

Cloud Files™

Developer Guide

API v1 (Apr 25, 2012)



docs.rackspace.com/api

Cloud Files™ Developer Guide

API v1 (2012-04-25)

Copyright © 2009-2012 Rackspace US, Inc. All rights reserved.

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."

RACKSPACE MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, AS TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS DOCUMENT AND RESERVES THE RIGHT TO MAKE CHANGES TO SPECIFICATIONS AND PRODUCT/SERVICES DESCRIPTION AT ANY TIME WITHOUT NOTICE. RACKSPACE SERVICES OFFERINGS ARE SUBJECT TO CHANGE WITHOUT NOTICE. USERS MUST TAKE FULL RESPONSIBILITY FOR APPLICATION OF ANY SERVICES MENTIONED HEREIN. EXCEPT AS SET FORTH IN RACKSPACE GENERAL TERMS AND CONDITIONS AND/OR CLOUD TERMS OF SERVICE, RACKSPACE ASSUMES NO LIABILITY WHATSOEVER, AND DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO ITS SERVICES INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT.

Except as expressly provided in any written license agreement from Rackspace, the furnishing of this document does not give you any license to patents, trademarks, copyrights, or other intellectual property.

Rackspace®, Rackspace logo and Fanatical Support® are registered service marks of Rackspace US, Inc. All other product names and trademarks used in this document are for identification purposes only and are property of their respective owners.

Table of Contents

1. Overview	1
1.1. Intended Audience	1
1.2. Document Change History	2
1.3. Additional Resources	2
2. Concepts	4
2.1. Accounts	4
2.2. Authentication	4
2.3. Permissions	4
2.4. Containers	4
2.5. Objects	5
2.6. Operations	5
2.7. CDN-Enabled Containers	5
2.8. Language-Specific API Bindings	6
3. General API Information	8
3.1. Authentication	8
3.2. Overview of API Operations	10
4. API Operations for Storage Services	12
4.1. Storage Account Services	12
4.1.1. List Containers	12
4.1.2. Retrieve Account Metadata	15
4.2. Storage Container Services	16
4.2.1. List Objects in a Container	16
4.2.2. Create Container	21
4.2.3. Delete Container	22
4.2.4. Retrieve Container Metadata	23
4.2.5. Create/Update Container Metadata	23
4.3. Storage Object Services	24
4.3.1. Retrieve Object	24
4.3.2. Create/Update Object	25
4.3.3. Copy Object	31
4.3.4. Delete Object	32
4.3.5. Retrieve Object Metadata	32
4.3.6. Update Object Metadata	33
5. API Operations for CDN Services	35
5.1. CDN Account Services	35
5.1.1. List CDN-Enabled Containers	35
5.2. CDN Container Services	38
5.2.1. CDN-Enable a Container	38
5.2.2. List a CDN-Enabled Container's Metadata	39
5.2.3. Purge CDN-Enabled Containers	40
5.2.4. Update CDN-Enabled Container Metadata	40
5.2.5. CDN-Enabled Containers Served via SSL	41
5.2.6. Streaming CDN-Enabled Containers	41
5.3. CDN Object Services	42
5.3.1. Purge CDN-Enabled Objects	42
6. Public Access to Your Cloud Files Account	44
6.1. TempURL	44
6.1.1. Set Account Metadata Key	44

6.1.2. Create the TempURL	45
6.2. FormPost	45
6.2.1. Set Account Metadata Key	45
6.2.2. Create the Form	46
7. Examples and Troubleshooting	48
7.1. Using cURL	48
7.2. Authentication with cURL	48
7.3. Determining Storage Usage with cURL	49
7.4. Creating a Storage Container with cURL	50
7.5. Uploading a Storage Object with cURL	50
7.6. CDN-Enabling the Container with cURL	51
7.7. Other cURL Commands	52

