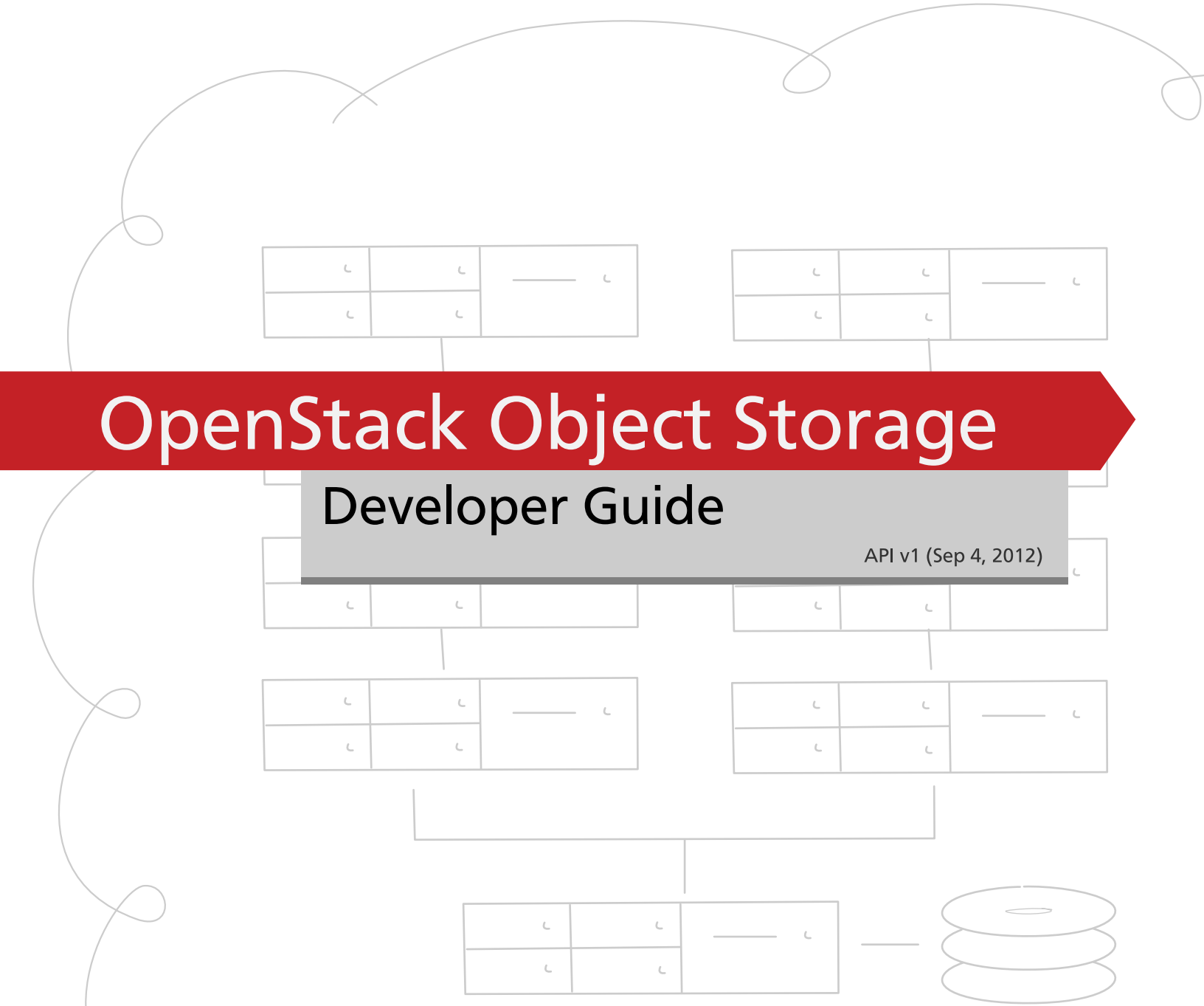

OpenStack Object Storage

Developer Guide

API v1 (Sep 4, 2012)

openstack™

docs.openstack.org

OpenStack Object Storage Developer Guide

API v1 (2012-09-04)

This document is intended for software developers interested in developing applications using the OpenStack™ Object Storage Application Programming Interface (API).

Table of Contents

List of Examples

1. Overview

OpenStack Object Storage is an affordable, redundant, scalable, and dynamic storage service offering. The core storage system is designed to provide a safe, secure, automatically re-sizing and network-accessible way to store data. You can store an unlimited quantity of files and each file can be as large as 5 gigabytes, plus with large object creation, you can upload and store objects of virtually any size.

OpenStack Object Storage allows users to store and retrieve files and content via a simple Web Service inteface (ReST: Representational State Transfer). There are also language-specific APIs that utilize the ReSTful API but make it much easier for developers to integrate into their applications.

For more details on the OpenStack Object Storage service, please refer to http://swift.openstack.org

We welcome feedback, comments, and bug reports at http://bugs.launchpad.net/swift.

1.1. Intended Audience

This guide is intended to assist software developers who want to develop applications using the OpenStack Object Storage API. It fully documents the ReST application programming interface (API) that allows developers to interact with the storage components of the OpenStack Object Storage system. To use the information provided here, you should first have a general understanding of the OpenStack Object Storage service and have access to an installation of OpenStack Object Storage. You should also be familiar with:

- ReSTful web services
- HTTP/1.1

Rackspace also provides Rackspace-supported, language-specific APIs in several popular programming languages. Currently, the supported APIs are C#/.NET, Java, PHP, Python, and Ruby. These APIs utilize the ReST API and are provided to help developers rapidly integrate OpenStack Object Storage support into their applications without needing to write at the ReST interface. Each API includes its own documentation in its native format. For example, the Java API includes JavaDocs and the C#/.NET API includes a CHM file.

1.2. Document Change History

This version of the Developer Guide was forked from the Rackspace Cloud Files Developer Guide.

Revision Date	Summary of Changes
Sep 12, 2012	• Fixed bug 1049362 - Adds back metadata deletion information.
Sep 5, 2012	• Fixed bugs 1009706 - Added Object Versioning and Static Web.
Mar 29, 2012	• Fixed bugs 890435 and 907563 - Add/Update Container Metadata and Expiring Objects. Changed to Maven 1.0.10.
Feb -10, 2011	• Revised to change first to last in the first range example for fetching a portion of an object.
Jan 25, 2011	• Revised for OpenStack Object Storage use by removing CDN references, Rackspace Cloud references, and revised account examples and URLs for generic implementations. It's not a

Revision Date	Summary of Changes
	changed requirement that Container and Object names are required to be UTF-8, but it's pointed out in the documentation.
Jan 12, 2011	• Removed references to ACL (Access Control List). Fixed error in examples referring to X-Auth-Key where it should be X-Auth-Token. Added section numbers.
Jan 4, 2011	• Expanded authentication information for UK release. Added "delimiter" as a Query Parameter and server-side object copy example.
May 5, 2008	• Initial Release.

1.3. Additional Resources

You can download the most current version of this document from the OpenStack Docs website at http://docs.openstack.org.

If you would like more information about the Rackspace implementation of the OpenStack Object Storage service, visit this address: http://www.rackspacecloud.com/cloud_hosting_products/files. Related documents are available at the same site, as are links to Rackspace's official support channels, including knowledge base articles, forums, phone, chat, and email.

2. General API Information

API Operations Reference Summary

Storage Accounts

Verb	URI	Description
GET	`/account`	List containers
HEAD	`account`	Retrieve account metadata

Storage Containers

Verb	URI	Description
GET	`/account/container`	List objects
PUT	`/account/container`	Create container
DELETE	`/account/container`	Delete container
HEAD	`/account/container`	Retrieve container metadata

Storage Objects

Verb	URI	Description
GET	`/account/container/object`	Retrieve object
PUT	`/account/container/object`	Create/Update Object
PUT	`/account/container/object`	Chunked transfer encoding
DELETE	`/account/container/object`	Delete container
HEAD	`/account/container/object`	Retrieve object metadata
POST	`/account/container/object`	Update object metadata

2.1. Authentication

Client authentication is provided via a ReST interface using the **GET** method, with `v1.0` supplied as the path. Additionally, two headers are required, `X-Auth-User` and `X-Auth-Key` with values for the username and API Access Key respectively.

Each ReST request against the OpenStack Object Storage system requires the inclusion of a specific authorization token HTTP x-header, defined as `X-Auth-Token`. Clients obtain this token, along with the Cloud Servers API URL, by first using an authentication service and supplying a valid username and API access key.

Request

To authenticate, you must supply your username and API access key in x-headers:

- Use your OpenStack Object Storage (Swift) username as the username for the API. Place it in the `X-Auth-User` x-header.
- Obtain your API access key from authentication service you chose when installing. You have some options for auth, including tempauth (which is included with Swift), swauth (an auth service for Swift as WSGI middleware that uses Swift itself as a backing store

that is provided via download from Github), the OpenStack Identity Service (project named Keystone), or you can use your own authentication system. Place your access key in the `X-Auth-Key` x-header.

Example 2.1. Authentication Request

```
GET /v1.0 HTTP/1.1
Host: auth.api.yourcloud.com
X-Auth-User: jdoe
X-Auth-Key: a86850deb2742ec3cb41518e26aa2d89
```

Response

When authentication is successful, an HTTP status 204 (No Content) is returned with the `X-Storage-Url` and `X-Auth-Token` headers. Any 2xx response is a good response. For example, a 202 response means the request has been accepted. Also, additional `X-` headers may be returned. These additional headers are related to other Rackspace services and can be ignored. An HTTP status of 401 (Unauthorized) is returned upon authentication failure. All subsequent container/object operations against OpenStack Object Storage should be made against the URI specified in `X-Storage-Url` and must include the `X-Auth-Token` header.

Example 2.2. Authentication Response

```
HTTP/1.1 204 No Content
Date: Mon, 12 Nov 2010 15:32:21 GMT
Server: Apache
X-Storage-Url: https://storage.swiftdrive.com/v1/CF_xer7_34
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
```

The `X-Storage-Url` will need to be parsed and used in the connection and request line of all subsequent requests against Object Storage. In the example response above, users connecting to OpenStack Object Storage would send most container/object requests with a host header of `storage.swiftdrive.com` and the request line's version and account as `/v1/CF_xer7_34`. Note that authentication tokens are valid for a 24 hour period for many authentication configurations.

