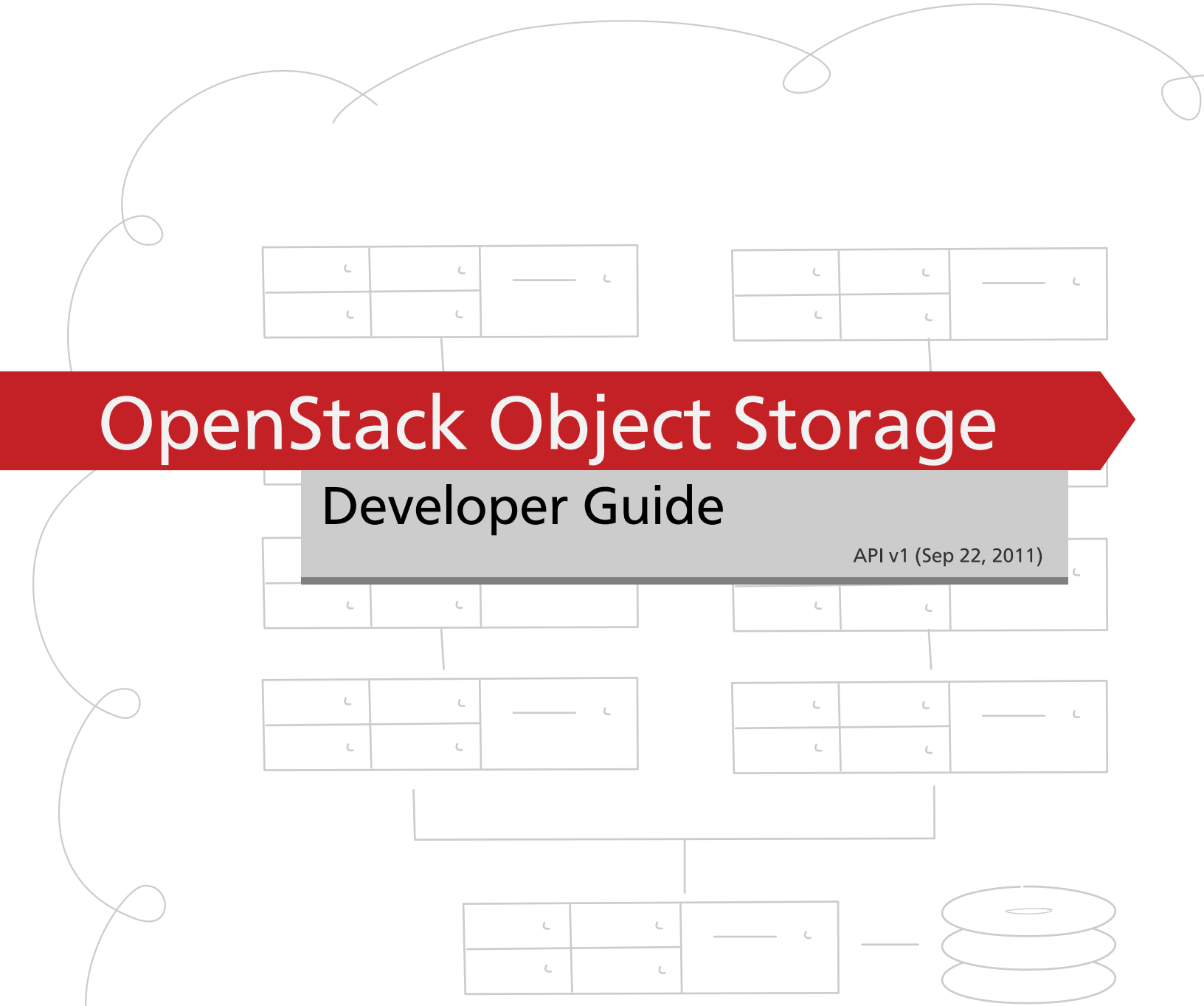

OpenStack Object Storage

Developer Guide

API v1 (Sep 22, 2011)

openstack™

docs.openstack.org

OpenStack Object Storage Developer Guide

API v1 (2011-09-22)

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 storage support you can download and .