List of Figures

3.1. Cloud Files System Interfaces	11
--	----

List of Examples

3.1. Authentication Request (US-Based Account)	8
3.2. Authentication Response	9
4.1. Storage Account HTTP Request: General Structure	12
4.2. Containers List Request	12
4.3. Containers List Response	13
4.4. Container Details Request: JSON	13
4.5. Container Details Response: JSON	13
4.6. Container Details Request: XML	13
4.7. Container Details Response: XML	14
4.8. List Large Number of Containers	14
4.9. Account Metadata Request	15
4.10. Account Metadata Response	15
4.11. Storage Container HTTP Request: General Structure	16
4.12. Objects List Request	16
4.13. Objects List Response	17
4.14. Object Details Request: JSON	17
4.15. Object Details Response: JSON	17
4.16. Object Details Request: XML	18
4.17. Object Details Request: XML	18
4.18. List Large Number of Objects	18
4.19. Pseudo-Hierarchical Folders/Directories	20
4.20. Container Create Request	21
4.21. Container Create Response	21
4.22. Container Create Request with Metadata	22
4.23. Container Create Response	22
4.24. Container Delete Request	22
4.25. Container Delete Response	22
4.26. Container Metadata Request	23
4.27. Container Metadata Response	23
4.28. Update Container Metadata Request	23
4.29. Update Container Metadata Response	23
4.30. View Container Metadata Request	24
4.31. View Container Metadata Response	24
4.32. Retrieve Object Request	25
4.33. Retrieve Object Response	25
4.34. Create/Update Object Request	26
4.35. Create/Update Object Response	26
4.36. Upload Segment of a Large Object	27
4.37. Upload Next Segment of the Large Object	27
4.38. Upload Manifest	28
4.39. Upload Unspecified Quantity of Content	28
4.40. Assign CORS Header	29
4.41. Content-Encoding Header Example	29
4.42. Content-Disposition Header Example	30
4.43. Delete At Example	30
4.44. Delete After Example	31
4.45. Object Delete Request	32
4.46. Object Delete Response	32

4.47. Object Metadata Request	32
4.48. Object Metadata Response	33
4.49. Update Object Metadata Request	33
4.50. Update Object Metadata Response	33
5.1. CDN HTTP Request: General Structure	35
5.2. CDN-Enabled Containers List Request	36
5.3. CDN-Enabled Containers List Response	36
5.4. CDN-Enabled Containers Details Request: JSON	36
5.5. CDN-Enabled Containers Details Response: JSON	36
5.6. CDN-Enabled Containers Details Request: XML	37
5.7. CDN-Enabled Containers Details Response: XML	37
5.8. CDN-Enabled Container HTTP Request: General Structure	38
5.9. Container CDN-Enable Request	38
5.10. Container CDN-Enable Response	39
5.11. Container CDN-Disable Request	39
5.12. CDN-Enabled Container Metadata Request	39
5.13. CDN-Enabled Container Metadata Response	40
5.14. Update CDN-Enabled Container Metadata Request	40
5.15. Update CDN-Enabled Container Metadata Response	40
5.16. CDN-Enabled Container Metadata Requests with SSL	41
5.17. CDN-Enabled Container Metadata with SSL	41
5.18. CDN-Enabled Container Metadata Requests (Streaming Enabled)	41
5.19. CDN-Enabled Container Metadata (Streaming Enabled)	42
5.20. Purge CDN-Enabled Object	42
5.21. Purge CDN-Enabled Object Response	43
6.1. Set Account Metadata Key for Public Access	44
6.2. Create TempURL	45
6.3. Set Account Metadata Key for Public Access	46
6.4. Layout of Web Form	46
6.5. Generate Signature for Form Post	47
7.1. cURL Authenticate	48
7.2. cURL Get Storage Space	49
7.3. cURL Create Storage Container	50
7.4. cURL Upload Storage Object	50
7.5. cURL CDN-Enable Container	51
7.6. cURL Download a File	52

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 interface (ReST: Representational State Transfer). There are also language-specific APIs that utilize the ReSTful API but make it much easier for developers to integrate into their applications.