2.2. Overview of API Operations

The OpenStack Object Storage API is implemented as a set of ReSTful (Representational State Transfer) web services. All authentication and container/object operations can be performed with standard HTTP calls. See the Wikipedia article on ReST for more information

The following constraints apply to the ReST API's HTTP requests:

- Maximum number of HTTP headers per request: 90

- Maximum length of all HTTP headers: 4096 bytes
- Maximum length per HTTP request line: 8192 bytes
- Maximum length of HTTP request: 5 gigabytes
- Maximum length of container name: 256 bytes
- Maximum length of object name: 1024 bytes

Container and object names should be properly URL-encoded prior to interacting with the ReST interface (the language APIs handle URL encoding/decoding) and the container and object names must be UTF-8 encoded. The length restrictions should be checked against the URL-encoded string.

Each ReST request against the OpenStack Object Storage system requires the inclusion of a specific *authorization token* HTTP header defined as `X-Auth-Token`. Clients obtain this token, along with the OpenStack Object Storage URLs, by first using the Authentication service and supplying a valid Username and API Access Key.

The ReST service identified with `X-Storage-Url` is used for managing the data stored in the system. Example operations are creating containers and uploading objects.

In the following sections, the purpose of each HTTP method depends upon which service the call is made against. For example, a **PUT** request against `X-Storage-Url` can be used to create a container or upload an object.

The language-specific APIs mask this system separation from the programmer. They simply create a container and mark it *public* and it handles calling out to the appropriate back-end services using the appropriate ReST API.

Note

All requests to authenticate and operate against OpenStack Object Storage are performed using SSL over HTTP (HTTPS) on TCP port 443.

3. API Operations for Storage Services

The following section describes the ReST API for interacting with the storage component of OpenStack Object Storage. All requests will be directed to the host and URL described in the `X-Storage-Url` HTTP header obtained during successful authentication.

The following are some requirements for the use of the storage services:

- Container names cannot exceed 256 bytes and cannot contain a '/' character
- Object names cannot exceed 1024 bytes and have no character restrictions
- Object and container names must be URL-encoded and UTF-8 encoded

The following sections describe the actions that may be performed within the storage system. The first section addresses actions that can be taken on the account level of the storage system. The second section addresses actions that may be performed on containers. The third section addresses actions that may be performed on objects.

3.1. Storage Account Services

The following operations can be performed at the account level of the URL. For example, the URL for the requests below will end with the OpenStack Object Storage account string:

Example 3.1. Storage Account HTTP Request: General Structure

```
METHOD /v1/<account> HTTP/1.1
```

3.1.1. List Containers

GET operations against the `X-Storage-Url` for an account are performed to retrieve a list of existing storage containers ordered by name. The sort order for the name is based on a binary comparison, a single built-in collating sequence that compares string data using SQLite's memcmp() function, regardless of text encoding. The following list describes the optional query parameters that are supported with this request.

Query Parameters

`limit`	For an integer value *n*, limits the number of results to *n* values.
`marker`	Given a string value *x*, return container names greater in value than the specified marker.
`end_marker`	Given a string value *x*, return container names less in value than the specified marker.
`format`	Specify either `json` or `xml` to return the respective serialized response.

At this time, a `prefix` query parameter is not supported at the account level.

Example 3.2. Containers List Request

```
GET /<api version>/<account> HTTP/1.1
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

A list of containers is returned in the response body, one container per line. A 204 (No Content) HTTP return code will be passed back if the account has no containers.

Example 3.3. Containers List Response

```
HTTP/1.1 200 Ok
Date: Thu, 07 Jun 2010 18:57:07 GMT
Server: Apache
Content-Type: text/plain; charset=UTF-8
Content-Length: 32
```

```
images
movies
documents
backups
```

3.1.1.1. Serialized List Output

If a `format=xml` or `format=json` argument is appended to the storage account URL, the service will serve extended container information serialized in the chosen format. The sample responses below are formatted for readability.

Example 3.4. Containers Details Request: JSON

```
GET /<api version>/<account>?format=json HTTP/1.1
Host: storage.swiftdrive.com
Content-Length: 0
X-Storage-Token: 182f9c0af0e828cfe3281767d29d19f4
```

Example 3.5. Containers Details Response: JSON

```
HTTP/1.1 200 OK
Date: Tue, 25 Nov 2008 19:39:13 GMT
Server: Apache
Content-Type: application/json; charset=utf-8
```

```
[
  {"name":"test_container_1", "count":2, "bytes":78},
  {"name":"test_container_2", "count":1, "bytes":17}
]
```

Example 3.6. Containers Details Request: XML

```
GET /<api version>/<account>?format=xml HTTP/1.1
Host: storage.swiftdrive.com
Content-Length: 0
X-Storage-Token: 182f9c0af0e828cfe3281767d29d19f4
```

Example 3.7. Containers Details Response: XML

```
HTTP/1.1 200 OK
Date: Tue, 25 Nov 2008 19:42:35 GMT
Server: Apache
Content-Type: application/xml; charset=utf-8
```

```
<?xml version="1.0" encoding="UTF-8"?>

<account name="MichaelBarton">
  <container>
    <name>test_container_1</name>
    <count>2</count>
    <bytes>78</bytes>
  </container>
  <container>
    <name>test_container_2</name>
    <count>1</count>
    <bytes>17</bytes>
  </container>
</account>
```

3.1.1.2. Controlling a Large List of Containers

The system returns a maximum of 10,000 container names per request. To retrieve subsequent container names, another request must be made with the 'marker' parameter. The marker indicates where the last list left off; the system returns container names greater than this marker, up to 10,000 again. Note that the ‘marker’ value should be URL-encoded prior to sending the HTTP request.

If 10,000 is larger than desired, use the 'limit' parameter.

If the number of container names returned equals the limit given (or 10,000 if no limit is given), you may assume there are more container names.

Example 3.8. List Large Number of Containers

For example, let's use a listing of five container names

```
apples
bananas
```

```
kiwis
oranges
pears
```

We'll use a limit of two to show how things work:

```
GET /<api version>/<account>?limit=2
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

```
apples
bananas
```

Since we received two items back, we can assume there are more container names to list, so we make another request with a marker of the last item returned:

```
GET /<api version>/<account>?limit=2&marker=bananas
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

```
kiwis
oranges
```

Again, two items are returned; there may be more:

```
GET /<api version>/<account>?limit=2&marker=oranges
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

```
pears
```

With this one-item response we received less than the limit number of container names, indicating that this is the end of the list.

By using `end_marker` we can limit the result set to container names less than the given value.

```
GET /<api version><account>?end_marker=oranges
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

```
apples
bananas
kiwis
```

3.1.2. Retrieve Account Metadata

HEAD operations against an account are performed to retrieve the number of containers and the total bytes stored in OpenStack Object Storage for the account. This information is returned in two custom headers, `X-Account-Container-Count` and `X-Account-Bytes-Used`. Since the storage system is designed to store large amounts of data, care should be taken when representing the total bytes response as an integer; when possible, convert it to a 64-bit unsigned integer if your platform supports that primitive type.

Example 3.9. Account Metadata Request

```
HEAD /<api version>/<account> HTTP/1.1
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

The HTTP return code will be 204 (No Content) if the request succeeds. A 401 (Unauthorized) will be returned for an invalid account or access key.

Example 3.10. Account Metadata Response

```
HTTP/1.1 204 No Content
Date: Thu, 07 Jun 2010 18:57:07 GMT
Server: Apache
X-Account-Container-Count: 3
X-Account-Bytes-Used: 323479
```

3.1.3. Create/Update Account Metadata

You can associate custom metadata headers with the account level URI. These headers must take the format `X-Account-Meta-*`.

To create or update an account metadata header use the **POST** query. Subsequent requests for the same key/value pair overwrite the previous value.

Example 3.11. Update Account Metadata Request

```
POST /<api version>/<account> HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
X-Account-Meta-Book: MobyDick
X-Account-Meta-Subject: Whaling
```

No response body is returned. A status code of 204 (No Content) indicates success.

Example 3.12. Update Account Metadata Response

```
HTTP/1.1 204 No Content
Content-Length: 0
Content-Type: text/html; charset=UTF-8
Date: Sat, 09 Jun 2012 19:16:29 GMT
```

To confirm your metadata changes, perform a **HEAD** request on the account. Do not send the metadata in your **HEAD** request.

Example 3.13. View Account Metadata Request

```
HEAD /<api version>/<account> HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

Example 3.14. View Account Metadata Response

```
HTTP/1.1 204 No Content
X-Account-Meta-Book: MobyDick
X-Account-Meta-Subject: Whaling
X-Account-Object-Count: 5
X-Timestamp: 1323466696.21566
X-Account-Container-Count: 5
X-Account-Bytes-Used: 46988
Accept-Ranges: bytes
Content-Length: 0
Date: Sat, 09 Jun 2012 19:16:59 GMT
```

3.1.4. Delete Account Metadata

To delete a metadata header send an empty value for that particular header, e.g. `X-Account-Meta-Book:`.

If the tool you're using to communicate with Swift doesn't support sending empty headers (e.g. older versions of curl) send the header "X-Remove-Account-Meta-*name*: *arbitrary value*", e.g. `X-Remove-Account-Meta-Book: x`. The *value* is ignored.

Example 3.15. Account Metadata Delete Request

```
POST /<api version>/<account> HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
X-Remove-Account-Meta-Book: x
```

No response body is returned. A status code of 204 (No Content) indicates success.

3.2. Storage Container Services

This section documents the ReST operations that can be performed on containers. All operations are valid HTTP request methods and will resemble this format:

Example 3.16. Storage Container HTTP Request: General Structure

```
METHOD /v1/<account>/<container> HTTP/1.1
```

3.2.1. List Objects

GET operations against a storage container name are performed to retrieve a list of objects stored in the container. Additionally, there are a number of optional query parameters that can be used to refine the list results.

A request with no query parameters will return the full list of object names stored in the container, up to 10,000 names. Optionally specifying the query parameters will filter the full list and return a subset of objects.

Query Parameters

`limit`	For an integer value *n*, limits the number of results to at most *n* values.
`marker`	Given a string value *x*, return object names greater in value than the specified marker.
`end_marker`	Given a string value *x*, return object names less in value than the specified marker.
`prefix`	For a string value *x*, causes the results to be limited to object names beginning with the substring *x*.
`format`	Specify either `json` or `xml` to return the respective serialized response.
`delimiter`	For a character *c*, return all the object names nested in the container (without the need for the directory marker objects).

Example 3.17. Objects List Request

```
GET /<api version>/<account>/<container>[?parm=value] HTTP/1.1
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

A list of objects is returned in the response body, one object name per line. A 204 (No Content) HTTP return code will be passed back if the container is empty. If the container does not exist, a 404 code will return. If an incorrect account is specified, the HTTP return code will be 404 (Not Found).

