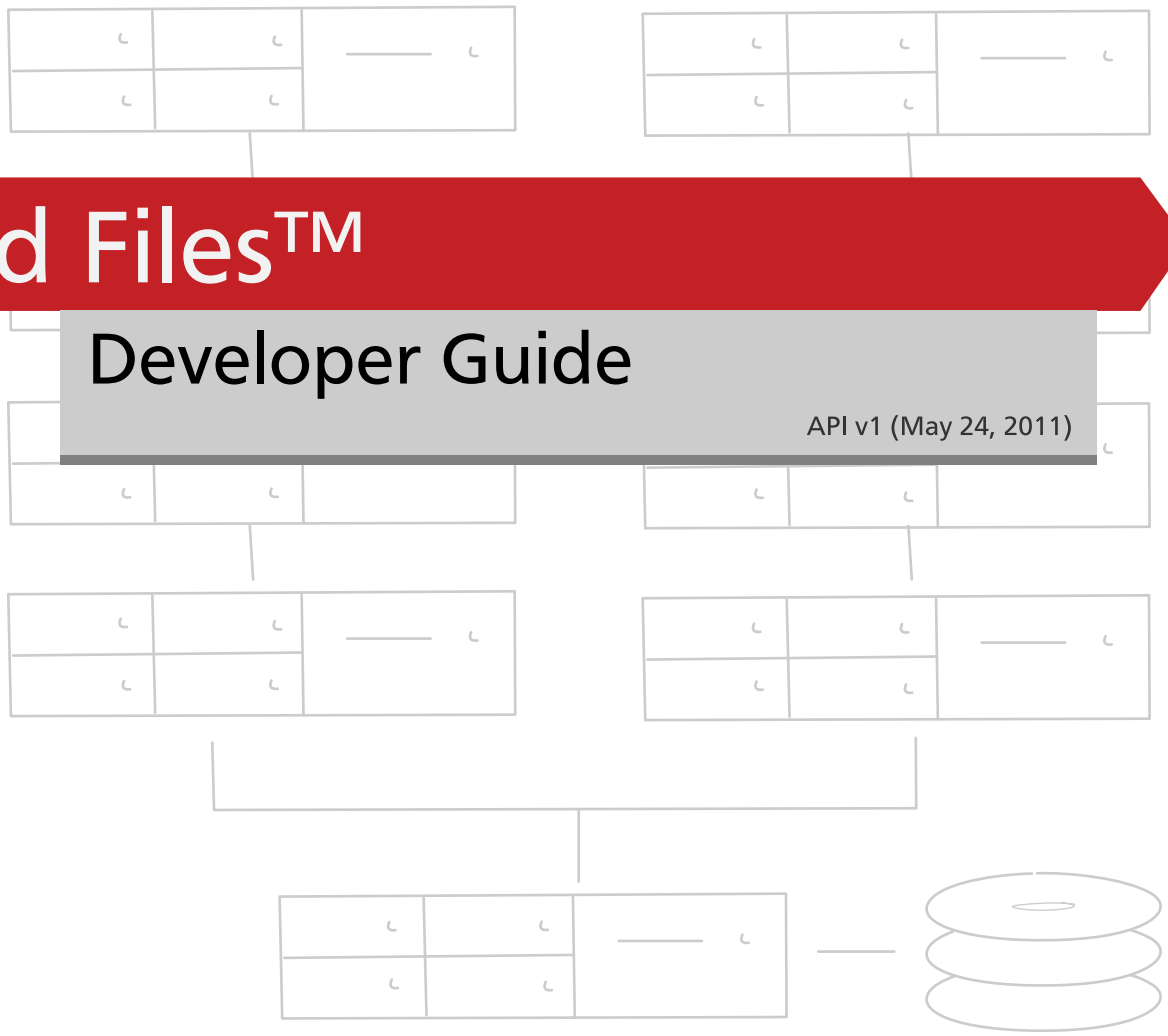
Cloud Files™

Developer Guide

API v1 (May 24, 2011)



rackspace® HOSTING

docs.rackspacecloud.com/api

Cloud Files™ Developer Guide

API v1 (2011-05-24)

This document is intended for software developers interested in developing applications using the Rackspace Cloud Files™ Application Programming Interface (API). The document is for informational purposes only and is provided "AS IS."

Table of Contents

List of Figures

List of Examples

1. Overview

Rackspace Cloud Files™ 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. Users can store as much as they want and pay only for storage space they actually use.

Additionally, Cloud Files provides a simple yet powerful way to publish and distribute content behind a Content Distribution Network. Cloud Files users get access to this network automatically without having to worry about contracts, additional costs, or technical hurdles.

Cloud Files allows users to store and retrieve files and CDN-enabled 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 Cloud Files service, please refer to http://www.rackspacecloud.com/cloud_hosting_products/files

We welcome feedback, comments, and bug reports at support@rackspacecloud.com.

