以降に変更された場合のみコピーします。

型

Date

必須

いいえ

If-Unmodified-Since

詳細

ソースオブジェクトの **last_modified** 属性の日時以降に変更した場合のみコピーします。

型

Date

必須

いいえ

Copy-If-Match**詳細**

リクエストの ETag がソースオブジェクトの ETag と一致する場合にのみコピーします。

型

ETag

必須

いいえ

Copy-If-None-Match**詳細**

リクエストの **ETag** がソースオブジェクトの ETag と一致しない場合にのみコピーします。

型

ETag

必須

いいえ

レスポンスヘッダー**Content-Range****詳細**

オブジェクトコンテンツのサブセットの範囲。range ヘッダーフィールドがリクエストで指定されている場合にのみ返されます。

4.6.4. Swift でオブジェクトの作成または更新

新規オブジェクトを作成するには、API バージョン、アカウント、コンテナ名、および新規オブジェクトの名前を使用して **PUT** 要求を行います。オブジェクトを作成または更新するには、コンテナに書き込みパーミッションが必要です。オブジェクト名は、コンテナ内で一意である必要があります。**PUT** リクエストはべき等ではないため、一意の名前を使用しないと、リクエストによりオブジェクトが更新されます。ただし、オブジェクト名に疑似階層構文を使用して、別の疑似階層ディレクトリーにある場合は、同じ名前の別のオブジェクトと区別することができます。リクエストには、アクセス制御ヘッダーおよびメタデータヘッダーを含めることができます。

構文

```
PUT /API_VERSION/ACCOUNT/TENANT:CONTAINER HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

リクエストヘッダー**ETag****詳細**

オブジェクトの内容の MD5 ハッシュ。推奨。

型

String

有効な値

該当なし

必須

いいえ

Content-Type**詳細**

オブジェクトの内容の MD5 ハッシュ。

型

String

有効な値

該当なし

必須

いいえ

Transfer-Encoding**説明**

オブジェクトが大規模な集約オブジェクトの一部であるかどうかを示します。

型

String

有効な値**chunked****必須**

いいえ

4.6.5. Swift でオブジェクトの削除

オブジェクトを削除するには、API バージョン、アカウント、コンテナ、およびオブジェクト名を使用して **DELETE** リクエストを行います。コンテナ内のオブジェクトを削除するには、コンテナに対する書き込み権限が必要です。オブジェクトが正常に削除されると、オブジェクト名を再利用できます。

構文

```
DELETE /API_VERSION/ACCOUNT/TENANT:CONTAINER/OBJECT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

4.6.6. Swift でオブジェクトのコピー

オブジェクトのコピーを使用すると、オブジェクトをダウンロードしたり、別のコンテナにアップロードしたりしなくてもよいように、オブジェクトのサーバー側のコピーを作成できます。あるオブジェクトのコンテンツを別のオブジェクトにコピーするには、API バージョン、アカウント、およびコンテナ名で **PUT** 要求または **COPY** 要求を行います。

PUT 要求の場合は、要求で宛先コンテナおよびオブジェクト名、および要求ヘッダーのソースコンテナおよびオブジェクトを使用します。

Copy リクエストには、要求でソースコンテナおよびオブジェクト、および要求ヘッダーの宛先コンテナおよびオブジェクトを使用します。オブジェクトをコピーするには、コンテナに書き込みパーミッションが必要です。宛先オブジェクト名は、コンテナ内で一意である必要があります。リクエストはべき等ではないため、一意の名前を使用しないと、リクエストにより宛先オブジェクトが更新されます。宛先オブジェクトが別の疑似階層ディレクトリーにある場合は、オブジェクト名に疑似階層構文を使用して、同じ名前前のソースオブジェクトと区別できます。リクエストには、アクセス制御ヘッダーおよびメタデータヘッダーを含めることができます。

構文

```
PUT /API_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
```

または、次のようになります。

構文

```
COPY /API_VERSION/ACCOUNT/TENANT:SOURCE_CONTAINER/SOURCE_OBJECT HTTP/1.1
Destination: TENANT:DEST_CONTAINER/DEST_OBJECT
```

リクエストヘッダー

X-Copy-From

詳細

ソースコンテナ/オブジェクトパスを定義するために **PUT** リクエストで使用されます。

型

String

必須

はい (**PUT** を使用している場合)

Destination

詳細

宛先コンテナ/オブジェクトパスを定義するために **COPY** 要求で使用されます。

型

String

必須

はい (**COPY** を使用している場合)

If-Modified-Since

詳細

ソースオブジェクトの **last_modified** 属性の日時以降に変更された場合のみコピーします。

型

Date

必須

いいえ

If-Unmodified-Since

詳細

ソースオブジェクトの **last_modified** 属性の日時以降に変更した場合のみコピーします。

型

Date

必須

いいえ

Copy-If-Match

詳細

リクエストの ETag がソースオブジェクトの ETag と一致する場合にのみコピーします。

型

ETag

必須

いいえ

Copy-If-None-Match

詳細

リクエストの **ETag** がソースオブジェクトの ETag と一致しない場合にのみコピーします。

型

ETag

必須

いいえ

4.6.7. Swift でオブジェクトメタデータの取得

オブジェクトのメタデータを取得するには、API バージョン、アカウント、コンテナ、およびオブジェクト名を使用して **HEAD** リクエストを行います。コンテナ内のオブジェクトからメタデータを取得するには、コンテナの読み取り権限が必要です。このリクエストは、オブジェクト自体の要求と同じヘッダー情報を返しますが、オブジェクトのデータを返しません。

構文

```
HEAD /API_VERSION/ACCOUNT/TENANT:CONTAINER/OBJECT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

4.6.8. Swift によるオブジェクトメタデータの追加または更新

オブジェクトにメタデータを追加するには、API バージョン、アカウント、コンテナ、およびオブジェクト名で **POST** リクエストを行います。メタデータを追加または更新するには、親コンテナに対する書き込み権限が必要です。

構文

```
POST /API_VERSION/ACCOUNT/TENANT:CONTAINER/OBJECT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

リクエストヘッダー

X-Object-Meta-KEY

詳細

任意の文字列の値を取得するユーザー定義のメタデータキー。

型

String

必須

いいえ

4.7. SWIFT の一時 URL 操作

一時的なアクセスを可能にするため、**radosgw** の swift エンドポイントによりサポートされます。たとえば、GET リクエストは、認証情報を共有せずにオブジェクトに送信されます。

この機能には、最初に **X-Account-Meta-Temp-URL-Key** の値を設定し、必要に応じて **X-Account-Meta-Temp-URL-Key-2** を設定する必要があります。Temp URL 機能は、これらの秘密鍵に対する HMAC-SHA1 署名に依存します。

4.7.1. Swift が一時 URL オブジェクトを取得

一時 URL は、以下の要素を含む暗号化 HMAC-SHA1 署名を使用します。

- Request メソッドの値 (例:GET)
- エポックからの経過時間 (秒単位)。つまり Unix 時間です。
- v1 以降のリクエストパス

上記の項目は、それらの間に新しい行が追加されて正規化され、HMAC は前述の Temp URL キーのいずれかに対して SHA-1 ハッシュアルゴリズムを使用して生成されます。

上記を示すサンプルの Python スクリプトを以下に示します。

例

```
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
```

出力例

```
https://objectstore.example.com/v1/your-bucket/your-object?
temp_url_sig=ff4657876227fc6025f04fcf1e82818266d022c6&temp_url_expires=1423200992
```

4.7.2. Swift POST 一時 URL キー

必要なキーを持つ swift アカウントへの **POST** リクエストは、一時 URL アクセスをアカウントに提供できるアカウントのシークレット一時 URL キーを設定します。最大 2 つのキーがサポートされ、一時 URL を無効化せずに鍵をローテーションできるように、両方のキーに対して署名がチェックされます。

構文

```
POST /API_VERSION/ACCOUNT HTTP/1.1
Host: FULLY_QUALIFIED_DOMAIN_NAME
X-Auth-Token: AUTH_TOKEN
```

リクエストヘッダー

X-Account-Meta-Temp-URL-Key

詳細

任意の文字列値を取るユーザー定義のキー。

型

String

必須

はい

X-Account-Meta-Temp-URL-Key-2

詳細

任意の文字列値を取るユーザー定義のキー。

型

String

必須

いいえ

4.8. SWIFT マルチテナンシーコンテナの操作

クライアントアプリケーションがコンテナにアクセスする場合は、常に特定ユーザーの認証情報で動作します。Red Hat Ceph Storage クラスターでは、すべてのユーザーがテナントに属します。そのため、テナントが明示的に指定されていない場合、すべてのコンテナ操作のコンテキストに暗黙的なテ

ナントがあります。したがって、マルチテナンシーは、参照されるコンテナと、参照しているユーザーが同じテナントに属する限り、以前のリリースと完全に後方互換性があります。

明示的なテナントの指定に使用される拡張機能は、使用されるプロトコルおよび認証システムによって異なります。

テナントとコンテナはコロンで区切ります。したがって、URL は以下のようになります。

例

```
https://rgw.domain.com/tenant:container
```

一方、**create_container()** メソッドでは、コンテナメソッド自体でテナントとコンテナを分離します。

例

```
create_container("tenant:container")
```

4.9. 関連情報

- マルチテナンシーの詳細は、[Red Hat Ceph Storage ObjectGateway ガイド](#)を参照してください。
- Swift リクエストヘッダーは、[付録E Swift リクエストヘッダー](#)を参照してください。
- Swift レスポンスヘッダーは、[付録F Swift レスポンスヘッダー](#)を参照してください。