Example 3.18. Objects List Response

```
HTTP/1.1 200 Ok
Date: Thu, 07 Jun 2010 18:50:19 GMT
Server: Apache
Content-Type: text/plain; charset=UTF-8
Content-Length: 171
```

```
kate_beckinsale.jpg
How To Win Friends And Influence People.pdf
moms_birthday.jpg
poodle_strut.mov
Disturbed - Down With The Sickness.mp3
army_of_darkness.avi
the_mad.avi
```

3.2.1.1. Serialized List Output

If a `format=xml` or `format=json` argument is appended to the storage account URL, the service will serve extended object information serialized in the chosen format. Other than the `?format=xml|json` parameter, it will return the same status/errors codes. The sample responses below are formatted for readability.

Example 3.19. Objects Details Request: JSON

```
GET /<api version>/<account>/<container>?format=json HTTP/1.1
Host: storage.swiftdrive.com
Content-Length: 0
X-Storage-Token: 182f9c0af0e828cfe3281767d29d19f4
```

Example 3.20. Objects Details Response: JSON

```
HTTP/1.1 200 OK
Date: Tue, 25 Nov 2008 19:39:13 GMT
Server: Apache
Content-Length: 387
Content-Type: application/json; charset=utf-8
```

```
[
  {"name":"test_obj_1",
   "hash":"4281c348eaf83e70ddce0e07221c3d28",
   "bytes":14,
   "content_type":"application\/octet-stream",
   "last_modified":"2009-02-03T05:26:32.612278"},
  {"name":"test_obj_2",
   "hash":"b039efe731ad111bc1b0ef221c3849d0",
   "bytes":64,
   "content_type":"application\/octet-stream",
   "last_modified":"2009-02-03T05:26:32.612278"},
]
```

Example 3.21. Objects Details Request: XML

```
GET /<api version>/<account>/<container>?format=xml HTTP/1.1
Host: storage.swiftdrive.com
X-Storage-Token: 182f9c0af0e828cfe3281767d29d19f4
```

Example 3.22. Objects Details Request: XML

```
HTTP/1.1 200 OK
Date: Tue, 25 Nov 2008 19:42:35 GMT
Server: Apache
Content-Length: 643
Content-Type: application/xml; charset=utf-8
```

```
<?xml version="1.0" encoding="UTF-8"?>

<container name="test_container_1">
  <object>
    <name>test_object_1</name>
    <hash>4281c348eaf83e70ddce0e07221c3d28</hash>
    <bytes>14</bytes>
    <content_type>application/octet-stream</content_type>
    <last_modified>2009-02-03T05:26:32.612278</last_modified>
  </object>
  <object>
    <name>test_object_2</name>
    <hash>b039efe731ad111bc1b0ef221c3849d0</hash>
    <bytes>64</bytes>
    <content_type>application/octet-stream</content_type>
    <last_modified>2009-02-03T05:26:32.612278</last_modified>
  </object>
</container>
```

3.2.1.2. Controlling a Large List of Objects

The system returns a maximum of 10,000 object names per request. To retrieve subsequent object names, another request must be made with the 'marker' parameter. The marker indicates where the last list left off and the system returns object names greater than this marker, up to 10,000 again. Note that the ‘marker’ value should be URL encoded prior to sending the HTTP request.

If 10,000 is larger than desired, a 'limit' parameter may be given.

If the number of object names returned equals the limit given (or 10,000 if no limit is given), it can be assumed there are more object names to be listed. If the container name list is exactly divisible by the limit, the last request will simply have no content.

Example 3.23. List Large Number of Objects

For an example, let's use a listing of five object names:

```
gala
grannysmith
honeycrisp
jonagold
reddelicious
```

We'll use a limit of two to show how things work:

```
GET /<api version>/<account>/<container>?limit=2
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

```
gala
grannysmith
```

Since we received two items back, we can assume there are more object names to list. So, we make another request with a marker of the last item returned:

```
GET /<api version>/<account>/<container>?limit=2&marker=grannysmith
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

```
honeycrisp
jonagold
```

Again we have two items returned; there may be more:

```
GET /<api version>/<account>/<container>?limit=2&marker=jonagold
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

```
reddelicious
```

Now we received less than the limit number of object names, indicating that we have the complete list.

By using `end_marker` we can limit the result set to object names less than the given value.

```
GET /<api version><account>/<container>?end_marker=jonagold
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

```
gala
grannysmith
honeycrisp
```

3.2.1.3. Pseudo-Hierarchical Folders/Directories

Although you cannot nest directories in OpenStack Object Storage, you can simulate a hierarchical structure within a single container by adding forward slash characters (`/`) in the object name. To navigate the pseudo-directory structure, you may use the `delimiter` query parameter. See the below examples for an illustration.

Note

In the example below, the objects reside in a container called `backups`. Within that container, the objects are organized in a pseudo-directory called `photos`. Keep in mind that the container name is not displayed in the example, but that it is a part of the object URLs. For instance, the URL of the picture `me.jpg` is `https://storage.swiftdrive.com/v1/CF_xer7_343/backups/photos/me.jpg`.

Example 3.24. Pseudo-Hierarchical Folders/Directories

To display a list of all the objects in the storage container, use **GET** without a `delimiter` or `prefix`.

```
GET /v1/AccountString/backups
```

The system returns status code 200 (OK) and the requested list of the objects.

```
photos/animals/cats/persian.jpg
photos/animals/cats/siamese.jpg
photos/animals/dogs/corgi.jpg
photos/animals/dogs/poodle.jpg
photos/animals/dogs/terrier.jpg
photos/me.jpg
photos/plants/fern.jpg
photos/plants/rose.jpg
```

Use the delimiter parameter to limit the displayed results. Any character may be used as a delimiter. However, to use `delimiter` with pseudo-directories, use the parameter slash (`/`).

```
GET /v1/AccountString/backups?delimiter=/
```

The system returns status code 200 (OK) and the requested matching objects. Because we use the slash, only the pseudo-directory `photos/` displays. Keep in mind that the returned values from a slash `delimiter` query are not real objects. They have a content-type of application/directory and are in a subdir section of json and xml results.

```
photos/
```

Use the `prefix` parameter with the `delimiter` parameter to view the objects inside a pseudo-directory, including further nested pseudo-directories.

```
GET /v1/AccountString/backups?prefix=photos/&delimiter=/
```

The system returns status code 200 (OK) and the objects and pseudo-directories within the top level pseudo-directory.

```
photos/animals/
photos/me.jpg
photos/plants/
```

There is no limit to the amount of nested pseudo-directories you can create. In order to navigate through them, use a longer `prefix` parameter coupled with the `delimiter` parameter. In the sample output below, there is a pseudo-directory called `dogs` within the pseudo-directory `animals`. In order to navigate directly to the files contained within `dogs`, enter the below command.

```
GET /v1/AccountString/backups?prefix=photos/animals/dogs/&delimiter=/
```

The system returns status code 200 (OK) and the objects and pseudo-directories within the nested pseudo-directory.

```
photos/animals/dogs/corgi.jpg
photos/animals/dogs/poodle.jpg
photos/animals/dogs/terrier.jpg
```

3.2.2. Create Container

PUT operations against a storage container are used to create that container.

Containers are storage compartments for your data. The URL encoded name must be less than 256 bytes and cannot contain a forward slash '/' character.

Containers can be assigned custom metadata by including additional HTTP headers on the **PUT** request. The custom metadata is assigned to a container via HTTP headers identified with the `X-Container-Meta-` prefix.

Example 3.25. Container Create Request

```
PUT /<api version>/<account>/<container> HTTP/1.1
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

No content is returned. A status code of 201 (Created) indicates that the container was created as requested. Container **PUT** requests are idempotent and a code of 202 (Accepted) is returned when the container already existed. If you request a **PUT** to a container with an `X-Container-Meta-` prefix in the header, your **GET**/**HEAD** request responses carry the metadata prefix from the container in subsequent requests.

Example 3.26. Container Create Response

```
HTTP/1.1 201 Created
Date: Thu, 07 Jun 2007 18:50:19 GMT
Server: Apache
Content-Type: text/plain; charset=UTF-8
```

Using custom container metadata, you can create information in the header to effectively "tag" a container with metadata. The container metadata restrictions are the same as object metadata: you can have 4096 bytes maximum overall metadata, 90 distinct metadata items at the most. Each may have a 128 character name length with a 256 max value length each. Any valid UTF-8 http header value is allowed for metadata, however we recommend that you URL-encode any non-ASCII values using a "%" symbol, followed by the two-digit hexadecimal representation of the ISO-Latin code for the character.

Example 3.27. Container Create Request with Metadata

```
PUT /<api version>/<account>/<container> HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
X-Container-Meta-InspectedBy: JackWolf
```

No content is returned. A status code of 201 (Created) indicates that the container was created as requested. Container **PUT** requests are idempotent and a code of 202 (Accepted) is returned if the container existed prior to the request. If you request a **PUT** to a container with an `X-Container-Meta-` prefix in the header, your **GET**/**HEAD** request responses carry the metadata prefix from the container in subsequent requests.

Example 3.28. Container Create Response

```
HTTP/1.1 201 Created
Date: Thu, 07 Jun 2010 18:50:19 GMT
Server: Apache
Content-Type: text/plain; charset=UTF-8
```

3.2.3. Delete Container

DELETE operations against a storage container permanently remove it. The container must be empty before it can be deleted.

A **HEAD** request against the container can be used to determine if it contains any objects.

Example 3.29. Container Delete Request

```
DELETE /<api version>/<account>/<container> HTTP/1.1
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

No content is returned. A status code of 204 (No Content) indicates success, 404 (Not Found) is returned if the requested container was not found, and a 409 (Conflict) if the container is not empty. No response body is generated.

