CONTAINER OPERATIONS

A container is a mechanism for storing data objects. An account may have many containers, but container names must be unique. This API enables a client to create a container, set access controls and metadata, retrieve a container's contents, and delete a container. Since this API makes requests related to information in a particular user's account, all requests in this API must be authenticated unless a container's access control is deliberately made publicly accessible (i.e., allows anonymous requests).

> **Note:** The Amazon S3 API uses the term 'bucket' to describe a data container. When you hear someone refer to a 'bucket' within the Swift API, the term 'bucket' may be construed as the equivalent of the term 'container.'

One facet of object storage is that it does not support hierachical paths or directories. Instead, it supports one level consisting of one or more containers, where each container may have objects. The RADOS Gateway's Swift-compatible API supports the notion of 'psuedo-hierarchical containers,' which is a means of using object naming to emulate a container (or directory) hierachy without actually implementing one in the storage system. You may name objects with pseudo-hiearchical names (e.g., photos/buildings/empire-state.jpg), but container names cannot contain a forward slash (/) character.

CREATE A CONTAINER

To create a new container, make a PUT request with the API version, account, and the name of the new container. The container name must be unique, must not contain a forward-slash (/) character, and should be less than 256 bytes. You may include access control headers and metadata headers in the request. The operation is idempotent; that is, if you make a request to create a container that already exists, it will return with a HTTP 202 return code, but will not create another container.

SYNTAX

```
PUT /{api version}/{account}/{container} HTTP/1.1
Host: {fqdn}
X-Auth-Token: {auth-token}
X-Container-Read: {comma-separated-uids}
X-Container-Write: {comma-separated-uids}
X-Container-Meta-{key}: {value}
```

HEADERS

`X-Container-Read`

Description: The user IDs with read permissions for the container.
Type: Comma-separated string values of user IDs.
Required: No

`X-Container-Write`

Description: The user IDs with write permissions for the container.
Type: Comma-separated string values of user IDs.
Required: No

`X-Container-Meta-{key}`

Description: A user-defined meta data key that takes an arbitrary string value.
Type: String
Required: No

HTTP RESPONSE

If a container with the same name already exists, and the user is the container owner then the operation will succeed. Otherwise the operation will fail.

409

Description: The container already exists under a different user's ownership.
Status Code: `BucketAlreadyExists`

LIST A CONTAINER'S OBJECTS

To list the objects within a container, make a GET request with the with the API version, account, and the name of the container. You can specify query parameters to filter the full list, or leave out the parameters to return a list of the first 10,000 object names stored in the container.

SYNTAX

```
GET /{api version}/{container} HTTP/1.1
     Host: {fqdn}
     X-Auth-Token: {auth-token}
```

PARAMETERS

`format`

Description: Defines the format of the result.
Type: String
Valid Values: `json` | `xml`
Required: No

`prefix`

Description: Limits the result set to objects beginning with the specified prefix.
Type: String
Required: No

`marker`

Description: Returns a list of results greater than the marker value.
Type: String
Required: No

`limit`

Description: Limits the number of results to the specified value.
Type: Integer
Valid Range: 0 - 10,000
Required: No

`delimiter`

Description: The delimiter between the prefix and the rest of the object name.
Type: String
Required: No

`path`

Description: The pseudo-hierarchical path of the objects.
Type: String
Required: No

RESPONSE ENTITIES

`container`

Description: The container.
Type: Container

`object`

Description: An object within the container.
Type: Container

`name`

Description: The name of an object within the container.
Type: String

`hash`

Description: A hash code of the object's contents.
Type: String

`last_modified`

Description: The last time the object's contents were modified.
Type: Date

`content_type`

Description: The type of content within the object.
Type: String

UPDATE A CONTAINER'S ACLS

When a user creates a container, the user has read and write access to the container by default. To allow other users to read a container's contents or write to a container, you must specifically enable the user. You may also specify * in the `X-Container-Read` or `X-Container-Write` settings, which effectively enables all users to either read from or write to the container. Setting * makes the container public. That is it enables anonymous users to either read from or write to the container.

SYNTAX

```
POST /{api version}/{account}/{container} HTTP/1.1
Host: {fqdn}
     X-Auth-Token: {auth-token}
     X-Container-Read: *
     X-Container-Write: {uid1}, {uid2}, {uid3}
```

REQUEST HEADERS

`X-Container-Read`

Description: The user IDs with read permissions for the container.
Type: Comma-separated string values of user IDs.
Required: No

`X-Container-Write`

Description: The user IDs with write permissions for the container.
Type: Comma-separated string values of user IDs.
Required: No

ADD/UPDATE CONTAINER METADATA

To add metadata to a container, make a POST request with the API version, account, and container name. You must have write

permissions on the container to add or update metadata.

SYNTAX

```
POST /{api version}/{account}/{container} HTTP/1.1
Host: {fqdn}
     X-Auth-Token: {auth-token}
     X-Container-Meta-Color: red
     X-Container-Meta-Taste: salty
```

REQUEST HEADERS

`X-Container-Meta-{key}`

Description: A user-defined meta data key that takes an arbitrary string value.
Type: String
Required: No

DELETE A CONTAINER

To delete a container, make a `DELETE` request with the API version, account, and the name of the container. The container must be empty. If you'd like to check if the container is empty, execute a `HEAD` request against the container. Once you've successfully removed the container, you'll be able to reuse the container name.

SYNTAX

```
DELETE /{api version}/{account}/{container} HTTP/1.1
Host: {fqdn}
X-Auth-Token: {auth-token}
```

HTTP RESPONSE

`204`

Description: The container was removed.
Status Code: `NoContent`