付録A CEPH RESTFUL API 仕様

ストレージ管理者は、Ceph RESTful API エンドポイントを通じてさまざまな Ceph サブシステムにアクセスできます。これは、利用可能な Ceph RESTful API メソッドの参照ガイドです。

利用可能な Ceph API エンドポイント:

- [「Ceph の概要」](#)
- [「認証」](#)
- [「Ceph ファイルシステム」](#)
- [「ストレージクラスターの設定」](#)
- [「CRUSH ルール」](#)
- [「イレイジャーコードプロファイル」](#)
- [「機能トグル」](#)
- [「Grafana」](#)
- [「ストレージクラスターの正常性」](#)
- [「ホスト」](#)
- [「iSCSI」](#)
- [「ログ」](#)
- [「Ceph Manager モジュール」](#)
- [「Ceph Monitor」](#)
- [「Ceph OSD」](#)
- [「Ceph Object Gateway」](#)
- [「ロールを操作する REST API」](#)
- [「NFS Ganesha」](#)
- [「Ceph Orchestrator」](#)
- [「Pools」](#)
- [「Prometheus」](#)
- [「RADOS ブロックデバイス」](#)
- [「パフォーマンスカウンター」](#)
- [「ロール」](#)
- [「サービス」](#)
- [「設定」](#)

- [「Ceph タスク」](#)
- [「テレメトリー」](#)
- [「Ceph ユーザー」](#)

A.1. 前提条件

- RESTful API の使用方法を理解します。
- 正常かつ実行中の Red Hat Ceph Storage クラスタ
- Ceph Manager の **dashboard** モジュールが有効化されている。

A.2. CEPH の概要

Ceph RESTful API **summary** エンドポイントを使用するためのメソッド参照。Ceph のサマリーの詳細を表示することができます。

GET /api/summary

詳細

Ceph の詳細の概要を表示します。

例

```
GET /api/summary HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.3. 認証

Ceph RESTful API **auth** エンドポイントを使用するメソッド参照。Red Hat Ceph Storage でセッションを開始します。

POST /api/auth

Curl の例

```
curl -i -k --location -X POST 'https://192.168.0.44:8443/api/auth' -H 'Accept: application/vnd.ceph.api.v1.0+json' -H 'Content-Type: application/json' --data '{"password": "admin@123", "username": "admin"}'
```

例

```
POST /api/auth HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "password": "STRING",
  "username": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/auth/check

詳細

認証トークンの要件を確認します。

例

```
POST /api/auth/check?token=STRING HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "token": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。

- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/auth/logout

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.4. CEPH ファイルシステム

Ceph RESTful API **cephfs** エンドポイントを使用して Ceph File Systems (CephFS) を管理するためのメソッド参照。

GET /api/cephfs

例

```
GET /api/cephfs HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。

- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/cephfs/FS_ID

パラメーター

- **FS_ID** は、Ceph File System の ID 文字列に置き換えます。

例

```
GET /api/cephfs/FS_ID HTTP/1.1  
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/cephfs/FS_ID/client/CLIENT_ID

パラメーター

- **FS_ID** は、Ceph File System の ID 文字列に置き換えます。
- **CLIENT_ID** は、Ceph クライアント識別子の文字列に置き換えます。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/cephfs/FS_ID/clients

パラメーター

- **FS_ID** は、Ceph File System の ID 文字列に置き換えます。

例

```
GET /api/cephfs/FS_ID/clients HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/cephfs/FS_ID/get_root_directory

詳細

ls_dir API 呼び出しを使用して取得できない root ディレクトリー。

パラメーター

- **FS_ID** は、Ceph File System の ID 文字列に置き換えます。

例

```
GET /api/cephfs/FS_ID/get_root_directory HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/cephfs/FS_ID/ls_dir

詳細

指定のパスのディレクトリーを一覧表示します。

パラメーター

- **FS_ID** は、Ceph File System の ID 文字列に置き換えます。
- クエリー:
 - **path**: リストを開始する文字列の値。デフォルトのパスは、指定されていない場合は / になります。
 - **depth**: ディレクトリーツリーを下るステップ数を指定する整数値。

例

```
GET /api/cephfs/FS_ID/ls_dir HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/cephfs/**FS_ID**/mds_counters

パラメーター

- **FS_ID** は、Ceph File System の ID 文字列に置き換えます。
- クエリー:
 - **カウンター**: 整数値。

例

```
GET /api/cephfs/FS_ID/mds_counters HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。

- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/cephfs/FS_ID/quota

詳細

指定されたパスの CephFS クォータを表示します。

パラメーター

- **FS_ID** は、Ceph File System の ID 文字列に置き換えます。
- クエリー:
 - **path**: ディレクトリーパスを指定する必須文字列の値。

例

```
GET /api/cephfs/FS_ID/quota?path=STRING HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

PUT /api/cephfs/FS_ID/quota

詳細

指定されたパスのクォータを設定します。

パラメーター

- **FS_ID** は、Ceph File System の ID 文字列に置き換えます。
- **max_bytes**: バイト制限を定義する文字列の値。
- **max_files**: ファイル制限を定義する文字列の値。
- **path**: ディレクトリーまたはファイルへのパスを定義する文字列値。

例

```
PUT /api/cephfs/FS_ID/quota HTTP/1.1
Host: example.com
```

```
Content-Type: application/json
```

```
{
  "max_bytes": "STRING",
  "max_files": "STRING",
  "path": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing, check the task queue.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

DELETE /api/cephfs/FS_ID/snapshot

詳細

スナップショットを削除します。

パラメーター

- **FS_ID** は、Ceph File System の ID 文字列に置き換えます。
- クエリー:
 - **name**: スナップショット名を指定する必須文字列の値。
 - **path**: ディレクトリーへのパスを定義する必須の文字列値。

ステータスコード

- 202 Accepted – Operation is still executing, check the task queue.
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/cephfs/FS_ID/snapshot

詳細

スナップショットを作成します。

パラメーター

- **FS_ID** は、Ceph File System の ID 文字列に置き換えます。
- **name**: スナップショット名を指定する文字列の値。名前が指定されていない場合は、RFC3339 UTC 形式の現在の時間を使用して名前が生成されます。
- **path**: ディレクトリーへのパスを定義する文字列の値です。

例

```
POST /api/cephfs/FS_ID/snapshot HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "name": "STRING",
  "path": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing, check the task queue.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/cephfs/FS_ID/tree

詳細

ディレクトリーを削除します。

パラメーター

- **FS_ID** は、Ceph File System の ID 文字列に置き換えます。
- クエリー:
 - **path**: ディレクトリーへのパスを定義する必須の文字列値。

ステータスコード

- 202 Accepted – Operation is still executing, check the task queue.
- 204 No Content – Resource deleted.

- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/cephfs/FS_ID/tree

詳細

ディレクトリーを作成します。

パラメーター

- **FS_ID** は、Ceph File System の ID 文字列に置き換えます。
- **path**: ディレクトリーへのパスを定義する文字列の値です。

例

```
POST /api/cephfs/FS_ID/tree HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "path": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing, check the task queue.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.5. ストレージクラスターの設定

Ceph RESTful API **cluster_conf** エンドポイントを使用して Red Hat Ceph Storage クラスターを管理するためのメソッド参照。

GET /api/cluster_conf

例

```
GET /api/cluster_conf HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/cluster_conf

例

```
POST /api/cluster_conf HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "name": "STRING",
  "value": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing, check the task queue.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

PUT /api/cluster_conf

例

```
PUT /api/cluster_conf HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "options": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing, check the task queue.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/cluster_conf/filter

詳細

名前でストレージクラスター設定を表示します。

パラメーター

- クエリー:
 - **names:** 設定オプション名の文字列値。

例

```
GET /api/cluster_conf/filter HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/cluster_conf/NAME

パラメーター

- **NAME** をストレージクラスター設定名に置き換えます。
- クエリー:
 - **section**: 必要な文字列値。

ステータスコード

- 202 Accepted – Operation is still executing, check the task queue.
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/cluster_conf/NAME

パラメーター

- **NAME** をストレージクラスター設定名に置き換えます。

例

```
GET /api/cluster_conf/NAME HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.6. CRUSH ルール

Ceph RESTful API **crush_rule** エンドポイントを使用して CRUSH ルールを管理するメソッド参照。

GET /api/crush_rule

詳細

CRUSH ルール設定を一覧表示します。

例

```
GET /api/crush_rule HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/crush_rule

例

```
POST /api/crush_rule HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "device_class": "STRING",
  "failure_domain": "STRING",
  "name": "STRING",
  "root": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing, check the task queue.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。

- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/crush_rule/NAME

パラメーター

- **NAME** は、ルール名に置き換えます。

ステータスコード

- 202 Accepted – Operation is still executing, check the task queue.
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/crush_rule/NAME

パラメーター

- **NAME** は、ルール名に置き換えます。

例

```
GET /api/crush_rule/NAME HTTP/1.1
Host: example.com
```

ステータスコード

- 202 Accepted – Operation is still executing, check the task queue.
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.7. イレイジャーコードプロファイル

Ceph RESTful API の **erasure_code_profile** エンドポイントを使用するメソッド参照を使用して、イレイジャーコーディングのプロファイルを管理します。

GET /api/erasure_code_profile

詳細

イレイジャーコーディングされたプロファイル情報を一覧表示します。

例

```
GET /api/erasure_code_profile HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/erasure_code_profile

例

```
POST /api/erasure_code_profile HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "name": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing, check the task queue.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。

- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/erasure_code_profile/NAME

パラメーター

- **NAME** は、プロファイル名に置き換えます。

ステータスコード

- 202 Accepted – Operation is still executing, check the task queue.
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/erasure_code_profile/NAME

パラメーター

- **NAME** は、プロファイル名に置き換えます。

例

```
GET /api/erasure_code_profile/NAME HTTP/1.1
Host: example.com
```

ステータスコード

- 202 Accepted – Operation is still executing, check the task queue.
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.8. 機能トグル

CRUSH ルールを管理する Ceph RESTful API **feature_toggles** エンドポイントを使用するメソッド参照。

GET /api/feature_toggles

詳細

Red Hat Ceph Storage の機能を一覧表示します。

例

```
GET /api/feature_toggles HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.9. GRAFANA

Ceph RESTful API **grafana** エンドポイントを使用して Grafana を管理するためのメソッド参照。

POST /api/grafana/dashboards

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。

- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/grafana/url

詳細

Grafana URL インスタンスを一覧表示します。

例

```
GET /api/grafana/url HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/grafana/validation/PARAMS

パラメーター

- **PARAMS** を文字列値に置き換えます。

例

```
GET /api/grafana/validation/PARAMS HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.10. ストレージクラスターの正常性

Ceph RESTful API の **health** エンドポイントを使用するメソッド参照。ストレージクラスターの正常性の詳細およびステータスを表示します。

GET /api/health/full

例

```
GET /api/health/full HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

GET /api/health/minimal

詳細

ストレージクラスターの最小限の正常性レポートを表示します。

例

```
GET /api/health/minimal HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.11. ホスト

Ceph RESTful API **host** エンドポイントを使用して、ノードとも呼ばれるホストの情報を表示するためのメソッド参照。

GET /api/host

詳細

ホストの仕様を一覧表示します。

パラメーター

- クエリー:
 - **sources**: ホストソースの文字列値。

例

```
GET /api/host HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/host

例

```
POST /api/host HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "hostname": "STRING",
  "status": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

DELETE /api/host/HOST_NAME

パラメーター

- **HOST_NAME** は、ノード名に置き換えます。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

GET /api/host/HOST_NAME

詳細

指定したホストの情報を表示します。

パラメーター

- **HOST_NAME** は、ノード名に置き換えます。

例

```
GET /api/host/HOST_NAME HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。

- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

PUT /api/host/HOST_NAME

詳細

指定したホストの情報を更新します。この方法は、Ceph Orchestrator が有効な場合にのみサポートされます。

パラメーター

- **HOST_NAME** は、ノード名に置き換えます。
- **force**: ホストがメンテナンスモードに強制します。
- **labels**: ラベルのリスト。
- **maintenance**: メンテナンスモードを入力するか、または終了します。
- **update_labels**: ラベルを更新します。

例

```
PUT /api/host/HOST_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "force": true,
  "labels": [
    "STRING"
  ],
  "maintenance": true,
  "update_labels": true
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/host/HOST_NAME/daemons

パラメーター

- **HOST_NAME** は、ノード名に置き換えます。

例

```
GET /api/host/HOST_NAME/daemons HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

GET /api/host/**HOST_NAME**/devices

パラメーター

- **HOST_NAME** は、ノード名に置き換えます。

例

```
GET /api/host/HOST_NAME/devices HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/host/**HOST_NAME**/identify_device

詳細

指定された秒数の間デバイスのライトをオンにして、デバイスを識別します。

パラメーター

- **HOST_NAME** は、ノード名に置き換えます。
- **device**: **/dev/dm-0**、**ABC1234DEF567-1R1234_ABC8DE0Q** などのデバイス ID。
- **期間** - デバイスの LED がフラッシュする秒数。

例

```
POST /api/host/HOST_NAME/identify_device HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "device": "STRING",
  "duration": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/host/**HOST_NAME**/inventory

詳細

ホストのインベントリを表示します。

パラメーター

- **HOST_NAME** は、ノード名に置き換えます。
- クエリー:
 - **refresh**: 非同期の更新をトリガーする文字列の値。

例

```
GET /api/host/HOST_NAME/inventory HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.

- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

GET /api/host/HOST_NAME/smart

パラメーター

- **HOST_NAME** は、ノード名に置き換えます。

例

```
GET /api/host/HOST_NAME/smart HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.12. iSCSI

Ceph RESTful API の **iscsi** エンドポイントを使用して iSCSI を管理するメソッド参照。

GET /api/iscsi/discoveryauth

詳細

iSCSI 検出認証の詳細を表示します。

例

```
GET /api/iscsi/discoveryauth HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

PUT /api/iscsi/discoveryauth

詳細

iSCSI 検出認証を設定します。

パラメーター

- クエリー:
 - **user**: 必要なユーザー名の文字列。
 - **password**: 必要なパスワード文字列。
 - **mutual_user**: 必要な相互ユーザー名の文字列。
 - **mutual_password**: 必要な相互パスワード文字列。

例

```
PUT /api/iscsi/discoveryauth?
user=STRING&password=STRING&mutual_user=STRING&mutual_password=STRING
HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "mutual_password": "STRING",
  "mutual_user": "STRING",
  "password": "STRING",
  "user": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。

- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

GET /api/iscsi/target

例

```
GET /api/iscsi/target HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/iscsi/target

例

```
POST /api/iscsi/target HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "acl_enabled": "STRING",
  "auth": "STRING",
  "clients": "STRING",
  "disks": "STRING",
  "groups": "STRING",
  "portals": "STRING",
  "target_controls": "STRING",
  "target_iqn": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。

- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/iscsi/target/TARGET_IQN

パラメーター

- **TARGET_IQN** をパス文字列に置き換えます。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/iscsi/target/TARGET_IQN

パラメーター

- **TARGET_IQN** をパス文字列に置き換えます。

例

```
GET /api/iscsi/target/TARGET_IQN HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

PUT /api/iscsi/target/TARGET_IQN

パラメーター

- **TARGET_IQN** をパス文字列に置き換えます。

例

```
PUT /api/iscsi/target/TARGET_IQN HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "acl_enabled": "STRING",
  "auth": "STRING",
  "clients": "STRING",
  "disks": "STRING",
  "groups": "STRING",
  "new_target_iqn": "STRING",
  "portals": "STRING",
  "target_controls": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.13. ログ

Ceph RESTful API の **logs** エンドポイントを使用してログ情報を表示するメソッド参照。

GET /api/logs/all

詳細

すべてのログ設定を表示します。

例

```
GET /api/logs/all HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.14. CEPH MANAGER モジュール

Ceph RESTful API の **mgr/module** エンドポイントを使用して Ceph Manager モジュールを管理するメソッド参照。

GET /api/mgr/module

詳細

管理モジュールの一覧を表示します。

例

```
GET /api/mgr/module HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/mgr/module/MODULE_NAME

詳細

永続設定の値を取得します。

パラメーター

- **MODULE_NAME** は Ceph Manager モジュール名に置き換えます。

例

```
GET /api/mgr/module/MODULE_NAME HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

PUT /api/mgr/module/**MODULE_NAME**

詳細

永続設定の値を設定します。

パラメーター

- **MODULE_NAME** は Ceph Manager モジュール名に置き換えます。
- **config**: モジュールオプションの値。

例

```
PUT /api/mgr/module/MODULE_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "config": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/mgr/module/MODULE_NAME/disable

詳細

指定の Ceph Manager モジュールを無効にします。

パラメーター

- **MODULE_NAME** は Ceph Manager モジュール名に置き換えます。

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/mgr/module/MODULE_NAME/enable

詳細

指定の Ceph Manager モジュールを有効にします。

パラメーター

- **MODULE_NAME** は Ceph Manager モジュール名に置き換えます。

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/mgr/module/MODULE_NAME/options

詳細

指定の Ceph Manager モジュールのオプションを表示します。

パラメーター

- **MODULE_NAME** は Ceph Manager モジュール名に置き換えます。

例

```
GET /api/mgr/module/MODULE_NAME/options HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.15. CEPH MONITOR

Ceph Monitor の情報を表示する Ceph RESTful API の **monitor** エンドポイントを使用するメソッド参照。

GET /api/monitor

詳細

Ceph Monitor の詳細を表示します。

例

```
GET /api/monitor HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.16. CEPH OSD

Ceph RESTful API の **osd** エンドポイントを使用して Ceph OSD を管理するメソッド参照。

GET /api/osd

例

```
GET /api/osd HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/osd

例

```
POST /api/osd HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "data": "STRING",
  "method": "STRING",
  "tracking_id": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。

- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/osd/flags

詳細

Ceph OSD フラグを表示します。

例

```
GET /api/osd/flags HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

PUT /api/osd/flags

詳細

ストレージクラスター全体の Ceph OSD フラグを設定します。

パラメーター

- **recovery_deletes**、**sortbitwise**、および **pglog_hardlimit** フラグの設定を解除することはできません。
- **purged_snapshots** フラグを設定できません。



重要

正常な操作には、以下の 4 つのフラグを追加する必要があります。

例

```
PUT /api/osd/flags HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "flags": [
    "STRING"
  ]
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

GET /api/osd/flags/individual

詳細

個別の Ceph OSD フラグを表示します。

例

```
GET /api/osd/flags/individual HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

PUT /api/osd/flags/individual

詳細

Ceph OSD の個別サブセットの **noout**、**noin**、**nodown**、および **noup** フラグを更新します。

例

```
PUT /api/osd/flags/individual HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "flags": {
    "nodown": true,
    "noin": true,
    "noout": true,
```