Example 3.30. Container Delete Response

```
HTTP/1.1 204 No Content
Date: Thu, 07 Jun 2010 18:57:07 GMT
Server: Apache
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
```

3.2.4. Retrieve Container Metadata

HEAD operations against a storage container are used to determine the number of objects, and the total bytes of all objects stored in the container. Since the storage system is designed to store large amounts of data, care should be taken when representing the total bytes response as an integer; when possible, convert it to a 64-bit unsigned integer if your platform supports that primitive type.

Example 3.31. Container Metadata Request

```
HEAD /<api version>/<account>/<container> HTTP/1.1
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

The HTTP return code will be 204 (No Content) if the container exists, and 404 (Not Found) if it does not. The object count and utilization are returned in the `X-Container-Object-Count` and `X-Container-Bytes-Used` headers respectively.

Example 3.32. Container Metadata Response

```
HTTP/1.1 204 No Content
Date: Wed, 11 Jul 2010 19:37:41 GMT
Content-type: text/html
X-Container-Object-Count: 7
X-Container-Bytes-Used: 413
X-Container-Meta-InspectedBy: JackWolf
```

3.2.5. Create/Update Container Metadata

You may create any custom or arbitrary metadata headers as you find useful. They must, however, take the format `X-Container-Meta-`.

To create or update the arbitrary container metadata, use the **POST** query. Subsequent requests of the same key/value pair overwrites the previous value.

Example 3.33. Update Container Metadata Request

```
POST /<api version>/<account>/<container>/ HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
X-Container-Meta-Book: MobyDick
X-Container-Meta-Subject: Whaling
```

No response body is returned. A status code of 204 (No Content) indicates success; status 404 (Not Found) is returned when the requested container does not exist.

Example 3.34. Update Container Metadata Response

```
HTTP/1.1 204 No Content
Date: Thu, 07 Mar 2012 20:42:51 GMT
Server: Apache
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
```

To confirm your metadata changes, perform a **HEAD** request on the container. Do not send the metadata in your **HEAD** request.

Example 3.35. View Container Metadata Request

```
HEAD /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

Example 3.36. View Container Metadata Response

```
HTTP/1.1 204 No Content
X-Container-Object-Count: 0
X-Trans-Id: tx028b40d228534c759f4d5fa69f8cf7fd
X-Container-Meta-Book: MobyDick
X-Container-Meta-Subject: Whaling
Accept-Ranges: bytes
Date: Mon, 12 Mar 2012 16:40:20 GMT
Content-Length: 0
X-Container-Bytes-Used: 0
```

3.2.6. Delete Container Metadata

To delete a metadata header send an empty value for that particular header, such as `X-Container-Meta-Book:`.

If the tool you're using to communicate with Object Storage doesn't support sending empty headers (older versions of curl) send the header "X-Remove-Container-Meta-*name*: *`arbitrary value`*". For example, send a header like `X-Remove-Container-Meta-Book: x`. The *value* (x in this example) is ignored.

Example 3.37. Container Metadata Delete Request

```
POST /<api version>/<account>/<container> HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
X-Remove-Container-Meta-Book: x
```

No response body is returned. A status code of 204 (No Content) indicates success.

3.3. Create Static Website

You may use your swift account to create a static website on the World Wide Web. This mode is normally only active for anonymous requests. If you want to use it with authenticated requests, set the header `X-Web-Mode` to TRUE on the request. The `staticweb` filter should be added to the pipeline in your `/etc/swift/proxy-server.conf` file just after any auth middleware. Beneath the pipeline, the `staticweb` middleware configuration must be added. For example:

```
[DEFAULT]
...

[pipeline:main]
pipeline = healthcheck cache tempauth staticweb proxy-server

...

[filter:staticweb]
use = egg:swift#staticweb
# Seconds to cache container x-container-meta-web-* header values.
# cache_timeout = 300
# You can override the default log routing for this filter here:
# set log_name = staticweb
# set log_facility = LOG_LOCAL0
# set log_level = INFO
# set access_log_name = staticweb
# set access_log_facility = LOG_LOCAL0
# set access_log_level = INFO
# set log_headers = False
```

Your publicly readable containers will be checked for two headers, `X-Container-Meta-Web-Index` and `X-Container-Meta-Web-Error`. (The latter header is discussed below, under Set Error Pages for Static Website.) With `X-Container-Meta-Web-Index`, you determine the index file (or default page served, such as `index.html`) displays your website. When someone initially enters your site, they don't have to specify the index file; index.html file displays automatically. If you create sub-directories for your site–which you do by creating pseudo-directories in your container–the index page displays by default for each sub-directory. If your pseduo-directory does not have a file with the same name as your index file, visits to the sub-directory return a 404 error.

You also have the option of displaying a list of files in your pseudo-directory instead of a web page. You do this by setting the `X-Container-Meta-Web-Listings` header to TRUE. You may add style to your file listing by setting `X-Container-Meta-Web-Listings-CSS:` to a stylesheet (i.e., `lists.css`).

3.3.1. Static Web Middleware via swift

Example 3.38. Make Container Publicly Readable

Make the container publicly readable. Once the container is publicly readable, you may access your objects directly, but you will need to set the index file to browse the main site URL and its sub-directories.

```
swift post -r '.r:*' container
```

Example 3.39. Set Site's Index File

Set the index file. In this case, `index.html` is the default file displayed when the site displays.

```
swift post -m 'web-index:index.html' container
```

Example 3.40. Enable File Listing

Turn on file listing. If you do not set the index file, list the objects in the container. Instructions on styling the list with the CSS follow.

```
swift post -m 'web-listings: true' container
```

Example 3.41. Enable CSS for File Listing

Style the file listing.

```
swift post -m 'web-listings-css:listings.css' container
```

3.3.2. Set Error Pages for Static Website

You may create and set custom error pages for visitors to your website; currently, only 401 (Unauthorized) and 404 (Not Found) errors are supported. To do this, set the metadata header, `X-Container-Meta-Web-Error`.

Error pages are served with the <status> code prepended to the name of the error page you set. For instance, if you set `X-Container-Meta-Web-Error` to `error.html`, 401 errors will display the page `401error.html`. Similarly, 404 errors will display `404error.html`. You must have both of these pages created in your container when you set the `X-Container-Meta-Web-Error` metadata, or your site will display generic error pages.

You need only set the `X-Container-Meta-Web-Error` metadata once for your entire static website.

Example 3.42. Set Error Pages for Static Website

```
swift post -m 'web-error:error.html' container
```

Any 2xx response indicates success.

3.4. Storage Object Services

An object represents the data and any metadata for the files stored in the system. Through the ReST interface, metadata for an object can be included by adding custom HTTP headers to the request and the data payload as the request body. Objects cannot exceed 5GB and must have names that do not exceed 1024 bytes after URL encoding. However, objects larger than 5GB can be segmented and then concatenated together so that you can upload 5 GB segments and download a single concatenated object. You can work with the segments and manifests directly with HTTP requests.

3.4.1. Retrieve Object

GET operations against an object are used to retrieve the object's data.

Note that you can perform conditional **GET** requests by using certain HTTP headers as documented in RFC 2616. OpenStack Object Storage supports the following headers:

RFC 2616: http://www.ietf.org/rfc/rfc2616.txt

- If-Match
- If-None-Match
- If-Modified-Since
- If-Unmodified-Since

It is also possible to fetch a portion of data using the HTTP `Range` header. At this time, OpenStack Object Storage does not support the full specification for `Range` but basic support is provided. OpenStack Object Storage only allows a single range that includes OFFSET and/or LENGTH. We support a sub-set of `Range` and do not adhere to the full RFC-2616 specification. We support specifying OFFSET-LENGTH where either OFFSET or LENGTH can be optional (not both at the same time). The following are supported forms of the header:

- `Range: bytes=-5` - last five bytes of the object
- `Range: bytes=10-15` - the five bytes after a 10-byte offset
- `Range: bytes=32-` - all data after the first 32 bytes of the object

Example 3.43. Retrieve Object Request

```
GET /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

The object's data is returned in the response body. Object metadata is returned as HTTP headers. A status of 200 (Ok) indicates success; status 404 (Not Found) is returned if no such object exists.

Example 3.44. Retrieve Object Response

```
HTTP/1.1 200 Ok
Date: Wed, 11 Jul 2010 19:37:41 GMT
Server: Apache
Last-Modified: Fri, 12 Jun 2010 13:40:18 GMT
ETag: b0dffe8254d152d8fd28f3c5e0404a10
Content-type: text/html
Content-Length: 512000
```

```
[ ... ]
```

3.4.2. Create/Update Object

PUT operations are used to write, or overwrite, an object's content and metadata.

You can ensure end-to-end data integrity by including an MD5 checksum of your object's data in the ETag header. You are not required to include the ETag header, but it is recommended to ensure that the storage system successfully stored your object's content.

You can cause an object to expire after a certain date by using the `X-Delete-At` or `X-Delete-After` headers during an object **PUT** operation. When Cloud Files detects one of these headers, the system automatically stops serving that object at the specified time and shortly after the expiration date, it removes the object from the storage system.

The HTTP response will include the MD5 checksum of the data written to the storage system. If you do not send the ETag in the request, you should compare the value returned with your content's MD5 locally to perform the end-to-end data validation on the client side. For segmented objects, the ETag is the MD5 sum of the concatenated string of ETags for each of the segments in the manifest, which only offers change detection but not direct comparison.

Objects can be assigned custom metadata by including additional HTTP headers on the **PUT** request.

The object can be created with custom metadata via HTTP headers identified with the `X-Object-Meta-` prefix.

Example 3.45. Create/Update Object Request

```
PUT /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
ETag: 8a964ee2a5e88be344f36c22562a6486
Content-Length: 512000
X-Object-Meta-PIN: 1234
```

```
[ ... ]
```

No response body is returned. A status code of 201 (Created) indicates a successful write; status 411 (Length Required) denotes a missing `Content-Length` or `Content-Type` header in the request. If the MD5 checksum of the data written to the storage system does NOT match the (optionally) supplied ETag value, a 422 (Unprocessable Entity) response is returned.

Example 3.46. Create/Update Object Response