1.1. Intended Audience

This guide is intended to assist software developers who want to develop applications using the Rackspace Cloud Files API. It fully documents the ReST application programming interface (API) that allows developers to interact with the storage and CDN components of the Cloud Files system. To use the information provided here, you should first have a general understanding of the Rackspace Cloud Files service and have access to an active Rackspace Cloud Files account. 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 ares C#/.NET, Java, PHP, Python, and Ruby. These APIs utilize the ReST API and are provided to help developers rapidly integrate Cloud Files 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.

System administrators and other users who are interested in the storage and CDN benefits of Cloud Files should consider using the File Manager interface within the Rackspace Cloud Control Panel, Jungle Disk, or third party tools such as Fileuploader, Cyberduck, or Cloud Files Manager. The control panel provides an easy to use web-based interface for uploading and downloading content to and from Cloud Files.

1.2. Document Change History

This version of the Developer Guide replaces and obsoletes all previous versions. The most recent changes are described in the table below:

Revision Date	Summary of Changes
May 24, 2011	• Added information about new headers including CORS headers.
Apr. 20, 2011	• HEAD returns 200 instead of 204 on an object metadata request. • TTL maximum value is now 50 years instead of 3 days, the minimum TTL is now 15 minutes (900 seconds), and the default is now 72 hours instead of 24 hours.
Mar. 25, 2011	• Added information about large object support.
Mar. 17, 2011	• Added information about container metadata.
Mar. 10, 2011	• Added a section about retrieving an SSL URI for CDN-enabled containers that are using https protocol. • Updated examples to contain SSL as appropriate.
Feb. 25, 2011	• Added information about the edge purge capability for CDN-enabled containers and objects.
Feb. 18, 2011	• Fixed error in the header range example that stated first instead of last when fetching a portion of the data. • Updated CDN URLs to match new format. • Fixed error referring to X-Auth-User instead of X-Auth-Key.
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 Rackspace Cloud website at http://docs.rackspacecloud.com/files/api/cf-devguide-latest.pdf.

For more details about the Cloud Files 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.

You can also follow updates and announcements via twitter at http://www.twitter.com/rackcloud

2. Concepts

Cloud Files is not a file system in the traditional sense. You will not be able to map or mount virtual disk drives like you can with other forms of storage such as a SAN or NAS. Since Cloud Files is a different way of thinking when it comes to storage, you should take a few moments to review the key concepts listed below.

2.1. Accounts

The Cloud Files system is designed to be used by many different customers. Your user account is your portion of the Cloud Files system. A user must identify themselves with their Rackspace Cloud username and API Access Key and once authenticated, that user has full read/write access to the files stored under that user account. Please visit http://www.rackspacecloud.com/signup to obtain a Cloud Files account and enable your API Access Key.

2.2. Authentication

The language and ReST APIs below describe how to authenticate against the Authentication service to receive Cloud Files connection parameters and an authentication token. The token must be passed in for all subsequent container/object operations.

Note

The language-specific APIs handle authentication, token passing, and HTTPS request/response communication.

2.3. Permissions

In Cloud Files, each user has their own storage account and has full access to that account. Users must authenticate with their credentials as described above, but once authenticated they can create/delete containers and objects within that account.

2.4. Containers

A container is a storage compartment for your data and provides a way for you to organize your data. You can think of a container as a folder in Windows® or a directory in UNIX®. The primary difference between a container and these other file system concepts is that containers cannot be nested. You can, however, create an unlimited number of containers within your account. Data must be stored in a container so you must have at least one container defined in your account prior to uploading data.

The only restrictions on container names is that they cannot contain a forward slash (`/`) and must be less than 256 bytes in length. Please note that the length restriction applies to the name after it has been URL encoded. For example, a container name of `Course Docs` would be URL encoded as `Course%20Docs` and therefore be 13 bytes in length rather than the expected 11.

2.5. Objects

An object is the basic storage entity and any optional metadata that represents the files you store in the Cloud Files system. When you upload data to Cloud Files, the data is stored as-is (no compression or encryption) and consists of a location (container), the object's name, and any metadata consisting of key/value pairs. For instance, you may chose to store a backup of your digital photos and organize them into albums. In this case, each object could be tagged with metadata such as `Album : Caribbean Cruise` or `Album : Aspen Ski Trip`.

The only restriction on object names is that they must be less than 1024 bytes in length after URL encoding. For example, an object name of `C++final(v2).txt` should be URL encoded as `C%2B%2Bfinal%28v2%29.txt` and therefore be 24 bytes in length rather than the expected 16.

Cloud Files has a limit on the size of a single uploaded object; by default this is 5 GB. However, the download size of a single object is virtually unlimited with the concept of segmentation. Segments of the larger object are uploaded and a special manifest file is created that, when downloaded, sends all the segments concatenated as a single object. This also offers much greater upload speed with the possibility of parallel uploads of the segments.