```

    "noup": true
  },
  "ids": [
    1
  ]
}

```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

GET /api/osd/safe_to_delete

パラメーター

- クエリー:
 - **svc_ids**: Ceph OSD サービス識別子に必要な文字列。

例

```

GET /api/osd/safe_to_delete?svc_ids=STRING HTTP/1.1
Host: example.com

```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

GET /api/osd/safe_to_destroy

詳細

Ceph OSD が破棄しても安全かどうかを確認します。

パラメーター

- クエリー:
 - **ID**: Ceph OSD サービス識別子の必要な文字列。

例

```
GET /api/osd/safe_to_destroy?ids=STRING HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

DELETE /api/osd/SVC_ID

パラメーター

- **SVC_ID** は、Ceph OSD サービス識別子の文字列値に置き換えます。
- クエリー:
 - **preserve_id**: 文字列の値。
 - **force**: 文字列の値。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

GET /api/osd/SVC_ID

詳細

Ceph OSD に関する収集したデータを返します。

パラメーター

- **SVC_ID** は、Ceph OSD サービス識別子の文字列値に置き換えます。

例

```
GET /api/osd/SVC_ID HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

PUT /api/osd/SVC_ID

パラメーター

- **SVC_ID** は、Ceph OSD サービス識別子の文字列値に置き換えます。

例

```
PUT /api/osd/SVC_ID HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "device_class": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/osd/SVC_ID/destroy

詳細

Ceph OSD に破棄されているとマークします。Ceph OSD は、破棄される前にダウンとマークする必要があります。この操作は Ceph OSD 識別子をそのまま保持しますが、Cephx キー、設定キーデータ、および lockbox キーを削除します。



警告

この操作により、データが永続的に読み取り不能になります。

パラメーター

- **SVC_ID** は、Ceph OSD サービス識別子の文字列値に置き換えます。

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

GET /api/osd/SVC_ID/devices

パラメーター

- **SVC_ID** は、Ceph OSD サービス識別子の文字列値に置き換えます。

例

```
GET /api/osd/SVC_ID/devices HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。

- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/osd/SVC_ID/histogram

詳細

Ceph OSD のヒストグラムデータを返します。

パラメーター

- **SVC_ID** は、Ceph OSD サービス識別子の文字列値に置き換えます。

例

```
GET /api/osd/SVC_ID/histogram HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

PUT /api/osd/SVC_ID/mark

詳細

Ceph OSD を **out**、**in**、**down**、および **lost** にマークします。



注記

Ceph OSD は、**lost** にする前に **down** とマークする必要があります。

パラメーター

- **SVC_ID** は、Ceph OSD サービス識別子の文字列値に置き換えます。

例

```
PUT /api/osd/SVC_ID/mark HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "action": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/osd/SVC_ID/purge

説明

CRUSH マップから Ceph OSD を削除します。



注記

Ceph OSD は、削除前に **down** とマークする必要があります。

パラメーター

- **SVC_ID** は、Ceph OSD サービス識別子の文字列値に置き換えます。

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/osd/SVC_ID/reweight

詳細

Ceph OSD の重みを一時的に変更します。Ceph OSD に **out** とマークが付けられると、OSD の重みは **0** に設定されます。Ceph OSD のマークが **in** に戻ると、OSD の重みは **1** に設定されます。

パラメーター

- **SVC_ID** は、Ceph OSD サービス識別子の文字列値に置き換えます。

例

```
POST /api/osd/SVC_ID/reweight HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "weight": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/osd/SVC_ID/scrub

パラメーター

- **SVC_ID** は、Ceph OSD サービス識別子の文字列値に置き換えます。
- クエリー:
 - **deep**: ブール値です。**true** または **false** になります。

例

```
POST /api/osd/SVC_ID/scrub HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "deep": true
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/osd/SVC_ID/smart

パラメーター

- **SVC_ID** は、Ceph OSD サービス識別子の文字列値に置き換えます。

例

```
GET /api/osd/SVC_ID/smart HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.17. CEPH OBJECT GATEWAY

Ceph RESTful API の **rgw** エンドポイントを使用して Ceph Object Gateway を管理するメソッド参照。

GET /api/rgw/status

詳細

Ceph Object Gateway のステータスを表示します。

例

GET /api/rgw/status HTTP/1.1
Host: example.com

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/rgw/daemon

詳細

Ceph Object Gateway デーモンを表示します。

例

GET /api/rgw/daemon HTTP/1.1
Host: example.com

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/rgw/daemon/SVC_ID

パラメーター

- **SVC_ID** は、サービス識別子を文字列値で置き換えます。

例

GET /api/rgw/daemon/**SVC_ID** HTTP/1.1
Host: example.com

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/rgw/site

パラメーター

- クエリー:
 - **query**: 文字列値。
 - **daemon_name**: デーモンの名前 (文字列値) を指定します。

例

```
GET /api/rgw/site HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

バケット管理

GET /api/rgw/bucket

パラメーター

- クエリー:
 - **stats**: バケット統計のブール値。
 - **daemon_name**: デーモンの名前 (文字列値) を指定します。

例

```
GET /api/rgw/bucket HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/rgw/bucket

例

```
POST /api/rgw/bucket HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "bucket": "STRING",
  "daemon_name": "STRING",
  "lock_enabled": "false",
  "lock_mode": "STRING",
  "lock_retention_period_days": "STRING",
  "lock_retention_period_years": "STRING",
  "placement_target": "STRING",
  "uid": "STRING",
  "zonegroup": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/rgw/bucket/BUCKET

パラメーター

- **BUCKET** は、バケット名 (文字列値) に置き換えます。
- クエリー:
 - **purge_objects**: 文字列の値。
 - **daemon_name**: デーモンの名前 (文字列値) を指定します。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/rgw/bucket/BUCKET

パラメーター

- **BUCKET** は、バケット名 (文字列値) に置き換えます。
- クエリー:
 - **daemon_name**: デーモンの名前 (文字列値) を指定します。

例

```
GET /api/rgw/bucket/BUCKET HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

PUT /api/rgw/bucket/BUCKET

パラメーター

- **BUCKET** は、バケット名 (文字列値) に置き換えます。

例

```
PUT /api/rgw/bucket/BUCKET HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "bucket_id": "STRING",
  "daemon_name": "STRING",
  "lock_mode": "STRING",
  "lock_retention_period_days": "STRING",
  "lock_retention_period_years": "STRING",
  "mfa_delete": "STRING",
  "mfa_token_pin": "STRING",
  "mfa_token_serial": "STRING",
  "uid": "STRING",
  "versioning_state": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

ユーザー管理

GET /api/rgw/user

詳細

Ceph Object Gateway ユーザーを表示します。

パラメーター

- クエリー:
 - **daemon_name**: デーモンの名前 (文字列値) を指定します。

例

■

```
GET /api/rgw/user HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/rgw/user

例

```
POST /api/rgw/user HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "access_key": "STRING",
  "daemon_name": "STRING",
  "display_name": "STRING",
  "email": "STRING",
  "generate_key": "STRING",
  "max_buckets": "STRING",
  "secret_key": "STRING",
  "suspended": "STRING",
  "uid": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

GET /api/rgw/user/get_emails

パラメーター

- クエリー:
 - daemon_name**: デーモンの名前 (文字列値) を指定します。

例

```
GET /api/rgw/user/get_emails HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/rgw/user/UID

パラメーター

- UID** を文字列としてユーザー ID に置き換えます。
- クエリー:
 - daemon_name**: デーモンの名前 (文字列値) を指定します。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/rgw/user/UID

パラメーター

- UID** を文字列としてユーザー ID に置き換えます。

- クエリー:
 - **daemon_name**: デーモンの名前 (文字列値) を指定します。
 - **stats**: ユーザー統計のブール値。

例

```
GET /api/rgw/user/UID HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

PUT /api/rgw/user/**UID**

パラメーター

- **UID** を文字列としてユーザー ID に置き換えます。

例

```
PUT /api/rgw/user/UID HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "daemon_name": "STRING",
  "display_name": "STRING",
  "email": "STRING",
  "max_buckets": "STRING",
  "suspended": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。

- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/rgw/user/UID/capability

パラメーター

- **UID** を文字列としてユーザー ID に置き換えます。
- クエリー:
 - **daemon_name**: デーモンの名前 (文字列値) を指定します。
 - **type**: 必須。文字列値。
 - **perm**: 必須。文字列値。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/rgw/user/UID/capability

パラメーター

- **UID** を文字列としてユーザー ID に置き換えます。

例

```
POST /api/rgw/user/UID/capability HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "daemon_name": "STRING",
  "perm": "STRING",
  "type": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/rgw/user/UID/key

パラメーター

- **UID** を文字列としてユーザー ID に置き換えます。
- クエリー:
 - **daemon_name**: デーモンの名前 (文字列値) を指定します。
 - **key_type**: 文字列値。
 - **subuser**: 文字列値。
 - **access_key**: 文字列値。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/rgw/user/UID/key

パラメーター

- **UID** を文字列としてユーザー ID に置き換えます。

例

```
POST /api/rgw/user/UID/key HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "access_key": "STRING",
  "daemon_name": "STRING",
  "generate_key": "true",
  "key_type": "s3",
  "secret_key": "STRING",
  "subuser": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

GET /api/rgw/user/UID/quota

パラメーター

- **UID** を文字列としてユーザー ID に置き換えます。

例

```
GET /api/rgw/user/UID/quota HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

PUT /api/rgw/user/UID/quota

パラメーター

- **UID** を文字列としてユーザー ID に置き換えます。

例