```
HTTP/1.1 201 Created
Date: Thu, 07 Jun 2010 18:57:07 GMT
Server: Apache
ETag: d9f5eb4bba4e2f2f046e54611bc8196b
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
```

3.4.2.1. Large Object Creation

Objects that are larger than 5GB must be segmented, prior to upload. You then upload the segments like you would any other object and create a manifest object telling OpenStack Object Storage how to find the segments of the large object. The segments remain individually addressable, but retrieving the manifest object streams all the segments concatenated. There is no limit to the number of segments that can be a part of a single large object.

In order to ensure the download works correctly, you must upload all the object segments to the same container, ensure each object name has a common prefix where their names sort in the order they should be concatenated. You also create and upload a manifest file. The manifest file is simply a zero-byte file with the extra X-Object-Manifest: <container>/<prefix> header, where <container> is the container the object segments are in and <prefix> is the common prefix for all the segments.

It is best to upload all the segments first and then create or update the manifest. With this method, the full object will not be available for downloading until the upload is complete. Also, you can upload a new set of segments to a second location and then update the manifest to point to this new location. During the upload of the new segments, the original manifest will still be available to download the first set of segments.

Example 3.47. Upload Segment of a Large Object

```
PUT /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
ETag: 8a964ee2a5e88be344f36c22562a6486
Content-Length: 1
X-Object-Meta-PIN: 1234
```

```
s
```

No response body is returned. A status code of 201 (Created) indicates a successful write; status 411 (Length Required) denotes a missing `Content-Length` or `Content-Type`

header in the request. If the MD5 checksum of the data written to the storage system does NOT match the (optionally) supplied ETag value, a 422 (Unprocessable Entity) response is returned.

You can continue uploading segments like this example shows, prior to uploading the manifest.

Example 3.48. Upload Next Segment of the Large Object

```
PUT /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
ETag: 8a964ee2a5e88be344f36c22562a6486
Content-Length: 1
X-Object-Meta-PIN: 1234
```

```
w
```

Next, upload the manifest you created that indicates the container the object segments reside within. Note that uploading additional segments after the manifest is created will cause the concatenated object to be that much larger but you do not need to recreate the manifest file for subsequent additional segments.

Example 3.49. Upload Manifest

```
PUT /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
Content-Length: 0
X-Object-Meta-PIN: 1234
X-Object-Manifest: container/object/segments
```

```
[...]
```

The response's Content-Type for a **GET** or **HEAD** on the manifest will be the same as the Content-Type set during the PUT request that created the manifest. You can easily change the Content-Type by reissuing the **PUT** request.

3.4.2.2. Chunked Transfer Encoding

Users can upload data without needing to know in advance the amount of data to be uploaded. Users can do this by specifying an HTTP header of `Transfer-Encoding: chunked` and not using a `Content-Length` header. A good use of this feature would be doing a DB dump, piping the output through gzip, then piping the data directly into OpenStack Object Storage without having to buffer the data to disk to compute the file size. If users attempt to upload more than 5GB with this method, the server will close the TCP/IP connection after 5GB and purge the customer data from the system. Users must take responsibility for ensuring the data they transfer will be less than 5GB or for splitting it into 5GB chunks, each in its own storage object. If you have files that are larger

than 5GB and still want to use Object Storage, you can segment them prior to upload, upload them to the same container, and then use a manifest file to allow downloading of a concatenated object containing all the segmented objects, concatenated as a single object.

Example 3.50. Upload Unspecified Quantity of Content

```
PUT /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
Transfer-Encoding: chunked
X-Object-Meta-PIN: 1234
```

```
19
A bunch of data broken up
D
 into chunks.
0
```

3.4.3. Assigning CORS Headers to Requests

CORS is a specification that stands for Cross-Origin Resource Sharing. It defines how browsers and servers communicate across origins using HTTP headers, such as those assigned by Cloud Files API requests. These headers are supported with the Cloud Files API. You can read more about the definition of the Access-Control- response headers and Origin response header at www.w3.org/TR/access-control/.

- Access-Control-Allow-Credentials
- Access-Control-Allow-Methods
- Access-Control-Allow-Origin
- Access-Control-Expose-Headers
- Access-Control-Max-Age
- Access-Control-Request-Headers
- Access-Control-Request-Method
- Origin

These headers can be assigned to objects only.

Example 3.51. Assign CORS Header

In the example, the origin header is assigned that indicates where the file came from. This allows you to provide security that requests to your Cloud Files repository are indeed from the correct origination:

```
PUT /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.clouddrive.com
```

```
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
Origin: http://storage.clouddrive.com
```

3.4.4. Enabling File Compression with the Content-Encoding Header

The Content-Encoding header allows a file to be compressed without losing the identity of the underlying media type of the file, for example, a video.

Example 3.52. Content-Encoding Header Example

In the example, the content-encoding header is assigned with an attachment type that indicates how the file should be downloaded:

```
PUT /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
Content-Type: video/mp4
Content-Encoding: gzip
```

3.4.5. Enabling Browser Bypass with the Content-Disposition Header

When an object is assigned the Content-Disposition header you can override a browser's default behavior for a file so that the downloader saves the file rather than displaying it using default browser settings.

Example 3.53. Content-Disposition Header Example

In the example, the content-encoding header is assigned with an attachment type that indicates how the file should be downloaded.

```
PUT /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
Content-Type: image/tiff
Content-Disposition: attachment; filename=platmap.tif
```

3.4.6. Expiring Objects with the X-Delete-After and X-Delete-At Headers

When an object is assigned either an `X-Delete-After` or `X-Delete-At` header when doing a **PUT** or **POST** on the object, it is scheduled for deletion. This feature is helpful for objects you do not want to permanently store, such as log files, recurring full backups of a dataset, or documents or images you know will be outdated at a future time.

The `X-Delete-At` header requires a Unix Epoch timestamp, in integer form; for example: 1348691905 represents Wed, 26 Sep 2012 20:38:25 GMT. By setting the header to a specific

Epoch time, you indicate when you want the object to expire, not be served, and be deleted completely from the storage system.

The `X-Delete-After` header takes an integer number of seconds and calculates the amount of time from now that you want the object to be deleted. The proxy server that receives the request converts this header into an `X-Delete-At` header and calculates the deletion time using its current time plus the value given in seconds.

For existing objects that you want to assign expiration headers to, use the **POST** operation.

Example 3.54. Delete At Example

In the example, the `X-Delete-At` header is assigned with a Unix Epoch timestamp in integer form for Mon, 11 Jun 2012 15:38:25 GMT. Use http://www.epochconverter.com/ for example timestamps and a batch converter.

```
PUT /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
Content-Type: image/jpeg
  X-Delete-At: 1339429105
```

Example 3.55. Delete After Example

In this example, the `X-Delete-After` header is assigned a value in seconds, equivalent to 10 days. After this time, the object shall expire.

```
PUT /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
Content-Type: image/jpeg
X-Delete-After: 864000
```

3.4.7. Object Versioning

Object Versioning allows you to store multiple versions of your content to recover from unintended overwrites. It provides an easy method to implement version control which can be used on any type of content. It is strongly recommended that you put non-current objects in a container apart from where the current versions exist. Once you enable Object Versioning on a container (such as a "current version" container), PUTs to existing objects in that container copy the prior object to a separate "non-current version" container. Each of the non-current versions of an object has a time stamp appended to it, so you know when it was created.

To enable Object Versioning, your cloud provider has to set `allow_versions` to TRUE in their container config. Then, create a container where your non-current versions will be written. Next, set the metadata `X-Versions-Location` header on the container that holds the current versions of your objects. Set the metadata header to point to the new non-current version container you created. This is where your non-current versions will be stored. Once this is done, each object in your current-version container will have Object

Versioning enabled; changes to the objects automatically create non-current versions in the separate container.

Nothing is written to the non-current version container when you initially **PUT** an object into the current-version container. Only when you make edits to the objects via **PUT** will you create non-current versions. These non-current versions are labeled according to the schema below.

Naming Schema: Non-current versions are assigned the name <length><object_name>/<timestamp>, where length is the 3-character zero-padded hexadecimal character length of the <object_name> and <timestamp> is when the it was initially created as a current version.

Any return status in the 2xx range, such as 202 (Accepted), notes success. Status codes in the 4xx or 5xx range note failure. You should retry your request if you receive an error. Please note, however, that if you have specified a container that does not exist as your non-current version container, a status of 412 (Precondition Failed) returns when you edit the versioned object. If you receive this error, check that the container exists.

A **GET** to a versioned object returns the current version of the object without having to do any request redirects or metadata lookups.

A **POST** to a versioned object only updates the object's metadata; it does not create a new version of the object. In other words, new versions are only created when the content of the object changes.

A **DELETE** to a versioned object removes the current version of the object and replaces it with the next-most current version, moving it from the non-current container to the current. This next-most current version carries with it any metadata last set on it. If want to completely remove an object and you have five total versions of it, you must **DELETE** it five times.

Note

Note: A large-object manifest file cannot be versioned, but it may point to versioned segments.

To turn off Object Versioning on your current version container, remove its `X-Versions-Location` metadata by sending an empty key value.

Example 3.56. Object Versioning with cURL

Make sure a version-storing container exists, creating it if necessary (this example names it "versions"). Then create a container with the `X-Versions-Location` header. In this example, this container is named "current". You can also add the `X-Versions-Location` header to an existing container. In this example, the name of the container is “versions”; the location for the current version is the container "current".

Create a container named versions.

```
curl -i -XPUT -H "X-Auth-Token: <token>" http://<storage_url>/versions
```

Create a container named current with the `X-Versions-Location` header that references "versions".

```
curl -i -XPUT -H "X-Auth-Token: <token>" \
    -H "X-Versions-Location: versions" http://<storage_url>/current
```

Create an object (the first version):

```
curl -i -XPUT --data-binary 1 -H "X-Auth-Token: <token>" \
    http://<storage_url>/current/myobject
```

Now create a new version of that object:

```
curl -i -XPUT --data-binary 2 -H "X-Auth-Token: <token>" \
    http://<storage_url>/current/myobject
```

See a listing of the older versions of the object:

```
curl -i -H "X-Auth-Token: <token>" \
    http://<storage_url>/versions?prefix=008myobject/
```

Now delete the current version of the object and see that the older version is gone:

```
curl -i -XDELETE -H "X-Auth-Token: <token>" \
    http://<storage_url>/current/myobject
curl -i -H "X-Auth-Token: <token>" \
    http://<storage_url>/versions?prefix=008myobject/
```

3.4.8. Copy Object

Suppose you upload a file with the wrong object name or content type, or you needed to move some objects to another container. Without a server-side copy feature, you would need to repeat uploading the same content and then delete the existing object. With server-side object copy, you can save the step of re-uploading the content and thus also save the associated bandwidth charges, if any were to apply.

There are two ways to copy an existing object to another object in OpenStack Object Storage. One way is to do a **PUT** to the new object (the target) location, but add the "`X-Copy-From`" header to designate the source of the data. The header value should be the container and object name of the source object in the form of "/container/object". Also, the `X-Copy-From` **PUT** requests require a Content-Length header, even if it is zero (0).