For metadata, you should not exceed 90 individual key/value pairs for any one object and the total byte length of all key/value pairs should not exceed 4KB (4096 bytes).

2.6. Operations

Operations are the actions you perform within your account. Creating or deleting containers, uploading or downloading objects, etc. The full list of operations is documented in the ReST API section. Operations may be performed via the ReST web service API or a language-specific API; currently, we support Python, PHP, Java, Ruby, and C#/.NET.

Important

All operations must include a valid authorization token.

2.7. CDN-Enabled Containers

To publish data that is to be served by a Content Distribution Network (CDN), containers which house the data must be CDN-enabled. When a container is CDN-enabled, any files within the container are publicly accessible and do not require an authentication token for read access. Uploading content into a CDN-enabled container is a secure operation and requires a valid authentication token.

Each CDN-enabled container has a unique Uniform Resource Locator (URL) that can be combined with its object names and openly distributed in web pages, emails, or other applications.

For example, a CDN-enabled container named `photos` might be referenced as http://c10171.r71.cf0.rackcdn.com. If that container houses a screenshot called `wow1.jpg`, then

that image can be served by a CDN with the full URL of http://c10171.r71.cf0.rackcdn.com/wow1.jpg. This URL can be embedded in HTML pages, email messages, blog posts, etc. When that URL is accessed, a copy of that image is fetched from the Cloud Files storage system and cached in a CDN and served from there for all subsequent requests for a configurable cache time to live (TTL) value. Setting the TTL of a CDN-enabled container translates to setting the `Expires` and `Cache-Control` HTTP headers. Cloud Files continues to serve content via the CDN until it receives a delete request, although extremely long TTL values do not guarantee that an object is served from a CDN edge location. When the TTL expires, the CDN checks Cloud Files to ensure that it has the most up-to-date content. A purge request forces the CDN to check with Cloud Files for the most up-to-date version of the file.

Containers tracked in the CDN management service are completely separate and distinct from the containers defined in the storage service. It is possible for a container to be CDN-enabled even if it doesn't exist in the storage system. Users may want the ability to pre-generate CDN URLs before actually uploading content and this separation gives them that ability.

However, for the content to be served from the CDN, the container names **MUST** match in both the CDN management service and the storage service. For example, you could CDN-enable a container called `images` and be assigned the CDN URL, but you also need to create a container called `images` in the storage service.

2.8. Language-Specific API Bindings

A set of supported API bindings in several popular languages are available to help put Cloud Files in the hands of developers. These bindings provide a layer of abstraction on top of the base ReST API, allowing programmers to work with a container and object model instead of working directly with HTTP requests and responses. These bindings are free (as in beer and as in speech) to download, use, and modify. They are all licensed under the MIT License as described in the COPYING file packaged with each binding. If you do make any improvements to an API, you are encouraged (but not required) to submit those changes back to us.

The API bindings are hosted at http://github.com/rackspace. Feel free to coordinate your changes through github or, if you prefer, send your changes to cloudfiles@rackspacecloud.com. Just make sure to indicate which language and version you modified and send us a unified diff.

Each binding includes its own documentation (either HTML, PDF, or CHM). They also include code snippets and examples to help you get started. The currently supported API binding for Cloud Files are:

- PHP (requires 5.x and the modules: cURL, FileInfo, mbstring)
- Python (requires 2.4 or newer)
- Java (requires JRE v1.5 or newer)
- C#/.NET (requires .NET Framework v3.5)
- Ruby (requires 1.8 or newer and mime-tools module)

There are no other supported language-specific bindings at this time. You are welcome to create your own language API bindings and we will help answer any questions during development, host your code if you like, and give you full credit for your work.

3. General API Information

3.1. 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 Cloud Files 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 the Rackspace Cloud Authentication Service and supplying a valid username and API access key.

The Rackspace Cloud Authentication Service is a ReSTful web service. It is the entry point to all Rackspace Cloud APIs.

To access the Authentication Service, you must know whether your account is US-based or UK-based:

- US-based accounts authenticate through https://auth.api.rackspacecloud.com/v1.0.
- UK-based accounts authenticate through https://lon.auth.api.rackspacecloud.com/v1.0.

Your account may be based in either the US or the UK; this is not determined by your physical location but by the location of the Rackspace retail site which was used to create your account:

- If your account was created via http://www.rackspacecloud.com, it is a US-based account.
- If your account was created via http:/www.rackspace.co.uk, it is a UK-based account.

If you are unsure how your account was created, use the Rackspace contact information at either site to ask for help.

Request

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

- Use your Rackspace Cloud username as the username for the API. Place it in the `X-Auth-User` x-header.
- Obtain your API access key from the Rackspace Cloud Control Panel in the Your Account | API Access section. Place it in the `X-Auth-Key` x-header.