```
PUT /api/rgw/user/UID/quota HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "daemon_name": "STRING",
  "enabled": "STRING",
  "max_objects": "STRING",
  "max_size_kb": 1,
  "quota_type": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/rgw/user/UID**/subuser**

パラメーター

- **UID** を文字列としてユーザー ID に置き換えます。

例

```
POST /api/rgw/user/UID/subuser HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "access": "STRING",
  "access_key": "STRING",
  "daemon_name": "STRING",
  "generate_secret": "true",
  "key_type": "s3",
  "secret_key": "STRING",
  "subuser": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/rgw/user/UID/subuser/SUBUSER

パラメーター

- **UID** を文字列としてユーザー ID に置き換えます。
- **SUBUSER** を文字列としてサブユーザー名に置き換えます。
- クエリー:
 - **purge_keys**: 鍵を消去しないように **false** に設定します。これは S3 サブユーザーでのみ機能します。
 - **daemon_name**: デーモンの名前 (文字列値) を指定します。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.18. ロールを操作する REST API

radosgw-admin role コマンドの他に、REST API を使用してロールを操作できます。

REST 管理 API を呼び出すには、admin キャップを持つユーザーを作成します。

例

```
[root@host01 ~]# radosgw-admin --uid TESTER --display-name "TestUser" --access_key TESTER --secret test123 user create
[root@host01 ~]# radosgw-admin caps add --uid="TESTER" --caps="roles=*
```

- ロールを作成します。

構文

```
POST "<hostname>?
Action=CreateRole&RoleName=ROLE_NAME&Path=PATH_TO_FILE&AssumeRolePolicyDocument=TRUST_RELATIONSHIP_POLICY_DOCUMENT"
```

例

```
POST "<hostname>?
Action=CreateRole&RoleName=S3Access&Path=/application_abc/component_xyz/&AssumeRolePolicyDocument={"Version":"2022-06-17","Statement":[{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam:::user/TESTER"]},"Action":["sts:AssumeRole"]}]"
```

応答の例

```
<role>
  <id>8f41f4e0-7094-4dc0-ac20-074a881ccbc5</id>
  <name>S3Access</name>
  <path>/application_abc/component_xyz</path>
  <arn>arn:aws:iam:::role/application_abc/component_xyz/S3Access</arn>
  <create_date>2022-06-23T07:43:42.811Z</create_date>
  <max_session_duration>3600</max_session_duration>
  <assume_role_policy_document>{"Version":"2022-06-17","Statement":[{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam:::user/TESTER"]},"Action":["sts:AssumeRole"]}]}</assume_role_policy_document>
</role>
```

- ロールを取得します。

構文

```
POST "<hostname>?Action=GetRole&RoleName=ROLE_NAME"
```

例

```
POST "<hostname>?Action=GetRole&RoleName=S3Access"
```

応答の例

```
<role>
  <id>8f41f4e0-7094-4dc0-ac20-074a881ccbc5</id>
  <name>S3Access</name>
  <path>/application_abc/component_xyz</path>
  <arn>arn:aws:iam:::role/application_abc/component_xyz/S3Access</arn>
```

```
<create_date>2022-06-23T07:43:42.811Z</create_date>
<max_session_duration>3600</max_session_duration>
<assume_role_policy_document>{"Version":"2022-06-17","Statement":
[{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam:::user/TESTER"]},"Action":
["sts:AssumeRole"]}]</assume_role_policy_document>
</role>
```

- ロールを一覧表示します。

構文

```
POST "<hostname>?
Action=GetRole&RoleName=ROLE_NAME&PathPrefix=PATH_PREFIX"
```

要求の例

```
POST "<hostname>?Action=ListRoles&RoleName=S3Access&PathPrefix=/application"
```

応答の例

```
<role>
<id>8f41f4e0-7094-4dc0-ac20-074a881ccbc5</id>
<name>S3Access</name>
<path>/application_abc/component_xyz</path>
<arn>arn:aws:iam:::role/application_abc/component_xyz/S3Access</arn>
<create_date>2022-06-23T07:43:42.811Z</create_date>
<max_session_duration>3600</max_session_duration>
<assume_role_policy_document>{"Version":"2022-06-17","Statement":
[{"Effect":"Allow","Principal":{"AWS":["arn:aws:iam:::user/TESTER"]},"Action":
["sts:AssumeRole"]}]</assume_role_policy_document>
</role>
```

- assume ロールポリシードキュメントを更新します。

構文

```
POST "<hostname>?
Action=UpdateAssumeRolePolicy&RoleName=ROLE_NAME&PolicyDocument=TRUST_RE
LATIONSHIP_POLICY_DOCUMENT"
```

例

```
POST "<hostname>?
Action=UpdateAssumeRolePolicy&RoleName=S3Access&PolicyDocument=
{"Version":"2022-06-17","Statement":[{"Effect":"Allow","Principal":{"AWS":
["arn:aws:iam:::user/TESTER2"]},"Action":["sts:AssumeRole"]}]"
```

- ロールに割り当てられたポリシーを更新します。

構文

```
POST "<hostname>?
Action=PutRolePolicy&RoleName=ROLE_NAME&PolicyName=POLICY_NAME&PolicyDocu
ment=TRUST_RELATIONSHIP_POLICY_DOCUMENT"
```

例

```
POST "<hostname>?
Action=PutRolePolicy&RoleName=S3Access&PolicyName=Policy1&PolicyDocument=
{"Version":"2022-06-17","Statement":[{"Effect":"Allow","Action":
["s3:CreateBucket"],"Resource":"arn:aws:s3:::example_bucket"}]}"
```

- ロールに割り当てられているパーミッションポリシー名を一覧表示します。

構文

```
POST "<hostname>?Action=ListRolePolicies&RoleName=ROLE_NAME"
```

例

```
POST "<hostname>?Action=ListRolePolicies&RoleName=S3Access"

<PolicyNames>
  <member>Policy1</member>
</PolicyNames>
```

- ロールに割り当てられたパーミッションポリシーを取得します。

構文

```
POST "<hostname>?
Action=GetRolePolicy&RoleName=ROLE_NAME&PolicyName=POLICY_NAME"
```

例

```
POST "<hostname>?Action=GetRolePolicy&RoleName=S3Access&PolicyName=Policy1"

<GetRolePolicyResult>
  <PolicyName>Policy1</PolicyName>
  <RoleName>S3Access</RoleName>
  <Permission_policy>{"Version":"2022-06-17","Statement":[{"Effect":"Allow","Action":
["s3:CreateBucket"],"Resource":"arn:aws:s3:::example_bucket"}]}</Permission_policy>
</GetRolePolicyResult>
```

- ロールに割り当てられたポリシーを削除します。

構文

```
POST "<hostname>?
Action=DeleteRolePolicy&RoleName=ROLE_NAME&PolicyName=POLICY_NAME"
```

例

POST "<hostname>?Action=DeleteRolePolicy&RoleName=S3Access&PolicyName=Policy1"

- ロールを削除します。



注記

ロールは、パーミッションポリシーが割り当てられていない場合にのみ削除できます。

構文

POST "<hostname>?Action=DeleteRole&RoleName=**ROLE_NAME**"

例

POST "<hostname>?Action=DeleteRole&RoleName=S3Access"

関連情報

- 詳細は、Red Hat Ceph Storage Object Gateway ガイドの [ロールの管理](#) セクションを参照してください。

A.19. NFS GANESHA

Ceph RESTful API の **nfs-ganesha** エンドポイントを使用して Ceph NFS ゲートウェイを管理するメソッド参照。

GET /api/nfs-ganesha/daemon

詳細

NFS Ganesha デーモンに関する情報を表示します。

例

GET /api/nfs-ganesha/daemon HTTP/1.1
Host: example.com

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/nfs-ganesha/export

詳細

すべての NFS Ganesha エクスポートを表示します。

例

```
GET /api/nfs-ganesha/export HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/nfs-ganesha/export

詳細

NFS Ganesha エクスポートを新たに作成します。

例

```
POST /api/nfs-ganesha/export HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "access_type": "STRING",
  "clients": [
    {
      "access_type": "STRING",
      "addresses": [
        "STRING"
      ],
      "squash": "STRING"
    }
  ],
  "cluster_id": "STRING",
  "daemons": [
    "STRING"
  ],
  "fsal": {
    "filesystem": "STRING",
    "name": "STRING",
    "rgw_user_id": "STRING",
    "sec_label_xattr": "STRING",
    "user_id": "STRING"
  },
  "path": "STRING",
```

```

    "protocols": [
      1
    ],
    "pseudo": "STRING",
    "reload_daemons": true,
    "security_label": "STRING",
    "squash": "STRING",
    "tag": "STRING",
    "transports": [
      "STRING"
    ]
  }

```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/nfs-ganesha/export/CLUSTER_ID/EXPORT_ID

詳細

NFS Ganesha エクスポートを削除します。

パラメーター

- **CLUSTER_ID** はストレージクラスター識別子の文字列に置き換えます。
- **EXPORT_ID** は、エクスポート ID を整数に置き換えます。
- クエリー:
 - **reload_daemons**: NFS Ganesha デーモン設定の再読み込みをトリガーするブール値。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。

- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/nfs-ganesha/export/CLUSTER_ID/EXPORT_ID

詳細

NFS Ganesha エクスポート情報を表示します。

パラメーター

- **CLUSTER_ID** はストレージクラスター識別子の文字列に置き換えます。
- **EXPORT_ID** は、エクスポート ID を整数に置き換えます。