```
PUT /<api version>/<account>/<container>/<destobject> HTTP/1.1
Host: <storage URL>
X-Auth-Token: <some-auth-token>
X-Copy-From: /<container>/<sourceobject>
Content-Length: 0
```

The second way to do an object copy is similar. Do a **COPY** to the existing object, and include the "Destination" header to specify the target of the copy. The header value is the container and new object name in the form of "/container/object".

```
COPY /<api version>/<account>/<container>/<sourceobject> HTTP/1.1
Host: <storage URL>
X-Auth-Token: <some-auth-token>
Destination: /<container>/<destobject>
```

With both of these methods, the destination container must exist before attempting the copy.

If you wanted to move the object rather than copy it, you need to send a **DELETE** request to the old object. A move is simply a **COPY** + **DELETE**. All metadata is preserved during the object copy. Note that you can set metadata on the request to copy the object (either the **PUT** or the **COPY**) and the metadata will overwrite any conflicting keys on the target (new) object. One interesting use case is to copy an object to itself and set the content type to a new value. This is the only way to change the content type of an existing object.

3.4.9. Delete Object

DELETE operations on an object are used to permanently remove an object from the storage system (metadata and data).

Deleting an object is processed immediately at the time of the request. Any subsequent **GET**, **HEAD**, **POST**, or **DELETE** operations will return a 404 (Not Found) error.

Objects with the `X-Delete-At` or `X-Delete-After` header assigned are deleted within one day of the expiration time and the object is not served immediately after the expiration time. Refer to Expiring Objects for more details.

Example 3.57. Object Delete Request

```
DELETE /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

No response body is returned. A status code of 204 (No Content) indicates success; status code 404 (Not Found) is returned when the object does not exist.

Example 3.58. Object Delete Response

```
HTTP/1.1 204 No Content
Date: Thu, 07 Jun 2010 20:59:39 GMT
Server: Apache
Content-Type: text/plain; charset=UTF-8
```

3.4.10. Retrieve Object Metadata

HEAD operations on an object are used to retrieve object metadata and other standard HTTP headers.

The only required header to be sent in the request is the authorization token.

Example 3.59. Object Metadata Request

```
HEAD /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

No response body is returned. Metadata is returned as HTTP headers. A status code of 200 (OK) indicates success; status 404 (Not Found) is returned when the object does not exist.

You may note that the **HEAD** return code for the object is different from that of the container. **HEAD** requests do not return a message body in the response, so anything in the 2xx response code range notes success. When a **HEAD** query is run against the container, it queries the container databases, and it does not retrieve the content of them, thus the 204 (No Content) return code. However, when a **HEAD** query is run against the object, it returns an "OK" response because it can view the content. In other words, the object **HEAD** query has a container length, but the container **HEAD** query has zero content length.

Example 3.60. Object Metadata Response

```
HTTP/1.1 200 OK
Date: Thu, 07 Jun 2010 20:59:39 GMT
Server: Apache
Last-Modified: Fri, 12 Jun 2010 13:40:18 GMT
ETag: 8a964ee2a5e88be344f36c22562a6486
Content-Length: 512000
Content-Type: text/plain; charset=UTF-8
X-Object-Meta-Meat: Bacon
X-Object-Meta-Fruit: Bacon
X-Object-Meta-Veggie: Bacon
X-Object-Meta-Dairy: Bacon
```

3.4.11. Update Object Metadata

POST operations against an object name are used to set and overwrite arbitrary key/value metadata or to assign headers not already assigned such as `X-Delete-At` or `X-Delete-After` for expiring objects. You cannot use the **POST** operation to change any of the object's other headers such as `Content-Type`, `ETag`, etc. It is not used to upload storage objects (see **PUT**). Also refer to copying an object when you need to update metadata or other headers such as Content-Type or CORS headers.

Key names must be prefixed with `X-Object-Meta-`. A **POST** request will delete all existing metadata added with a previous **PUT**/**POST**.

Example 3.61. Update Object Metadata Request

```
POST /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
X-Object-Meta-Fruit: Apple
X-Object-Meta-Veggie: Carrot
```

No response body is returned. A status code of 202 (Accepted) indicates success; status 404 (Not Found) is returned if the requested object does not exist.

Example 3.62. Update Object Metadata Response

```
HTTP/1.1 202 Accepted
Date: Thu, 07 Jun 2010 20:59:39 GMT
Server: Apache
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
```

4. Troubleshooting and Examples

This section introduces a command-line utility, cURL, and demonstrates interacting with the ReST interfaces through that utility.

4.1. Using cURL

cURL is a command-line tool which is available on most UNIX®-like environments and Mac OS X® and can be downloaded for Windows®. For more information on cURL, visit http://curl.haxx.se/.

cURL allows you to transmit and receive HTTP requests and responses from the command-line or from within a shell script. This makes it possible to work with the ReST API directly without using one of the client APIs.

The following cURL command-line options will be used

cURL Command-Line Options

`-X METHOD` Specify the HTTP method to request (**HEAD**, **GET**, etc.)

`-i` Dump HTTP response headers to stdout.

`-H HEADER` Specify an HTTP header in the request.

4.2. Authentication

In order to use the ReST API, you will first need to obtain a authorization token, which will need to be passed in for each request using the `X-Auth-Token` header. The following example demonstrates how to use cURL to obtain the authorization token and the URL of the storage system.

Example 4.1. cURL Authenticate

```
curl -i \
    -H "X-Auth-Key: jdoesecretpassword" \
    -H "X-Auth-User: jdoe" \
    https://auth.api.yourcloud.com/v1.0
```

```
HTTP/1.1 204 No Content
Date: Thu, 09 Jul 2009 15:31:39 GMT
Server: Apache/2.2.3
X-Storage-Url: https://storage.swiftdrive.com/v1/CF_xer7_343
X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae
Content-Length: 0
Connection: close
Content-Type: application/octet-stream
```

The storage URL and authentication token are returned in the headers of the response. After authentication, you can use cURL to perform **HEAD**, **GET**, **DELETE**, **POST** and **PUT** requests on the storage service.

4.3. Determining Storage Usage

A **HEAD** request can be sent to the storage service to determine how much data you have stored in the system and the number of containers you are using. Use the `-X` switch to specify the correct HTTP method and the `-i` to dump the HTTP response headers to terminal output (stdout).

Example 4.2. cURL Get Storage Space

```
curl –X HEAD -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343
```

```
HTTP/1.1 204 No Content
Date: Thu, 09 Jul 2009 15:38:14 GMT
Server: Apache
X-Account-Container-Count: 22
X-Account-Bytes-Used: 9891628380
Content-Type: text/plain
```

The HTTP request must include a header to specify the authentication token. The HTTP headers in the response indicate the number of containers in this storage account and the total bytes stored for the entire account.

4.4. Listing and Creating Containers

The simplest operation for Object Storage is to simply list the containers you have, which when you don't have any containers yet isn't terribly exciting:

Example 4.3. cURL List Storage Container

```
curl –X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343
```

```
HTTP/1.1 204 No Content
X-Account-Object-Count: 0
X-Account-Bytes-Used: 0
X-Account-Container-Count: 0
```

```
Accept-Ranges: bytes
X-Trans-Id: txe8ca5138ac8643ec84070543a0c9c91e
Content-Length: 0
Date: Mon, 07 Nov 2011 17:07:01 GMT
```

So, you take the X-Auth-Token obtained from the authentication operation, pass it as a header value, execute the operation against the URL obtained from the authentication operation, and force the GET verb with the -X switch. What you get back tells you there aren't any containers.

Next, let's create a container and then do the listing again:

Example 4.4. cURL Create Storage Container

```
curl –X PUT -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/george
```

```
HTTP/1.1 201 Created
Content-Length: 18
Content-Type: text/html; charset=UTF-8
X-Trans-Id: txb25576385284476d9fa6c73835f21650
Date: Mon, 07 Nov 2011 17:44:20 GMT

201 Created
```

Append the container name to the URL and force the PUT verb. That creates a container, which we can now see when we do a listing:

Example 4.5. cURL List Storage Container After a Creation

```
curl –X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343
```

```
HTTP/1.1 200 OK
X-Account-Object-Count: 0
X-Account-Bytes-Used: 0
X-Account-Container-Count: 1
Accept-Ranges: bytes
Content-Length: 7
Content-Type: text/plain; charset=utf-8
X-Trans-Id: txaedd6b080626453399c9f5febbddb73b
Date: Mon, 07 Nov 2011 17:44:23 GMT

george
```

You may have noticed the account metadata that comes back from the listing call. As you'd guess, it'll tell you how many objects you have, how much space you are using, and how many containers you are using.

4.5. Paging Lists of Containers

If you have a large number of containers, it is sometimes more convenient to page through them than getting some big long list of them. If I create more containers and then do a regular listing, here's what it looks like with five containers:

Example 4.6. cURL List Storage Container (long list)

```
curl –X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343
```

```
HTTP/1.1 200 OK
X-Account-Object-Count: 0
X-Account-Bytes-Used: 0
X-Account-Container-Count: 5
Accept-Ranges: bytes
Content-Length: 31
Content-Type: text/plain; charset=utf-8
X-Trans-Id: txb28795cc25b04f0dbce408dfa5a3cfc9
Date: Mon, 07 Nov 2011 19:03:06 GMT

cosmo
dogs
elaine
george
jerry
```

Suppose I want a page size of 2, all I do is append a ""?limit=2"" to my URL:

Example 4.7. cURL List Storage Container with Paging (first page)

```
curl –X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343?limit=2
```

```
HTTP/1.1 200 OK
X-Account-Object-Count: 0
X-Account-Bytes-Used: 0
X-Account-Container-Count: 5
Accept-Ranges: bytes
Content-Length: 11
```

```
Content-Type: text/plain; charset=utf-8
X-Trans-Id: tx940ee02c1a65451e96a2a2532e3a7ce7
Date: Mon, 07 Nov 2011 19:05:30 GMT

cosmo
dogs
```

Not surprisingly, I only get two containers. To get the next page, you tell the system which item you last saw with the "marker=" specifier:

Example 4.8. cURL List Storage Container with Paging (later pages)