Example 3.1. Authentication Request (US-Based Account)

```
GET /v1.0 HTTP/1.1
Host: auth.api.rackspacecloud.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`, `X-CDN-Management-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 Cloud Files should be made against the URI specified in `X-Storage-Url` or `X-CDN-Management-Url` and must include the `X-Auth-Token` header.

Example 3.2. Authentication Response

```
HTTP/1.1 204 No Content
Date: Mon, 12 Nov 2007 15:32:21 GMT
Server: Apache
X-Storage-Url: https://storage.clouddrive.com/v1/CF_xer7_34
X-CDN-Management-Url: https://cdn.clouddrive.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` and `X-CDN-Management-Url` will need to be parsed and used in the connection and request line of all subsequent requests against Cloud Files. In the example response above, users connecting to Cloud Files would send most container/object requests with a host header of `storage.clouddrive.com` and the request line's version and account as `/v1/CF_xer7_34`. To CDN-enable Containers or adjust CDN attributes, ReST requests should be sent to `cdn.clouddrive.com`. Note that authentication tokens are valid for a 24 hour period.

3.2. Overview of API Operations

The Cloud Files 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). The length restrictions should be checked against the URL encoded string.

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

There are actually two different sets of ReST services that make up the full Cloud Files product. The first 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. The second ReST service is for managing the CDN feature of Cloud Files and is identified by `X-CDN-Management-Url`.

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, while a **PUT** request against the `X-CDN-Management-Url` is used to CDN-enable a container.

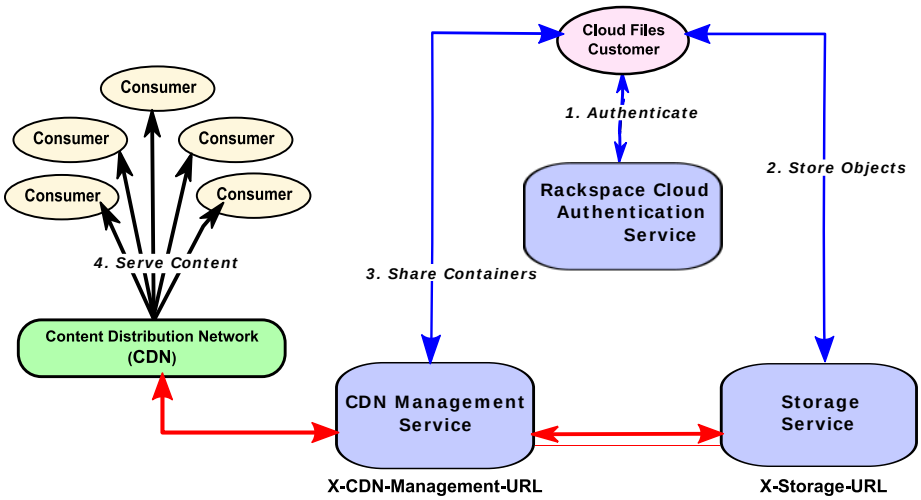
The language-specific APIs mask this system separation from the programmer. They simply create a container and mark it *public* and it handles calling out to the appropriate back-end services using the appropriate ReST API.

Note

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

The following diagram illustrates the various system interfaces and the ease with which content can be distributed over the CDN. The process is simple: authenticate, create a container, upload objects, mark the container as public, and begin serving that content from a powerful CDN.

Figure 3.1. Cloud Files System Interfaces



4. API Operations for Storage Services

The following section describes the ReST API for interacting with the storage component of Cloud Files. 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

4.1. 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 Cloud Files account string:

Example 4.1. Storage Account HTTP Request: General Structure

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

4.1.1. 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 4.2. Containers List Request

```
GET /<api version>/<account> HTTP/1.1
Host: storage.clouddrive.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 4.3. Containers List Response

```
HTTP/1.1 200 Ok
Date: Thu, 07 Jun 2007 18:57:07 GMT
Server: Apache
Content-Type: text/plain; charset=UTF-8
Content-Length: 32
```

```
images
movies
documents
backups
```

4.1.1.1. 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 4.4. Containers Details Request: JSON

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

Example 4.5. 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 4.6. Containers Details Request: XML

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

Example 4.7. 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>
```

4.1.1.2. 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 4.8. 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.clouddrive.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.clouddrive.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.clouddrive.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.