例

```
GET /api/nfs-ganesha/export/CLUSTER_ID/EXPORT_ID HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

PUT /api/nfs-ganesha/export/CLUSTER_ID/EXPORT_ID

詳細

NFS Ganesha エクスポート情報を更新します。

パラメーター

- **CLUSTER_ID** はストレージクラスター識別子の文字列に置き換えます。
- **EXPORT_ID** は、エクスポート ID を整数に置き換えます。

例

```
PUT /api/nfs-ganesha/export/CLUSTER_ID/EXPORT_ID HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "access_type": "STRING",
  "clients": [
    {
      "access_type": "STRING",
```

```

        "addresses": [
            "STRING"
        ],
        "squash": "STRING"
    }
],
"daemons": [
    "STRING"
],
"fsal": {
    "filesystem": "STRING",
    "name": "STRING",
    "rgw_user_id": "STRING",
    "sec_label_xattr": "STRING",
    "user_id": "STRING"
},
"path": "STRING",
"protocols": [
    1
],
"pseudo": "STRING",
"reload_daemons": true,
"security_label": "STRING",
"squash": "STRING",
>tag": "STRING",
"transports": [
    "STRING"
]
}

```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/nfs-ganesha/status

詳細

NFS Ganesha 管理機能のステータス情報を表示します。

例

```

GET /api/nfs-ganesha/status HTTP/1.1
Host: example.com

```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。
- 詳細は、Red Hat Ceph Storage Object Gateway ガイドの [namespace から NFS-Ganesha へのエクスポート](#) セクションを参照してください。

A.20. CEPH ORCHESTRATOR

Ceph RESTful API の **orchestrator** エンドポイントを使用して Ceph Orchestrator ステータスを表示するメソッド参照。

GET /api/orchestrator/status

詳細

Ceph Orchestrator ステータスを表示します。

例

```
GET /api/orchestrator/status HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.21. POOLS

Ceph RESTful API の **pool** エンドポイントを使用してストレージプールを管理するメソッド参照。

GET /api/pool

詳細

プールの一覧を表示します。

パラメーター

- クエリー:
 - **attrs**: プール属性の文字列値。
 - **stats**: プール統計のブール値。

例

```
GET /api/pool HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/pool

例

```
POST /api/pool HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "application_metadata": "STRING",
  "configuration": "STRING",
  "erasure_code_profile": "STRING",
  "flags": "STRING",
  "pg_num": 1,
  "pool": "STRING",
```

```
    "pool_type": "STRING",  
    "rule_name": "STRING"  
  }
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

DELETE /api/pool/POOL_NAME

パラメーター

- **POOL_NAME** をプールの名前に置き換えます。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

GET /api/pool/POOL_NAME

パラメーター

- **POOL_NAME** をプールの名前に置き換えます。
- クエリー:
 - **attrs**: プール属性の文字列値。
 - **stats**: プール統計のブール値。

例


```
GET /api/pool/POOL_NAME HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

PUT /api/pool/**POOL_NAME**

パラメーター

- **POOL_NAME** をプールの名前に置き換えます。

例

```
PUT /api/pool/POOL_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "application_metadata": "STRING",
  "configuration": "STRING",
  "flags": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/pool/**POOL_NAME**/configuration

パラメーター

- **POOL_NAME** をプールの名前に置き換えます。

例

```
GET /api/pool/POOL_NAME/configuration HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.22. PROMETHEUS

Ceph RESTful API の **prometheus** エンドポイントを使用して Prometheus を管理するためのメソッド参照。

GET /api/prometheus

例

```
GET /api/prometheus/rules HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/prometheus/rules

例

```
GET /api/prometheus/rules HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/prometheus/silence

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/prometheus/silence/S_ID

パラメーター

- **S_ID** を文字列値に置き換えます。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。

- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/prometheus/silences

例

```
GET /api/prometheus/silences HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/prometheus/notifications

例

```
GET /api/prometheus/notifications HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.23. RADOS ブロックデバイス

Ceph RESTful API の **block** エンドポイントを使用して RADOS ブロックデバイス (RBD) を管理するためのメソッド参照。この参照には、利用可能なすべての RBD 機能エンドポイントが含まれます。以下に例を示します。

- [RBD 名前空間](#)
- [RBD スナップショット](#)
- [RBD ゴミ箱](#)
- [RBD ミラーニング](#)
 - [RBD ミラーリングの概要](#)
 - [RBD ミラーリングプールブートストラップ](#)
 - [RBD ミラーリングプールモード](#)
 - [RBD ミラーリングプールピア](#)

RBD イメージ

GET /api/block/image

詳細

RBD イメージを表示します。

パラメーター

- クエリー:
 - **pool_name**: プール名 (文字列)

例

```
GET /api/block/image HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/block/image

例

```
POST /api/block/image HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "configuration": "STRING",
  "data_pool": "STRING",
  "features": "STRING",
  "name": "STRING",
  "namespace": "STRING",
  "obj_size": 1,
  "pool_name": "STRING",
  "size": 1,
  "stripe_count": 1,
  "stripe_unit": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/block/image/clone_format_version

詳細

RBD クローン形式のバージョンを返します。

例

```
GET /api/block/image/clone_format_version HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。

- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/block/image/default_features

例

```
GET /api/block/image/default_features HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/block/image/default_features

例

```
GET /api/block/image/default_features HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/block/image/IMAGE_SPEC

パラメーター

- **IMAGE_SPEC** を文字列の値としてイメージ名に置き換えます。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。

- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

GET /api/block/image/IMAGE_SPEC

パラメーター

- **IMAGE_SPEC** を文字列の値としてイメージ名に置き換えます。

例

```
GET /api/block/image/IMAGE_SPEC HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

PUT /api/block/image/IMAGE_SPEC

パラメーター

- **IMAGE_SPEC** を文字列の値としてイメージ名に置き換えます。

例

```
PUT /api/block/image/IMAGE_SPEC HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "configuration": "STRING",
  "features": "STRING",
  "name": "STRING",
  "size": 1
}
```


ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/block/image/IMAGE_SPEC/copy

パラメーター

- **IMAGE_SPEC** を文字列の値としてイメージ名に置き換えます。

例

```
POST /api/block/image/IMAGE_SPEC/copy HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "configuration": "STRING",
  "data_pool": "STRING",
  "dest_image_name": "STRING",
  "dest_namespace": "STRING",
  "dest_pool_name": "STRING",
  "features": "STRING",
  "obj_size": 1,
  "snapshot_name": "STRING",
  "stripe_count": 1,
  "stripe_unit": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/block/image/IMAGE_SPEC/flatten

パラメーター

- **IMAGE_SPEC** を文字列の値としてイメージ名に置き換えます。

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/block/image/IMAGE_SPEC/move_trash

詳細

イメージをゴミ箱に移動します。クローンによってアクティブに使用されているイメージは、ゴミ箱に移動して、後で削除することができます。

パラメーター

- **IMAGE_SPEC** を文字列の値としてイメージ名に置き換えます。

例

```
POST /api/block/image/IMAGE_SPEC/move_trash HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "delay": 1
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

RBD ミラーニング

GET /api/block/mirroring/site_name

詳細

RBD ミラーリングサイト名を表示します。

例

```
GET /api/block/mirroring/site_name HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

PUT /api/block/mirroring/site_name

例

```
PUT /api/block/mirroring/site_name HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "site_name": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

RBD ミラーリングプールブートストラップ

POST /api/block/mirroring/pool/**POOL_NAME**/bootstrap/peer

パラメーター

- **POOL_NAME** を文字列としてプールの名前に置き換えます。

例

```
POST /api/block/mirroring/pool/POOL_NAME/bootstrap/peer HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "direction": "STRING",
  "token": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/block/mirroring/pool/**POOL_NAME**/bootstrap/token

パラメーター

- **POOL_NAME** を文字列としてプールの名前に置き換えます。

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

RBD ミラーリングプールモード

GET /api/block/mirroring/pool/POOL_NAME

詳細

RBD ミラーリングの概要を表示します。

パラメーター

- **POOL_NAME** を文字列としてプールの名前に置き換えます。

例

```
GET /api/block/mirroring/pool/POOL_NAME HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

PUT /api/block/mirroring/pool/POOL_NAME

パラメーター

- **POOL_NAME** を文字列としてプールの名前に置き換えます。

例

```
PUT /api/block/mirroring/pool/POOL_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "mirror_mode": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。

- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

RBD ミラーリングプールピア

GET /api/block/mirroring/pool/**POOL_NAME**/peer

パラメーター

- **POOL_NAME** を文字列としてプールの名前に置き換えます。

例

```
GET /api/block/mirroring/pool/POOL_NAME/peer HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/block/mirroring/pool/**POOL_NAME**/peer

パラメーター

- **POOL_NAME** を文字列としてプールの名前に置き換えます。

例

```
POST /api/block/mirroring/pool/POOL_NAME/peer HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "client_id": "STRING",
  "cluster_name": "STRING",
  "key": "STRING",
  "mon_host": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/block/mirroring/pool/**POOL_NAME**/peer/**PEER_UUID**

パラメーター

- **POOL_NAME** を文字列としてプールの名前に置き換えます。
- **PEER_UUID** は、文字列としてピアの UUID に置き換えます。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/block/mirroring/pool/**POOL_NAME**/peer/**PEER_UUID**