```
curl –X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343?marker=dogs\&limit=2
```

```
HTTP/1.1 200 OK
X-Account-Object-Count: 0
X-Account-Bytes-Used: 0
X-Account-Container-Count: 5
Accept-Ranges: bytes
Content-Length: 14
Content-Type: text/plain; charset=utf-8
X-Trans-Id: tx2a69f7ec38c34078a185c5875a4c0e34
Date: Mon, 07 Nov 2011 19:15:00 GMT

elaine
george
```

Notice that I had to use \& so that my bash shell didn't try to interpret the & as wanting to run something in its own thread. With that in place, you get the next page of items that appear after the marker.

4.6. Serialized Output

In other situations, like if you are working on a language binding on top of the REST API, you might want more structured data back from the method calls. By appending a "format=" and then choosing either json or xml, you can get that structured data back you've been dreaming about.

Example 4.9. cURL List Storage Container (JSON output)

```
curl –X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343?format=json
```

```
HTTP/1.1 200 OK
X-Account-Object-Count: 0
X-Account-Bytes-Used: 0
X-Account-Container-Count: 5
Accept-Ranges: bytes
Content-Length: 187
Content-Type: application/json; charset=utf-8
X-Trans-Id: txd408573a51d2423c848cba191fbede9b
Date: Mon, 07 Nov 2011 19:17:33 GMT

[{"name":"cosmo", "count":0,"bytes":0},
{"name":"dogs","count":0,"bytes":0},
{"name":"elaine","count":0,"bytes":0},
{"name":"george","count":0,"bytes":0},
{"name":"jerry","count":0,"bytes":0}]
```

Example 4.10. cURL List Storage Container (XML output)

```
curl –X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343?format=xml
```

```
HTTP/1.1 200 OK
X-Account-Object-Count: 0
X-Account-Bytes-Used: 0
X-Account-Container-Count: 5
Accept-Ranges: bytes
Content-Length: 479
Content-Type: application/xml; charset=utf-8
X-Trans-Id: tx5e5685a15d0b406799b6a425b1150e4c
Date: Mon, 07 Nov 2011 19:17:38 GMT

<?xml version="1.0" encoding="UTF-8"?>
<account name="AUTH_a23f73d2-abfb-4656-af94-32ddec35dab8">
<container><name>cosmo</name><count>0</count><bytes>0</bytes></container>
<container><name>dogs</name><count>0</count><bytes>0</bytes></container>
<container><name>elaine</name><count>0</count><bytes>0</bytes></container>
<container><name>george</name><count>0</count><bytes>0</bytes></container>
<container><name>jerry</name><count>0</count><bytes>0</bytes></container>
</account>
```

The remainder of the examples in this document will use the standard, non-serialized output but all operations accept the format argument. You might notice that when you use one of the formats, you get more information about the containers. That's the per-container metadata, which is covered in the next section.

4.7. Container Metadata and Deleting Containers

You can get at container metadata directly simply by appending the name of the container to a HEAD request:

Example 4.11. cURL List Container Metadata

```
curl –X HEAD -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/dogs
```

```
HTTP/1.1 204 No Content
X-Container-Object-Count: 0
X-Container-Bytes-Used: 0
Accept-Ranges: bytes
X-Trans-Id: tx3dd984f9482341dd97546e9d49d65e90
Content-Length: 0
Date: Mon, 07 Nov 2011 20:39:41 GMT
```

Not very exciting without any objects in the container, but you get the idea. While you cannot update or delete container metadata, you can delete a container:

Example 4.12. cURL Delete Storage Container

```
curl –X DELETE -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/george
```

```
HTTP/1.1 204 No Content
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx3fa3857f266f44319d9b8f4bf7ce7fc8
Date: Mon, 07 Nov 2011 20:42:58 GMT
```

Then let's confirm the delete by listing the containers again:

Example 4.13. cURL List Containers After a Delete

```
curl –X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343
```

```
HTTP/1.1 200 OK
X-Account-Object-Count: 0
X-Account-Bytes-Used: 0
X-Account-Container-Count: 4
Accept-Ranges: bytes
Content-Length: 24
```

```
Content-Type: text/plain; charset=utf-8
X-Trans-Id: tx2475741852b849ce9403e382fe3f8015
Date: Mon, 07 Nov 2011 20:43:08 GMT

cosmo
dogs
elaine
jerry
```

4.8. Special Metadata: Container ACLs

A particularly important metadata element for containers is X-Container-Read, which establishes the ACL permissions on who can read objects in the container. Prior to being set, the ACL logic default to only be accessible to someone with a valid X-Auth-Token for the account in question. Doing a simple listing of a container shows us the absence of X-Container-Read in this default situation:

Example 4.14. cURL List Container Showing Lack of ACL

```
curl –X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/jerry
```

```
HTTP/1.1 204 No Content
X-Container-Object-Count: 0
X-Container-Bytes-Used: 0
Accept-Ranges: bytes
X-Trans-Id: tx3aa52e951fc64b63bc1fda27902b9bd3
Content-Length: 0
Date: Tue, 15 Nov 2011 03:29:22 GMT
```

Now we'll set the X-Container-Read. For a full explanation of valid values, see: http://swift.openstack.org/misc.html#acls but for our simple needs, we'll enable read access and listing access to anybody:

Example 4.15. cURL Setting an ACL on a Container

```
curl –X PUT -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    -H "X-Container-Read: .r:*,.rlistings" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/jerry
```

```
HTTP/1.1 202 Accepted
Content-Length: 58
Content-Type: text/html; charset=UTF-8
```

```
X-Trans-Id: txf2befb56b1854a50995f710f2db48089
Date: Tue, 15 Nov 2011 03:33:16 GMT

202 Accepted

The request is accepted for processing.
```

To see the metadata change, do a listing again:

Example 4.16. cURL List Container Showing with an ACL

```
curl -X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/jerry
```

```
HTTP/1.1 204 No Content
X-Container-Object-Count: 0
X-Container-Read: .r:*,.rlistings
X-Container-Bytes-Used: 0
Accept-Ranges: bytes
X-Trans-Id: txb40eb86d949345f7bc66b01e8b63c3a5
Content-Length: 0
Date: Tue, 15 Nov 2011 03:33:36 GMT
```

The side effect of giving anybody read access is that any object in the container is now accessible from a browser simply by entering the X-Storage-URL used throughout the session and append the object name. For example:

https://storage.swiftdrive.com/v1/CF_xer7_343/jerry/cereal.jpg

would be the URL of an object named "cereal.jpg" in the container "jerry" that has been made publicly accessible using this method.

4.9. Creating Objects

Enough with containers already, let's start to upload some objects. Suppose you had a local directory full of dog pictures:

Example 4.17. Sample File Listing

```
$ ls -l
total 504
-rw-r--r--@ 1 petecj2  staff   44765 Nov  7 14:49 JingleRocky.jpg
-rw-r--r--@ 1 petecj2  staff  100864 Nov  7 14:47 RockyAndBuster.jpg
-rw-r--r--@ 1 petecj2  staff  107103 Nov  7 14:47 SittingBuster.jpg
```

In order to put one of them in a container called "dogs" with cURL, you'd do this:

Example 4.18. Creating and Uploading an Object to a Container

```
curl –X PUT -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    -T JingleRocky.jpg" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/dogs/JingleRocky.jpg
```

```
HTTP/1.1 201 Created
Content-Length: 118
Content-Type: text/html; charset=UTF-8
Etag: f7d40eceffdd9c2ecab226105737b2a6
Last-Modified: Mon, 07 Nov 2011 22:51:29 GMT
X-Trans-Id: txd131cc897c78403daf5fad010d4d7152
Date: Mon, 07 Nov 2011 22:51:30 GMT

<html>
 <head>
  <title>201 Created</title>
 </head>
 <body>
  <h1>201 Created</h1>
  <br /><br />



 </body>
</html>
```

The object gets named from whatever we append to the URL path beyond the container name and the -T switch lets us name a file to push with the operation as the request body. We can confirm the upload by checking the container again:

Example 4.19. cURL List Container Showing Newly Uploaded Object

```
curl –X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/dogs
```

```
HTTP/1.1 200 OK
X-Container-Object-Count: 1
X-Container-Read: .r:*,.rlistings
X-Container-Bytes-Used: 44765
Accept-Ranges: bytes
Content-Length: 16
Content-Type: text/plain; charset=utf-8
```

```
X-Trans-Id: tx83be89d4e1a34eacbfeebcdfc7a7f2e7
Date: Mon, 07 Nov 2011 22:56:25 GMT

JingleRocky.jpg
```

Notice that the container metadata now reflects the number of objects and the bytes match what we saw when we did the directory listing. After uploading the other two similarly, we get a full object listing:

Example 4.20. cURL List Container Showing Multiple Newly Uploaded Objects

```
curl -X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/dogs
```

```
HTTP/1.1 200 OK
X-Container-Object-Count: 3
X-Container-Read: .r:*,.rlistings
X-Container-Bytes-Used: 252732
Accept-Ranges: bytes
Content-Length: 53
Content-Type: text/plain; charset=utf-8
X-Trans-Id: txae17dfa78da64117aaf07585a1b02115
Date: Mon, 07 Nov 2011 23:00:56 GMT

JingleRocky.jpg
RockyAndBuster.jpg
SittingBuster.jpg
```

4.10. Paging Lists of Objects

Exactly like listing containers, objects can be listed in pages at a time using markers to denote pages. From the previous example with 3 objects in the container "dogs", the list can be paged with the "limit" query string variable:

Example 4.21. cURL List Objects (first page)

```
curl -X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/dogs?limit=2
```

```
HTTP/1.1 200 OK
X-Container-Object-Count: 3
X-Container-Read: .r:*,.rlistings
X-Container-Bytes-Used: 252732
```

```
Accept-Ranges: bytes
Content-Length: 35
Content-Type: text/plain; charset=utf-8
X-Trans-Id: tx5e00fa9fa895423198bc814cb0c6162d
Date: Tue, 15 Nov 2011 03:53:51 GMT

JingleRocky.jpg
RockyAndBuster.jpg
```

And the second page fetched with:

Example 4.22. cURL List Objects with Paging (later pages)

```
curl –X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/dogs?marker=RockyAndBuster.
jpg\&limit=2
```