4.1.2. Retrieve Account Metadata

HEAD operations against an account are performed to retrieve the number of containers and the total bytes stored in Cloud Files 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 4.9. Account Metadata Request

```
HEAD /<api version>/<account> HTTP/1.1
Host: storage.clouddrive.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 4.10. Account Metadata Response

```
HTTP/1.1 204 No Content
Date: Thu, 07 Jun 2007 18:57:07 GMT
```

```
Server: Apache
X-Account-Container-Count: 3
X-Account-Total-Bytes-Used: 323479
```

4.2. 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 4.11. Storage Container HTTP Request: General Structure

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

4.2.1. 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 4.12. Objects List Request

```
GET /<api version>/<account>/<container>[?parm=value] HTTP/1.1
Host: storage.clouddrive.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 4.13. Objects List Response

```
HTTP/1.1 200 Ok
Date: Thu, 07 Jun 2007 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
```

4.2.1.1. 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 4.14. Objects Details Request: JSON

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

Example 4.15. 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 4.16. Objects Details Request: XML

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

Example 4.17. 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>
```

4.2.1.2. 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 4.18. 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.clouddrive.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.clouddrive.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.clouddrive.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.

4.2.1.3. Pseudo-Hierarchical Folders/Directories

You can simulate a hierarchical structure in Cloud Files 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.clouddrive.com/v1/CF_xer7_343/backups/photos/me.jpg`

Example 4.19. 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.clouddrive.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.clouddrive.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.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

4.2.2. 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 4.20. Container Create Request

```
PUT /<api version>/<account>/<container> HTTP/1.1
Host: storage.clouddrive.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 4.21. 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 UTF8 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 4.22. 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 4.23. 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
```

4.2.3. 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 4.24. Container Delete Request

```
DELETE /<api version>/<account>/<container> HTTP/1.1
Host: storage.clouddrive.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 4.25. Container Delete Response

```
HTTP/1.1 204 No Content
Date: Thu, 07 Jun 2007 18:57:07 GMT
Server: Apache
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
```

4.2.4. 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 4.26. Container Metadata Request

```
HEAD /<api version>/<account>/<container> HTTP/1.1
Host: storage.clouddrive.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 4.27. Container Metadata Response

```
HTTP/1.1 204 No Content
Date: Wed, 16 Mar 2010 19:37:41 GMT
Content-type: text/html
X-Container-Object-Count: 7
X-Container-Bytes-Used: 413
X-Container-Meta-InspectedBy: JackWolf
```

4.3. 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.

4.3.1. 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. Cloud Files 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, Cloud Files does not support the full specification for `Range` but basic support is provided. Cloud Files 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 4.28. Retrieve Object Request

```
GET /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.clouddrive.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 4.29. Retrieve Object Response

```
HTTP/1.1 200 Ok
Date: Wed, 11 Jul 2007 19:37:41 GMT
Server: Apache
Last-Modified: Fri, 12 Jun 2007 13:40:18 GMT
ETag: b0dffe8254d152d8fd28f3c5e0404a10
Content-type: text/html
Content-Length: 512000
```

```
[ ... ]
```

4.3.2. 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 4.30. Create/Update Object Request

```
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: 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 4.31. Create/Update Object Response

```
HTTP/1.1 201 Created
Date: Thu, 07 Jun 2007 18:57:07 GMT
Server: Apache
ETag: d9f5eb4bba4e2f2f046e54611bc8196b
Content-Length: 0
```

```
Content-Type: text/plain; charset=UTF-8
```

4.3.2.1. 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 Cloud Files 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 4.32. 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 4.33. 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 4.34. 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.

4.3.2.2. 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 Cloud Files 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 Cloud Files, 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 4.35. Upload Unspecified Quantity of Content

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

4.3.2.3. 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 4.36. 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
```

4.3.2.4. 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 4.37. 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
```

4.3.2.5. 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 4.38. 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
```

4.3.3. 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 Cloud Files. 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.

4.3.4. 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 4.39. Object Delete Request

```
DELETE /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.clouddrive.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 4.40. Object Delete Response

```
HTTP/1.1 204 No Content
Date: Thu, 07 Jun 2007 20:59:39 GMT
Server: Apache
Content-Type: text/plain; charset=UTF-8
```

4.3.5. 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 4.41. Object Metadata Request

```
HEAD /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.clouddrive.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 4.42. Object Metadata Response

```
HTTP/1.1 200 OK
Date: Thu, 07 Jun 2007 20:59:39 GMT
Server: Apache
Last-Modified: Fri, 12 Jun 2007 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
```

4.3.6. 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 4.43. Update Object Metadata Request

```
POST /<api version>/<account>/<container>/<object> HTTP/1.1
Host: storage.clouddrive.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 4.44. Update Object Metadata Response

```
HTTP/1.1 202 Accepted
Date: Thu, 07 Jun 2007 20:59:39 GMT
Server: Apache
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
```

5. API Operations for CDN Services

The following is a description of API calls that can be used for CDN account and container operations. All of the ReST methods described below must be issued against the CDN management service as defined in the `X-CDN-Management-Url` returned by a successful authentication.

5.1. CDN Account Operations

This section describes the methods allowed against the account portion URI and conform to the following format:

Example 5.1. CDN HTTP Request: General Structure

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

5.1.1. List CDN-Enabled Containers

GET operations against the `X-CDN-Management-Url` for an account are performed to retrieve a list of existing CDN-enabled containers. Like the storage system's **GET** container, the CDN management service allows the following query parameters:

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.
`enabled_only`	Set to `true` to return only the CDN-enabled containers.

Using the `format` query parameter, you can request the output in a serialized format in either JSON or XML.

Using `limit` and `marker` provides a mechanism to iterate through the entire list of containers. Keep in mind that the value for marker will need to be URL encoded before issuing the request.

There is also support for filtering the list to return only the list of containers that are currently CDN-enabled. Passing in a query parameter of `?enabled_only=true` will suppress any *private* containers from appearing in the list.

The list of CDN-enabled containers is returned in the response body, one container name per line.

Example 5.2. CDN-Enabled Containers List Request