パラメーター

- **POOL_NAME** を文字列としてプールの名前に置き換えます。
- **PEER_UUID** は、文字列としてピアの UUID に置き換えます。

例

```
GET /api/block/mirroring/pool/POOL_NAME/peer/PEER_UUID HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.

- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

PUT /api/block/mirroring/pool/**POOL_NAME**/peer/**PEER_UUID**

パラメーター

- **POOL_NAME** を文字列としてプールの名前に置き換えます。
- **PEER_UUID** は、文字列としてピアの UUID に置き換えます。

例

```
PUT /api/block/mirroring/pool/POOL_NAME/peer/PEER_UUID HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "client_id": "STRING",
  "cluster_name": "STRING",
  "key": "STRING",
  "mon_host": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

RBD ミラーリングの概要

GET /api/block/mirroring/summary

詳細

RBD ミラーリングの概要を表示します。

例

GET /api/block/mirroring/summary HTTP/1.1
Host: example.com

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

RBD 名前空間

GET /api/block/pool/**POOL_NAME**/namespace

パラメーター

- **POOL_NAME** を文字列としてプールの名前に置き換えます。

例

GET /api/block/pool/**POOL_NAME**/namespace HTTP/1.1
Host: example.com

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/block/pool/**POOL_NAME**/namespace

パラメーター

- **POOL_NAME** を文字列としてプールの名前に置き換えます。

例

POST /api/block/pool/**POOL_NAME**/namespace HTTP/1.1

```
Host: example.com
Content-Type: application/json

{
  "namespace": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

DELETE /api/block/pool/POOL_NAME/namespace/NAMESPACE

パラメーター

- **POOL_NAME** を文字列としてプールの名前に置き換えます。
- **NAMESPACE** を文字列として namespace に置き換えます。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

RBD スナップショット

POST /api/block/image/IMAGE_SPEC/snap

パラメーター

- **IMAGE_SPEC** を文字列の値としてイメージ名に置き換えます。

例

```
POST /api/block/image/IMAGE_SPEC/snap HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "snapshot_name": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME

パラメーター

- **IMAGE_SPEC** を文字列の値としてイメージ名に置き換えます。
- **SNAPSHOT_NAME** は、スナップショットの名前を文字列値として置き換えます。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

PUT /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME

パラメーター

- **IMAGE_SPEC** を文字列の値としてイメージ名に置き換えます。

- **SNAPSHOT_NAME** は、スナップショットの名前を文字列値として置き換えます。

例

```
PUT /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "is_protected": true,
  "new_snap_name": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/block/image/**IMAGE_SPEC**/snap/**SNAPSHOT_NAME**/clone

詳細

スナップショットをイメージにクローンします。

パラメーター

- **IMAGE_SPEC** を文字列の値としてイメージ名に置き換えます。
- **SNAPSHOT_NAME** は、スナップショットの名前を文字列値として置き換えます。

例

```
POST /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME/clone HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "child_image_name": "STRING",
  "child_namespace": "STRING",
  "child_pool_name": "STRING",
  "configuration": "STRING",
  "data_pool": "STRING",
  "features": "STRING",
  "obj_size": 1,
```

```

    "stripe_count": 1,
    "stripe_unit": "STRING"
  }

```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/block/image/IMAGE_SPEC/snap/SNAPSHOT_NAME/rollback

パラメーター

- **IMAGE_SPEC** を文字列の値としてイメージ名に置き換えます。
- **SNAPSHOT_NAME** は、スナップショットの名前を文字列値として置き換えます。

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

RBD ゴミ箱

GET /api/block/image/trash

詳細

すべての RBD ゴミ箱エントリー、またはプール名ごとの RBD ゴミ箱の詳細を表示します。

パラメーター

- クエリー:
 - **pool_name**: プールの名前 (文字列値)

例

```
GET /api/block/image/trash HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/block/image/trash/purge

詳細

ゴミ箱から期限切れのイメージをすべて削除します。

パラメーター

- クエリー:
 - **pool_name**: プールの名前 (文字列値)

例

```
POST /api/block/image/trash/purge HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "pool_name": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/block/image/trash/IMAGE_ID_SPEC**詳細**

ゴミ箱からイメージを削除します。イメージの延期時間が経過していない場合は、**force** を使用しない限り削除できません。クローン別にアクティブに使用中のイメージや、スナップショットが含まれているため、削除できません。

パラメーター

- **IMAGE_ID_SPEC** は、イメージ名を文字列値として置き換えます。
- クエリー:
 - **force**: ごみ箱からイメージの削除を強制するブール値。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

POST /api/block/image/trash/IMAGE_ID_SPEC/restore**詳細**

ゴミ箱からイメージを復元します。

パラメーター

- **IMAGE_ID_SPEC** は、イメージ名を文字列値として置き換えます。

例

```
POST /api/block/image/trash/IMAGE_ID_SPEC/restore HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "new_image_name": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。

- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.24. パフォーマンスカウンター

Ceph RESTful API の **perf_counters** エンドポイントを使用するメソッド参照。さまざまな Ceph パフォーマンスカウンターを表示します。この参照には、利用可能なすべてのパフォーマンスカウンターが含まれます。以下に例を示します。

- [Ceph Metadata Server \(MDS\)](#)
- [Ceph Manager](#)
- [Ceph Monitor](#)
- [Ceph OSD](#)
- [Ceph Object Gateway](#)
- [Ceph RADOS ブロックデバイス \(RBD\) ミラーリング](#)
- [TCMU ランナー](#)

GET /api/perf_counters

詳細

パフォーマンスカウンターを表示します。

例

```
GET /api/perf_counters HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。

- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

Ceph Metadata Server

GET /api/perf_counters/mds/SERVICE_ID

パラメーター

- **SERVICE_ID** を文字列として必要なサービス識別子に置き換えます。

例

```
GET /api/perf_counters/mds/SERVICE_ID HTTP/1.1  
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

Ceph Manager

GET /api/perf_counters/mgr/SERVICE_ID

パラメーター

- **SERVICE_ID** を文字列として必要なサービス識別子に置き換えます。

例

```
GET /api/perf_counters/mgr/SERVICE_ID HTTP/1.1  
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。

- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

Ceph Monitor

GET /api/perf_counters/mon/SERVICE_ID

パラメーター

- **SERVICE_ID** を文字列として必要なサービス識別子に置き換えます。

例

```
GET /api/perf_counters/mon/SERVICE_ID HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

Ceph OSD

GET /api/perf_counters/osd/SERVICE_ID

パラメーター

- **SERVICE_ID** を文字列として必要なサービス識別子に置き換えます。

例

```
GET /api/perf_counters/osd/SERVICE_ID HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。

- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

Ceph RADOS ブロックデバイス (RBD) ミラーリング

GET /api/perf_counters/rbd-mirror/SERVICE_ID

パラメーター

- **SERVICE_ID** を文字列として必要なサービス識別子に置き換えます。

例

```
GET /api/perf_counters/rbd-mirror/SERVICE_ID HTTP/1.1  
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

Ceph Object Gateway

GET /api/perf_counters/rgw/SERVICE_ID

パラメーター

- **SERVICE_ID** を文字列として必要なサービス識別子に置き換えます。

例

```
GET /api/perf_counters/rgw/SERVICE_ID HTTP/1.1  
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。

- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

TCMU ランナー

GET /api/perf_counters/tcmu-runner/SERVICE_ID

パラメーター

- **SERVICE_ID** を文字列として必要なサービス識別子に置き換えます。

例

```
GET /api/perf_counters/tcmu-runner/SERVICE_ID HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.25. ロール

Ceph RESTful API の **role** エンドポイントを使用して Ceph のさまざまなユーザーロールを管理するためのメソッド参照。

GET /api/role

詳細

ロールの一覧を表示します。

例

```
GET /api/role HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/role

例

```
POST /api/role HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "description": "STRING",
  "name": "STRING",
  "scopes_permissions": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/role/NAME

パラメーター

- **NAME** を文字列としてロール名に置き換えます。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.

- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

GET /api/role/NAME

パラメーター

- **NAME** を文字列としてロール名に置き換えます。

例

```
GET /api/role/NAME HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

PUT /api/role/NAME

パラメーター

- **NAME** を文字列としてロール名に置き換えます。

例

```
PUT /api/role/NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "description": "STRING",
  "scopes_permissions": "STRING"
}
```

ステータスコード

- 200 OK – Okay.

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/role/NAME/clone

パラメーター

- **NAME** を文字列としてロール名に置き換えます。

例

```
POST /api/role/NAME/clone HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "new_name": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.26. サービス

Ceph RESTful API の **service** エンドポイントを使用してさまざまな Ceph サービスを管理するためのメソッド参照。

GET /api/service

パラメーター

- クエリー:
 - service_name**: 文字列としてサービス名。

例

```
GET /api/service HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/service

パラメーター

- service_spec**: サービスの JSON ファイルとしての指定。
- service_name**: サービスの名前

例

```
POST /api/service HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "service_name": "STRING",
  "service_spec": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。

- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/service/known_types

詳細

既知のサービスタイプの一覧を表示します。

例

```
GET /api/service/known_types HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/service/SERVICE_NAME

パラメーター

- **SERVICE_NAME** は、サービス名を文字列として置き換えます。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/service/SERVICE_NAME

パラメーター

- **SERVICE_NAME** は、サービス名を文字列として置き換えます。

例

```
GET /api/service/SERVICE_NAME HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

GET /api/service/**SERVICE_NAME**/daemons

パラメーター

- **SERVICE_NAME** は、サービス名を文字列として置き換えます。

例

```
GET /api/service/SERVICE_NAME/daemons HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.27. 設定

Ceph RESTful API の **settings** エンドポイントを使用してさまざまな Ceph 設定を管理するメソッド参照。

GET /api/settings

詳細

利用可能なオプションの一覧を表示します。

パラメーター

- クエリー:
 - names:** オプション名のコンマ区切りリスト

例