OpenStack Object Storage allows users to store and retrieve files and content via a simple Web Service (ReST: Representational State Transfer) interface. 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 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. The most recent changes for both guides are described in the table below:

Revision Date	Summary of Changes
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 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.

Revision Date	Summary of Changes
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.

For more details about the Rackspace Cloud Files implementation of the OpenStack Object Storage service, please refer to 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, or swauth, an auth service for Swift as WSGI middleware that uses Swift itself as a backing store that is

provided via download from Github, or 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 for more information about ReST.

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 URIs, 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 pointers 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

3.1. Storage Account Services

The following operations can be performed at the account level of the URI. For example, the URI 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 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 at most *n* values.

`marker` Given a string value *x*, return object names greater 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. List Large Number of Containers

The system will return a maximum of 10,000 container names per request. To retrieve subsequent container names, another request must be made with a 'marker' parameter. The marker indicates where the last list left off; the system will return 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, a 'limit' parameter may be given.

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

Example 3.8. 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.

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
```

3.2. 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 3.11. 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.
`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.
`path`	For a string value *x*, return the object names nested in the pseudo path (assuming preconditions are met - see below).
`delimiter`	For a character *c*, return all the object names nested in the container (without the need for the directory marker objects).

Example 3.12. 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 or does not exist

for the specified account. If an incorrect account is specified, the HTTP return code will be 404 (Not Found).

Example 3.13. 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 3.14. 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 3.15. 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 3.16. Objects Details Request: XML

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

Example 3.17. 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. List Large Number of Objects

The system will return a maximum of 10,000 object names per request. To retrieve subsequent object names, another request must be made with a 'marker' parameter. The marker indicates where the last list left off and the system will return 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 3.18. 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=oranges
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

```
reddelicious
```

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

3.2.1.3. Pseudo-Hierarchical Folders/Directories

You can simulate a hierarchical structure in OpenStack Object Storage by following a few guidelines. Object names must contain the forward slash character / as a path element

separator and also create *directory marker* objects; then they will be able to traverse this nested structure with the new *path* query parameter. This can best be illustrated by example:

Note

For the purposes of this example, the container where the objects reside is called `backups`. All objects in this example start with a prefix of `photos` and should **NOT** be confused with the container name. In the example, the full URI of the `me.jpg` file would be `https://storage.swiftdrive.com/v1/CF_xer7_343/backups/photos/me.jpg`

Example 3.19. Pseudo-Hierarchical Folders/Directories

In the example, the following *real* objects are uploaded to the storage system with names representing their full filesystem path:

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

To take advantage of this feature, the *directory marker* objects must also be created to represent the appropriate directories. The following additional objects need to be created. A good convention would be to create these as zero- or one-byte files with a Content-Type of `application/directory`.

```
photos/animals/dogs
photos/animals/cats
photos/animals
photos/plants
photos
```

Now issuing a **GET** request against the container name coupled with the `path` query parameter of the directory to list can traverse these *directories*. Only the request line and results are depicted below excluding other request/response headers.

```
GET /v1/AccountString/backups?path=photos HTTP/1.1
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

```
photos/animals
photos/cats
photos/me.jpg
```

To traverse down into the `animals` directory, specify that path.

```
GET /v1/AccountString/backups?path=photos/animals
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

```
photos/animals/dogs
photos/animals/cats
```

By combining this `path` query parameter with the `format` query parameter, users will be able to easily distinguish between virtual folders/directories by Content-Type and build interfaces that allow traversal of the pseudo-nested structure.

You can also use a delimiter parameter to represent a nested directory hierarchy without the need for the directory marker objects. You can use any single character as a delimiter. The listings can return virtual directories - they are virtual in that they don't actually represent real objects. like the directory markers, though, they will have a content-type of application/directory and be in a subdir section of json and xml results.

If you have the following objects—photos/photo1, photos/photo2, movieobject, videos/movieobj4—in a container, your delimiter parameter query using slash (/) would give you photos, movieobject, videos.