```
GET /<api version>/<account> HTTP/1.1
Host: cdn.clouddrive.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 5.3. CDN-Enabled Containers List Response

```
HTTP/1.1 200 Ok
Date: Thu, 07 Jun 2007 18:57:07 GMT
Server: Apache
Content-Type: text/plain; charset=UTF-8
Content-Length: 13
```

```
images
movies
```

5.1.1.1. Serialized List Output

If a `format=xml` or `format=json` argument is appended to the CDN management URL, the service will serve extended container 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 5.4. CDN-Enabled Containers Details Request: JSON

```
GET /v1/<account>?format=json HTTP/1.1
Host: cdn.clouddrive.com
X-Auth-Token: a6e3359b-3749-440a-9292-0bdcb0e33617
```

Example 5.5. CDN-Enabled Containers Details Response: JSON

```
HTTP/1.1 200 OK
Date: Mon, 09 Mar 2009 20:07:47 GMT
Server: Apache
Content-Length: 127
Content-Type: application/json; charset=utf-8
```

```
[
{"name":"test_container",
 "cdn_enabled":"true",
 "ttl":28800,
 "log_retention":"true",
 "cdn_uri":"http://c2.r2.cf1.rackcdn.com",
```

```
 "cdn_ssl_uri":"https://c2.ssl.cf1.rackcdn.com"}
]
```

Example 5.6. CDN-Enabled Containers Details Request: XML

```
GET /v1/<account>?format=xml HTTP/1.1
Host: cdn.clouddrive.com
X-Auth-Token: a6e3359b-3749-440a-9292-0bdcb0e33617
```

Example 5.7. CDN-Enabled Containers Details Response: XML

```
HTTP/1.1 200 OK
Date: Mon, 09 Mar 2009 20:11:27 GMT
Server: Apache
Content-Length: 267
Content-Type: application/xml; charset=utf-8
```

```
<?xml version="1.0" encoding="UTF-8"?>
<account name="WidgetsRNotUs.invalid">
  <container>
      <name>images</name>
      <cdn_enabled>True</cdn_enabled>
      <ttl>86400</ttl>
      <log_retention>True</log_retention>
      <cdn_url>
          http://c2.r2.cf1.rackcdn.com
      </cdn_url>
      <cdn_ssl_url>
          https://c2.ssl.cf1.rackcdn.com
      </cdn_ssl_url>
 </container>
</account>
```

5.2. CDN Container Services

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

Example 5.8. CDN-Enabled Container HTTP Request: General Structure

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

Containers tracked in the CDN management service are separate and distinct from the containers defined in the storage service. It is possible for a container to be CDN-enabled even if it doesn't exist in the storage system. Users may want the ability to pre-generate CDN URLs before actually uploading content; this separation gives them that ability.

However, for the content to be served from the CDN, the container names **MUST** match in both the CDN management service and the storage service. For example, you could CDN-enable a container called `images` and be assigned the CDN URL, but you also need to create a container called `images` in the storage service and populate it with the content you want to serve over the CDN.

5.2.1. CDN-Enabled Container

PUT operations against a container are used to initially CDN-enable the container and set its attributes.

When a container is CDN-enabled, any objects stored in that container are publicly accessible over a CDN by combining the container's CDN URI with the object name. Any objects accessed will be cached in the CDN for Time To Live or TTL(value) number of seconds; the default is 72 hours or 259200 seconds. On the next access after the TTL expiration, the CDN will re-fetch the object and cache it again for another TTL(value) seconds. The minimum TTL that can be set is 15 minutes (900 seconds); the maximum TTL is 50 years (range of 900 to 1577836800 seconds).

To specify the TTL, include an HTTP header of `X-TTL: integer_seconds` Setting the TTL is the same as setting the HTTP `Expires` and `Cache-Control` headers for the cached object. Setting a TTL for a long time, such as 5 years, does not guarantee that their content will stay populated on CDN edge servers for the entire five-year period. The most popular objects stay cached based on the edge locations logic.

Example 5.9. Container CDN-Enable Request

```
PUT /<api version>/<account>/<container> HTTP/1.1
Host: cdn.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
X-TTL: 2592000
X-Log-Retention: True
```

No content is returned. A status code of 201 (Created) indicates that the container was CDN-enabled as requested. The response will contain an HTTP header to indicate the URL

that can be combined with object names to serve objects through the CDN. If the container is already CDN-enabled, a 202 (Accepted) response is returned and the TTL is adjusted.