```
GET /api/settings HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

PUT /api/settings

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

DELETE /api/settings/NAME

パラメーター

- NAME** を文字列としてオプション名に置き換えます。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/settings/NAME

詳細

指定オプションを表示します。

パラメーター

- **NAME** を文字列としてオプション名に置き換えます。

例

```
GET /api/settings/NAME HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

PUT /api/settings/NAME

パラメーター

- **NAME** を文字列としてオプション名に置き換えます。

例

```
PUT /api/settings/NAME HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "value": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.28. CEPH タスク

Ceph RESTful API の **task** エンドポイントを使用して Ceph タスクを表示するためのメソッド参照。

GET /api/task

詳細

Ceph タスクを表示します。

パラメーター

- クエリー:
 - 名前: タスクの名前。

例

```
GET /api/task HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。

- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.29. テレメトリー

Ceph RESTful API の **telemetry** エンドポイントを使用して telemetry Ceph Manager モジュールのデータを管理するためのメソッド参照。

PUT /api/telemetry

詳細

telemetry モジュールで収集したデータの送信を有効または無効にします。

パラメーター

- **enable**: ブール値です。
- **license_name**: 文字列の値 (例: **sharing-1-0**) です。ユーザーが認識し、テレメトリーデータを共有するライセンスを受け入れるようにしてください。

例

```
PUT /api/telemetry HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "enable": true,
  "license_name": "STRING"
}
```

ステータスコード

- 200 OK – Okay.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/telemetry/report

詳細

Ceph およびデバイスのレポートデータを表示します。

例

```
GET /api/telemetry/report HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

A.30. CEPH ユーザー

Ceph RESTful API の **user** エンドポイントを使用して Ceph ユーザーの詳細を表示し、Ceph ユーザーパスワードを管理する方法の参照です。

GET /api/user

詳細

ユーザーの一覧を表示します。

例

```
GET /api/user HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。

- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/user

例

```
POST /api/user HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "email": "STRING",
  "enabled": true,
  "name": "STRING",
  "password": "STRING",
  "pwdExpirationDate": "STRING",
  "pwdUpdateRequired": true,
  "roles": "STRING",
  "username": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

DELETE /api/user/USER_NAME

パラメーター

- **USER_NAME** は、文字列としてユーザーの名前に置き換えます。

ステータスコード

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 204 No Content – Resource deleted.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。

- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

GET /api/user/USER_NAME

パラメーター

- **USER_NAME** は、文字列としてユーザーの名前に置き換えます。

例

```
GET /api/user/USER_NAME HTTP/1.1
Host: example.com
```

ステータスコード

- 200 OK – Okay.
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

PUT /api/user/USER_NAME

パラメーター

- **USER_NAME** は、文字列としてユーザーの名前に置き換えます。

例

```
PUT /api/user/USER_NAME HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "email": "STRING",
  "enabled": "STRING",
  "name": "STRING",
  "password": "STRING",
  "pwdExpirationDate": "STRING",
  "pwdUpdateRequired": true,
  "roles": "STRING"
}
```

ステータスコード

- 200 OK – Okay.

- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/user/**USER_NAME**/change_password

パラメーター

- **USER_NAME** は、文字列としてユーザーの名前に置き換えます。

例

```
POST /api/user/USER_NAME/change_password HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "new_password": "STRING",
  "old_password": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディを確認してください。

POST /api/user/validate_password

詳細

パスワードを確認して、パスワードポリシーが満たされているかどうかを確認します。

パラメーター

- **Password**: 検証するパスワード。
- **ユーザー名**: 任意。ユーザーの名前。

- **old_password**: 任意。古いパスワード。

例

```
POST /api/user/validate_password HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "old_password": "STRING",
  "password": "STRING",
  "username": "STRING"
}
```

ステータスコード

- 201 Created – Resource created.
- 202 Accepted – Operation is still executing.タスクキューを確認してください。
- 400 Bad Request – Operation exception.詳細は、レスポンスボディーを確認してください。
- 401 Unauthorized – Unauthenticated access.最初にログインしてください。
- 403 Forbidden – Unauthorized access.パーミッションを確認してください。
- 500 Internal Server Error – Unexpected error.スタックトレースのレスポンスボディーを確認してください。

関連情報

- 詳細は、Red Hat Ceph Storage 開発者ガイドの [Ceph RESTful API](#) の章を参照してください。

付録B S3 の一般的なリクエストヘッダー

以下の表には、有効な一般的なリクエストヘッダーとその説明をまとめています。

表B.1 リクエストヘッダー

リクエストヘッダー	詳細
CONTENT_LENGTH	リクエストボディの長さ。
DATE	要求の日時と日付 (UTC 単位)。
HOST	ホストサーバーの名前。
AUTHORIZATION	承認トークン。

付録C S3 の一般的なレスポンスステータスコード

以下の表は、有効な一般的な HTTP レスポンスステータスと対応するコードを示しています。

表C.1 レスポンスのステータス

HTTP ステータス	レスポンスコード
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 ステータス	レスポンスコード
404	NoSuchKey
404	NoSuchBucket
404	NoSuchUpload
405	MethodNotAllowed
408	RequestTimeout
409	BucketAlreadyExists
409	BucketNotEmpty
411	MissingContentLength
412	PreconditionFailed
416	InvalidRange
422	UnprocessableEntity
500	InternalServerError

付録D S3 サポートされないヘッダーフィールド

表D.1 サポートされないヘッダーフィールド

名前	型
x-amz-security-token	リクエスト
Server	レスポンス
x-amz-delete-marker	レスポンス
x-amz-id-2	レスポンス
x-amz-request-id	レスポンス
x-amz-version-id	レスポンス

付録E SWIFT リクエストヘッダー

表E.1 リクエストヘッダー

名前	説明	型	必須
X-Auth-User	認証するキーの Ceph Object Gateway のユーザー名。	String	はい
X-Auth-Key	Ceph Object Gateway のユーザー名に関連付けられたキー。	String	はい

付録F SWIFT レスponseヘッダー

サーバーからのレスponseには、**X-Auth-Token** の値が含まれている必要があります。レスponseには、API のドキュメント全体で他のリクエストに指定される **API_VERSION/ACCOUNT** 接頭辞を提供する **X-Storage-Url** も含まれる可能性があります。

表F.1 レスponseヘッダー

名前	説明	型
X-Storage-Token	要求に指定された X-Auth-User の承認トークン。	String
X-Storage-Url	ユーザーの URL および API_VERSION/ACCOUNT パス。	String

付録G SECURE TOKEN SERVICE API の使用例

これらの例は、Python の **boto3** モジュールを使用して、Ceph Object Gateway の Secure Token Service (STS) の実装と対話しています。これらの例では、**TESTER2** は **TESTER1** によって作成されたロールを想定しています。これは、ロールに割り当てられたパーミッションポリシーに基づいて **TESTER1** が所有する S3 リソースにアクセスするためです。

AssumeRole のサンプルはロールを作成し、ポリシーをロールに割り当てます。次に、一時認証情報を取得し、それらの一時認証情報を使用して S3 リソースにアクセスするロールを想定します。

AssumeRoleWithWebIdentity の例は、OpenID Connect ID プロバイダーである Keycloak を使用して外部アプリケーションを使用してユーザーを認証し、ロールのアクセス許可ポリシーに従って一時的な認証情報を取得して S3 リソースにアクセスするロールを引き受けます。

AssumeRole の例

```
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 の例

```
import boto3

iam_client = boto3.client('iam',
aws_access_key_id=ACCESS_KEY_OF_TESTER1,
aws_secret_access_key=SECRET_KEY_OF_TESTER1,
endpoint_url=<IAM URL>,
region_name="
)

oidc_response = iam_client.create_open_id_connect_provider(
    Url=<URL of the OpenID Connect Provider>,
    ClientIDList=[
        <Client id registered with the IDP>
    ],
    ThumbprintList=[
        <IDP THUMBPRINT>
    ]
)

policy_document = "{\"Version\":\"2012-10-17\",\"Statement\":\":[{\"Effect\":\"Allow\",\"Principal\":\
{\"Federated\":\":[\"arn:aws:iam::oidc-provider/localhost:8080/auth/realms/demo/\"],\"Action\":\
[\"sts:AssumeRoleWithWebIdentity\"],\"Condition\":\{\"StringEquals\":\
{\"localhost:8080/auth/realms/demo:app_id\": \"customer-portal/\"}}}]}"
role_response = iam_client.create_role(
    AssumeRolePolicyDocument=policy_document,
    Path='/',
    RoleName='S3Access',
)

role_policy = "{\"Version\":\"2012-10-17\",\"Statement\":\
[\"Effect\": \"Allow\", \"Action\": \"s3:*\", \"Resource\": \"arn:aws:s3:::*\"]}"

response = iam_client.put_role_policy(
    RoleName='S3Access',
    PolicyName='Policy1',
    PolicyDocument=role_policy
)

sts_client = boto3.client('sts',
aws_access_key_id=ACCESS_KEY_OF_TESTER2,
aws_secret_access_key=SECRET_KEY_OF_TESTER2,
endpoint_url=<STS URL>,
region_name=",
```

```
)

response = 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()
```

関連情報

- Python の **boto** モジュールの使用に関する詳細は、**Red Hat Ceph Storage Object Gateway 設定および管理ガイド**の [S3 アクセスのテスト](#) セクションを参照してください。