For more details on the 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 are 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.

For additional information about available language bindings, refer to [Language-Specific API Bindings](#).

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
Apr 25, 2012	<ul style="list-style-type: none"> Added TempURL and FormPost.
Mar 19, 2012	<ul style="list-style-type: none"> Fixed doc tickets.
Feb 6, 2012	<ul style="list-style-type: none"> Revisions to clarify issues brought up in doc tickets. Formatted HEAD like other commands. Standardized on URL. Added Expiring Objects and ServiceNet information.
Jan 12, 2012	<ul style="list-style-type: none"> Revisions for the addition of expiring object functionality plus doc bug fixes including adding more cross-references for finding language bindings.
Nov 15, 2011	<ul style="list-style-type: none"> Revised information about how to perform a CDN purge, indicating you must contact support to request a container purge operation.
Oct 21, 2011	<ul style="list-style-type: none"> Added more detail about reasons to perform a CDN purge, clarifying that it isn't required for deleting objects.
Sep 13, 2011	<ul style="list-style-type: none"> Added information about streaming containers to support this new streaming feature, including changing examples to match the streaming headers and URLs returned.
Jun 29, 2011	<ul style="list-style-type: none"> In the 6.1.1 Authorization example, changed X-Auth-Token to X-Auth-Key.
Jun 15, 2011	<ul style="list-style-type: none"> Added best practices for authentication tokens.
May 24, 2011	<ul style="list-style-type: none"> Added information about new headers including CORS headers.
Apr 20, 2011	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> Added information about large object support.
Mar 17, 2011	<ul style="list-style-type: none"> Added information about container metadata.
Mar 10, 2011	<ul style="list-style-type: none"> Added a section about retrieving an SSL URL for CDN-enabled containers that are using https protocol. Updated examples to contain SSL as appropriate.
Feb 25, 2011	<ul style="list-style-type: none"> Added information about the edge purge capability for CDN-enabled containers and objects.
Feb 18, 2011	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> Expanded authentication information for UK release. Added "delimiter" as a Query Parameter and server-side object copy example.
May 5, 2008	<ul style="list-style-type: none"> 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 official Rackspace support channels, including knowledge base articles, forums, phone, chat, and email.

Language bindings are available. 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)

For additional information about available language bindings, refer to [Language-Specific API Bindings](#).

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 during the time it is valid.



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. If you expect to have containers with millions of objects, we recommend organizing the object storage across multiple containers.

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

Objects are the basic storage entities in Cloud Files. They represent the files and their optional metadata you upload to the system. When you upload objects to Cloud Files, the data is stored as-is (without compression or encryption) and consists of a location (container), the object's name, and any metadata you assign consisting of key/value pairs. For instance, you may choose 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 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 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. Make sure to indicate which language and version you modified and send us a unified diff.

Each binding includes its own documentation (in HTML, PDF, or CHM format). 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. The headers can be processed as is, and 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 URL 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-81b4-663c99e1cbb
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`. Authentication tokens are valid for a 24-hour period. Once a token expires, you must request a new one.

Note: To CDN-enable Containers or adjust CDN attributes, requests must be sent to `cdn.clouddrive.com`.

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](#) on ReST for more information.

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

- Maximum number of HTTP headers per request: 90
- Maximum length of all HTTP headers: 4096 bytes
- Maximum length per HTTP request line: 8192 bytes
- Maximum length of HTTP request: 5 gigabytes
- Maximum length of container name: 256 bytes
- Maximum length of object name: 1024 bytes

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

Each ReST request against the 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 URLs, 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.