```
HTTP/1.1 200 OK
X-Container-Object-Count: 3
X-Container-Read: .r:*,.rlistings
X-Container-Bytes-Used: 252732
Accept-Ranges: bytes
Content-Length: 18
Content-Type: text/plain; charset=utf-8
X-Trans-Id: txe1287a7179dc4dfd98610850a0fff157
Date: Tue, 15 Nov 2011 03:54:21 GMT

SittingBuster.jpg
```

4.11. Retrieve, Copy, and Delete Objects

Now we'll retrieve an object previously uploaded. First, we'll remove the local copy:

Example 4.23. Removing Local Copies

```
$ ls -l
total 504
-rw-r--r--@ 1 petecj2  staff   44765 Nov  7 14:49 JingleRocky.jpg
-rw-r--r--@ 1 petecj2  staff  100864 Nov  7 14:47 RockyAndBuster.jpg
-rw-r--r--@ 1 petecj2  staff  107103 Nov  7 14:47 SittingBuster.jpg
$ rm JingleRocky.jpg
$ ls -l
total 416
-rw-r--r--@ 1 petecj2  staff  100864 Nov  7 14:47 RockyAndBuster.jpg
-rw-r--r--@ 1 petecj2  staff  107103 Nov  7 14:47 SittingBuster.jpg
```

Be sure not to use -i switch here since what we want is the raw data, which we'll then pipe to a file:

Example 4.24. cURL Retrieve an Object

```
curl –X GET \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/dogs/JingleRocky.jpg >
 JingleRocky.jpg
```

```
$ ls -l
total 504
-rw-r--r--  1 petecj2  staff   44765 Nov  7 15:11 JingleRocky.jpg
-rw-r--r--@ 1 petecj2  staff  100864 Nov  7 14:47 RockyAndBuster.jpg
-rw-r--r--@ 1 petecj2  staff  107103 Nov  7 14:47 SittingBuster.jpg
```

Next, Object Storage provides a facility to copy objects from one container to another entirely on the server side. To do this, you do a PUT with the destination container and new object name while passing a special X-Copy-From header and a Content-Length of zero:

Example 4.25. cURL Server-side Copy an Object

```
curl –X PUT -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    -H "X-Copy-From: /dogs/JingleRocky.jpg" \
    -H "Content-Length: 0" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/elaine/JingleRocky.jpg
```

```
HTTP/1.1 201 Created
Content-Length: 118
Content-Type: text/html; charset=UTF-8
Etag: f7d40eceffdd9c2ecab226105737b2a6
X-Copied-From: dogs/JingleRocky.jpg
Last-Modified: Mon, 07 Nov 2011 23:23:53 GMT
X-Trans-Id: tx244cd14df1b94d8c91ec5dcf8c5f9da4
Date: Mon, 07 Nov 2011 23:23:54 GMT

<html>
 <head>
  <title>201 Created</title>
 </head>
 <body>
  <h1>201 Created</h1>
  <br /><br />



 </body>
</html>
```

You can then confirm the new location of the object. To do this, you do a GET with the destination container to see the listing of the object:

Example 4.26. cURL Confirming the Server-side Copy an Object

```
curl –X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/elaine/
```

```
HTTP/1.1 200 OK
X-Container-Object-Count: 1
X-Container-Bytes-Used: 44765
Accept-Ranges: bytes
Content-Length: 16
Content-Type: text/plain; charset=utf-8
X-Trans-Id: tx46986b4a09b34790924fd43842b2b0dd
Date: Mon, 07 Nov 2011 23:24:05 GMT

JingleRocky.jpg
```

To delete an object from its container, simply use the DELETE verb:

Example 4.27. cURL Delete an Object

```
curl –X DELETE -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/elaine/JingleRocky.jpg
```

```
HTTP/1.1 204 No Content
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: txd45f04422b034e6f8447de400b78cbf3
Date: Mon, 07 Nov 2011 23:32:39 GMT
```

Confirming the deletion by doing a container listing:

Example 4.28. cURL Confirming the Delete an Object

```
curl –X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/elaine/
```

```
HTTP/1.1 204 No Content
X-Container-Object-Count: 0
X-Container-Bytes-Used: 0
Accept-Ranges: bytes
X-Trans-Id: txc9b43bf4d896405eb9a88ca468bf7b2d
Content-Length: 0
Date: Mon, 07 Nov 2011 23:32:41 GMT
```

4.12. Object Metadata

Objects can have whatever metadata keys/values you choose. Simply POST an HTTP Header to the object in the form of X-Object-Meta-<key>: <value>. Like this:

Example 4.29. cURL Set Object Metadata

```
curl -X POST -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    -H "X-Object-Meta-Breed: Terrier pit bull mix"" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/dogs/JingleRocky.jpg
```

```
<html>
 <head>
  <title>202 Accepted</title>
 </head>
 <body>
  <h1>202 Accepted</h1>
  The request is accepted for processing.<br /><br />


 </body>
</html>
```

And then read the object metadata with a HEAD on the object path:

Example 4.30. cURL Reading Object Metadata

```
curl -X HEAD -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/dogs/JingleRocky.jpg
```

```
   HTTP/1.1 200 OK
X-Object-Meta-Breed: Terrier pit bull mix
Last-Modified: Tue, 08 Nov 2011 01:26:49 GMT
Etag: f7d40eceffdd9c2ecab226105737b2a6
```

```
Accept-Ranges: bytes
Content-Length: 44765
Content-Type: image/jpeg
X-Trans-Id: txa8bff9ad7ef844829103c1f9b8c20781
Date: Tue, 08 Nov 2011 01:29:35 GMT
```

4.13. Pseudo-Hierarchical Folders/Directories

For the last section, we come to the most confusing concept in Object Storage. In most storage systems, you have the ability to create custom hierarchies of files so that you can better organize them. On its surface, Object Storage only gives you one level of hierarchy in the form of containers. However, it turns out that you can get creative with naming your objects to give yourself the same effect as having hierarchical containers.

Let's start with a fresh container without any objects in it:

Example 4.31. cURL Create New Container for Folders

```
curl -X PUT -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/photos
```

```
HTTP/1.1 201 Created
Content-Length: 18
Content-Type: text/html; charset=UTF-8
X-Trans-Id: txc78254a41b374b6ea10590d90874f769
Date: Wed, 16 Nov 2011 00:06:22 GMT

201 Created
```

Now list the new container:

Example 4.32. cURL Listing the New Container

```
curl -X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/photos
```

```
HTTP/1.1 204 No Content
X-Container-Object-Count: 0
X-Container-Bytes-Used: 0
Accept-Ranges: bytes
X-Trans-Id: tx49112200f7934c2bab1de3ae103c368e
Content-Length: 0
Date: Wed, 16 Nov 2011 00:06:26 GMT
```

Next, add an object but prefix the name with the hierarchy desired:

Example 4.33. cURL Upload an Object with a Prefix

```
curl -X PUT -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    -T JingleRocky.jpg \
    https://storage.swiftdrive.com/v1/CF_xer7_343/photos/terriers/JingleRocky.
jpg
```

```
HTTP/1.1 201 Created
Content-Length: 118
Content-Type: text/html; charset=UTF-8
Etag: f7d40eceffdd9c2ecab226105737b2a6
Last-Modified: Wed, 16 Nov 2011 00:09:18 GMT
X-Trans-Id: txe34fdf2704f044e3a7102256386b1cb7
Date: Wed, 16 Nov 2011 00:09:19 GMT

<html>
 <head>
   <title>201 Created</title>
 </head>
 <body>
   <h1>201 Created</h1>
   <br /><br />



 </body>
</html>
```

Do it again with a different object and prefix:

Example 4.34. cURL Upload a Different Object with a Different Prefix

```
curl -X PUT -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    -T SittingBuster.jpg \
    https://storage.swiftdrive.com/v1/CF_xer7_343/photos/chihuahuas/
SittingBuster.jpg
```

```
HTTP/1.1 201 Created
Content-Length: 118
Content-Type: text/html; charset=UTF-8
Etag: e692e744c7180ee368166a24f1a2fa9b
Last-Modified: Wed, 16 Nov 2011 00:52:25 GMT
X-Trans-Id: txe229d03af5ea4d2ea1071def213c3f02
Date: Wed, 16 Nov 2011 00:52:25 GMT
```

```
<html>
 <head>
   <title>201 Created</title>
 </head>
 <body>
   <h1>201 Created</h1>
   <br /><br />



 </body>
</html>
```

Now list the container, revealing the prefixes:

Example 4.35. cURL Listing a Container with Object Prefix

```
curl –X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/photos
```

```
HTTP/1.1 200 OK
X-Container-Object-Count: 2
X-Container-Bytes-Used: 151868
Accept-Ranges: bytes
Content-Length: 54
Content-Type: text/plain; charset=utf-8
X-Trans-Id: tx8544a17e8b1e4da693145fb5f2e6db43
Date: Wed, 16 Nov 2011 00:53:43 GMT

chihuahuas/SittingBuster.jpg
terriers/JingleRocky.jpg
```

If you want to perform hierarchical listings, with the prefixes in place, you can use the "path" query string variable:

Example 4.36. cURL Listing a Container with a Path

```
curl –X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/photos?path=terriers
```

```
HTTP/1.1 200 OK
X-Container-Object-Count: 2
X-Container-Bytes-Used: 151868
Accept-Ranges: bytes
Content-Length: 25
```

```
Content-Type: text/plain; charset=utf-8
X-Trans-Id: tx3f1b9575d4de4a7d97ba3f9ad81923cc
Date: Wed, 16 Nov 2011 00:55:12 GMT

terriers/JingleRocky.jpg
```

If you wanted to see what prefixes were in place, you can use the "delimiter" query string variable to distinguish prefix paths from object names:

Example 4.37. cURL Listing a Container with a Delimiter

```
curl –X GET -i \
    -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
    https://storage.swiftdrive.com/v1/CF_xer7_343/photos?delimiter=/
```

```
HTTP/1.1 200 OK
X-Container-Object-Count: 2
X-Container-Bytes-Used: 151868
Accept-Ranges: bytes
Content-Length: 22
Content-Type: text/plain; charset=utf-8
X-Trans-Id: tx7222a3dd73fe44b888db4e58cc647d1e
Date: Wed, 16 Nov 2011 00:57:40 GMT

chihuahuas/
terriers/
```

Using these in combination allows you to discover directories within a particular path and then further drill down based on the results.