```
GET /v1/acct/container?delimiter=/
Host: storage.swiftdrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

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 3.20. 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 3.21. 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 3.22. 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 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 3.23. 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 are used to permanently remove that container. 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 3.24. Container Delete Request

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

'Response '

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 will be generated.

Example 3.25. 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 3.26. 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 3.27. 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
```

3.3. 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.

3.3.1. 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 3.28. 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 3.29. 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
```

```
[ ... ]
```

3.3.2. Create/Update Object

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

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.

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 3.30. 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 412 (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 3.31. 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
```

3.3.2.1. 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 3.32. 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 412 (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 3.33. 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 3.34. 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.

3.3.2.2. 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 that 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 3.35. 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
```

3.3.3. 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 3.36. 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
```

3.3.4. 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 3.37. 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
```

3.3.5. 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 3.38. 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
```

3.3.6. 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 were wanting to perform a move of the objects rather than a copy, you would need to send a DELETE request to the old object. A move simply becomes 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.

3.3.7. Delete Object

DELETE operations on an object are used to permanently remove that 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.

Example 3.39. 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 404 (Not Found) is returned when the object does not exist.

Example 3.40. 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
```

3.3.8. 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 3.41. 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.

Example 3.42. 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
```

3.3.9. Update Object Metadata

POST operations against an object name are used to set and overwrite arbitrary key/value metadata. 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**).

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

Example 3.43. 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 when the requested object does not exist.

Example 3.44. 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

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.)

`-D` Dump HTTP response headers to stdout.

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

4.1.1. 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 -D - \
     -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.

4.1.2. 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 `-D` to dump the HTTP response headers to terminal output (stdout).

Example 4.2. cURL Get Storage Space

```
curl –X HEAD -D - \
     -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.

4.1.3. Creating a Storage Container

Before uploading any data to OpenStack Object Storage, you must create a storage container. You do this with a **PUT** request; cURL can be used for that, too.

Example 4.3. cURL Create Storage Container

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

```
HTTP/1.1 201 Created
Date: Thu, 09 Jul 2009 17:03:36 GMT
Server: Apache
Content-Length: 0
Content-Type: text/plain
```

Returning an HTTP status code of 201 (Created) indicates that the container was successfully created.

4.1.4. Uploading a Storage Object

After creating a container, you can upload a local file. For this example, let's upload a screenshot image. The `-T` switch specifies the full path to the local file to upload.

Example 4.4. cURL Upload Storage Object

```
curl –X PUT -T screenies/wow1.jpg-D - \
  -H "ETag: 805120ec285a7ed28f74024422fe3594" \
  -H "Content-Type: image/jpeg" \
  -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
  -H "X-Object-Meta-Screenie: Mel visits Outland" \
  https://storage.swiftdrive.com/v1/CF_xer7_343/images/wow1.jpg
```

```
HTTP/1.1 201 Created
Date: Thu, 09 Jul 2009 17:03:36 GMT
Server: Apache
Content-Length: 0
Etag: 805120ec285a7ed28f74024422fe3594
Content-Type: text/plain
```

4.1.5. Other cURL Commands

You can issue any of the ReST methods defined for OpenStack Object Storage with the cURL utility. For example, you can use cURL to send **POST** and **DELETE** requests even though we haven't provided specific examples.

Note that generally each time you invoke `curl` to perform an operation, the system creates a separate TCP/IP and SSL connection and then throws it away. The language APIs, however, are designed to re-use these connections between operations and therefore provide much better performance. We recommend that you use one of the supported language APIs in your production applications and limit `curl` to quick-and-easy testing/troubleshooting.