Example 5.10. Container CDN-Enable Response

```
HTTP/1.1 201 Created
Date: Thu, 07 Jun 2007 18:50:19 GMT
Server: Apache
Content-Type: text/plain; charset=UTF-8
X-CDN-URI: http://c10171.r71.cf0.rackcdn.com
X-CDN-SSL-URI: http://c10171.ssl.cf0.rackcdn.com
```

5.2.2. List CDN-Enabled Container Metadata

HEAD operations against a CDN-enabled container are used to determine the CDN attributes of the container.

If the container is (or ever has been) CDN-enabled, the URI, TTL, enabled status, and log retention status are returned in the response headers. Its CDN URI can be combined with any object name within the container to form the publicly accessible URL for that object for distribution over a CDN system. The TTL value is the number of seconds that the object will be cached in the CDN system before being refetched. The enabled status indicates whether the container is currently marked to allow public serving of objects via CDN. The log_retention setting specifies whether the CDN access logs should be collected and stored in the Cloud Files storage system.

Example 5.11. CDN-Enabled Container Metadata Request

```
HEAD /<api version>/<account>/<container> HTTP/1.1
Host: cdn.clouddrive.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 CDN attributes are returned in HTTP headers. If SSL is available for the container, an X-CDN-SSL-URI header is returned in addition to X-CDN-URI.

Example 5.12. CDN-Enabled Container Metadata Response

```
HTTP/1.1 204 No Content
Date: Wed, 11 Jul 2007 19:37:41 GMT
Content-type: text/html
X-CDN-Enabled: True
X-CDN-URI: http://c10171.r71.cf0.rackcdn.com
X-CDN-SSL-URI: https://c10171.ssl.cf0.rackcdn.com
X-TTL: 86400
X-Log-Retention: True
```

5.2.3. Purge CDN-Enabled Containers or Objects

DELETE operations against a CDN-enabled container or object are used to remove an outdated or unwanted object from the CDN. You can manually purge CDN-enabled objects

or containers without having to wait for the TTL to expire, and you can optionally be notified by email that the object has been purged. There are two methods for purging content from the edge: one for purging individual objects, one for purging entire containers.

Example 5.13. Purge CDN-Enabled Object

```
DELETE /<api version>/<account>/<object> HTTP/1.1
Host: cdn.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
X-Purge-Email: user@domain.com
```

Example 5.14. Purge CDN-Enabled Container

```
DELETE /<api version>/<account>/<container> HTTP/1.1
Host: cdn.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
X-Purge-Email: user@domain.com, user2@domain.com, user3@domain.com
```

A 204 No Content response is returned. The system purges the object from the CDN, and sends an email to the indicated address or multiple addresses. The email address is optional. You can enter a comma-separated list of addresses if you want to notify more than one person about the deletion. A status code of 204 (No Content) indicates success; 404 (Not Found) is returned if the requested container was never CDN-enabled, and it returns a 403 if an authorization problem occurs. The CDN URI is returned in the HTTP header, X-CDN-URI. Purging a container may take a long time, 45 minutes or longer, please be patient while waiting for a response.

Example 5.15. Purge CDN-Enabled Container or Object Response

```
HTTP/1.1 204 No Content
Date: Thu, 13 Jan 2010 18:57:07 GMT
Server: Apache
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
```

5.2.4. Update CDN-Enabled Container Metadata

POST operations against a CDN-enabled container are used to adjust CDN attributes.

The **POST** operation can be used to set a new TTL cache expiration value or to enable/disable public sharing over the CDN. Keep in mind that if you have content currently cached in the CDN, setting your container back to private will NOT purge the CDN cache; you will have to wait for the TTL to expire.

Example 5.16. Update CDN-Enabled Container Metadata Request

```
POST /<api version>/<account>/<container> HTTP/1.1
```

```
Host: cdn.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
X-TTL: 86400
X-CDN-Enabled: True
X-Log-Retention: True
```

No content is returned. A status code of 202 (Accepted) indicates success; 404 (Not Found) is returned if the requested container was not found. The CDN URI and the CDN SSL URI are both returned in the HTTP headers, X-CDN-URI and X-CDN-SSL-URI.

Example 5.17. Update CDN-Enabled Container Metadata Response

```
HTTP/1.1 204 No Content
Date: Thu, 07 Jun 2011 18:57:07 GMT
Server: Apache
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
X-CDN-URI: http://c10171.r71.cf0.rackcdn.com
X-CDN-SSL-URI: https://c10171.ssl.cf0.rackcdn.com
```

5.2.5. CDN-Enabled Containers Served via SSL

HEAD operations against a CDN-enabled container can also return an SSL URI. When SSL is available, another header gets returned with calls to the CDN Management URL, X-CDN-SSL-URI, in addition to X-CDN-URI. This feature enables users to use https protocol in URLS used for requesting objects stored in CDN-enabled containers.

Example 5.18. CDN-Enabled Container Metadata Requests with SSL

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

No content is returned. A status code of 202 (Accepted) indicates success; 404 (Not Found) is returned if the requested container was not found. The CDN SSL URI is returned in the HTTP header, X-CDN-SSL-URI.

Example 5.19. CDN-Enabled Container Metadata with SSL

```
HTTPS/1.1 204 No Content
Date: Thu, 07 Jan 2011 18:57:07 GMT
Server: Apache
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
X-CDN-URI: http://c10171.r71.cf0.rackcdn.com
X-CDN-SSL-URI: https://c10171.ssl.cf0.rackcdn.com
X-CDN-Enabled: True
X-TTL: 259000
X-Log-Retention: False
Connection: close
Content-Type: text/plain; charset=UTF-8
```

6. Troubleshooting

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

6.1. 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.

6.1.1. 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 6.1. cURL Authenticate

```
curl -D - \
     -H "X-Auth-Token: a86850deb2742ec3cb41518e26aa2d89" \
     -H "X-Auth-User: jdoe" \
     https://auth.api.rackspacecloud.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.clouddrive.com/v1/CF_xer7_343
X-CDN-Management-Url: https://cdn.clouddrive.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, CDN management 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 and CDN services.

6.1.2. 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 6.2. cURL Get Storage Space

```
curl -X HEAD -D - \
     -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
     https://storage.clouddrive.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.

6.1.3. Creating a Storage Container

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

Example 6.3. cURL Create Storage Container

```
curl -X PUT -D - \
     -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
     https://storage.clouddrive.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.

6.1.4. 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. Please note that if you intend to distribute this object via the CDN you MUST make sure that the object's `Content-Type` is set correctly. This is the mechanism by which a user's web browser knows how to display the file or launch a helper application to view the file.

Example 6.4. 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.clouddrive.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
```

6.1.5. CDN-Enabling the Container

After creating a container and storing a file in it, you can choose to share the file. Since the data in Cloud Files is all private, you can share your screenshot via the CDN. To CDN-enable a container, issue a **PUT** request against the CDN management service. The default TTL is 72 hours and supports a minimum of 15 minutes (900 seconds) and a maximum of 50 years (1577836800 seconds). Note that the target URL specifies the CDN system.

Example 6.5. cURL CDN-Enable Container

```
curl –X PUT -D - \
     -H "X-Auth-Token: fc81aaa6-98a1-9ab0-94ba-aba9a89aa9ae" \
     -H "X-CDN-Enabled: True" \
     -H "X-TTL: 259200" \
     https://cdn.clouddrive.com/v1/CF_xer7_343/images
```

```
HTTP/1.1 202 Accepted
Date: Thu, 06 Aug 2009 01:34:13 GMT
Server: Apache
X-CDN-URI: http://c10171.r71.cf0.rackcdn.com
X-CDN-SSL-URI: https://c10171.ssl.cf0.rackcnd.com
Content-Length: 0
Connection: close
Content-Type: text/plain; charset=UTF-8
```

When the container is CDN-enabled, the service returns its public URI in the `X-CDN-URI` header of the response, plus the SSL URL in the `X-CDN-SSL-URL` header of the response. Now you can combine this URI with the object name to access the file via the CDN, or use the https:// URI in combination with the object name to access the file over a secure SSL connection via the CDN.

You can verify the CDN's cache settings that you specified with your TTL value by sending a **GET** request to the object's CDN URL and viewing the response headers. The TTL value you specify translates to the `Expires` and `Cache-Control` headers of the CDN's cached Object.

The cURL command below issues a **GET** request which downloads the entire file but writes it to `/dev/null`, a data sink that won't actually save the content to your local drive (This is only valid on UNIX-like systems).

Example 6.6. cURL Download a File

```
curl –s -D - \
  http://c10171.r71.cf0.rackcdn.com/wow1.jpg \
  -O /dev/null
```

```
HTTP/1.1 200 OK
Date: Thu, 06 Aug 2009 01:40:12 GMT
Server: Apache
Expires: Fri, 07 Aug 2009 01:40:12 GMT
Last-Modified: Thu, 09 Jul 2009 17:14:46 GMT
Cache-Control: max-age=86400, public
ETag: b20237bff6828976d2eb348e1ca8adae
Content-Length: 1255764
Content-Type: image/jpeg
Connection: keep-alive
```

6.1.6. Other cURL Commands

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

It should be noted that generally each time `curl` is invoked to perform an operation, a separate TCP/IP and SSL connection is created and thrown away. The language APIs. however. are designed to re-use these connections between operations and therefore provide much better performance. It is recommended that you use one of the supported language APIs in your production applications and limit `curl` to quick-and-easy testing/ troubleshooting.