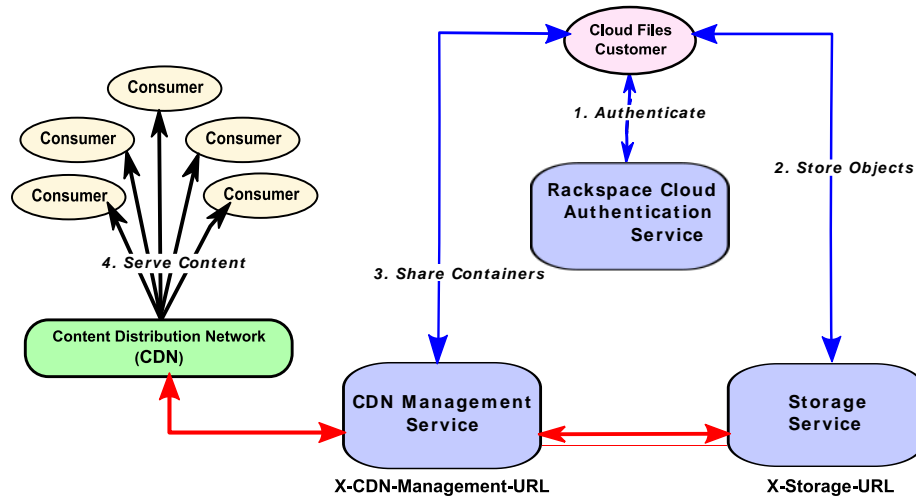


Note

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 requirements for the use of the storage services:

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

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

4.1. Storage Account Services

The following operations can be performed at the account level of the URL. For example, the URL for the requests below will end with the 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 sort order for the name is based on a [binary comparison](#), a single built-in collating sequence that compares string data using SQLite's `memcmp()` function, regardless of text encoding. The following list describes the optional query parameters that are supported with this request.

Query Parameters

- `limit` For an integer value n , limits the number of results to n values.
- `marker` Given a string value x , return object names greater in value than the specified marker. Only strings using UTF-8 encoding are valid.
- `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. Container 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. Container 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. Container 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. Container 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. Controlling a Large List of Containers

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

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

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

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-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 in a Container

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

<code>limit</code>	For an integer value n , limits the number of results to n values.
<code>marker</code>	Given a string value x , return object names greater in value than the specified marker. Only strings using UTF-8 encoding are valid.
<code>prefix</code>	For a string value x , causes the results to be limited to object names beginning with the substring x .
<code>format</code>	Specify either <code>json</code> or <code>xml</code> to return the respective serialized response.
<code>path</code>	For a string value x , return the object names nested in the pseudo path (assuming preconditions are met - see below).
<code>delimiter</code>	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.cloudrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

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

Example 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. Object 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. Object 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. Object Details Request: XML

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

Example 4.17. Object 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>b039efe731ad111bcb0ef221c3849d0</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. Controlling a Large List of Objects

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

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

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

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

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



Note

In the example below, the objects reside in a container called `backups`. Within that container, the objects are organized in a pseudo-directory called `photos`.

Keep in mind that the container name is not displayed in the example, but that it is a part of the object URLs. For instance, the URL of the picture `me.jpg` is `https://storage.clouddrive.com/v1/CF_xer7_343/backups/photos/me.jpg`.

Example 4.19. Pseudo-Hierarchical Folders/Directories

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

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

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

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

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

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

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

```
photos/
```

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

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

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

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

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

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

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

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

4.2.2. Create Container

PUT operations against a storage container are used to create that container. You may create up to 500,000 containers in your Cloud Files account.

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, with 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 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 if the container existed prior to the request. If you request a **PUT** to a container with an X-Container-Meta- prefix in the header, your **GET/HEAD** request responses carry the metadata prefix from the container in subsequent requests.

Example 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 permanently remove it. The container must be empty before it can be deleted.

A **HEAD** request against the container will list any objects it contains.

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
```

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; a 409 (Conflict) is returned if the container is not empty. No response body is 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
```

4.2.5. Create/Update Container Metadata

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

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

Example 4.28. Update Container Metadata Request

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

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

Example 4.29. Update Container Metadata Response

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

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

Example 4.30. View Container Metadata Request

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

Example 4.31. View Container Metadata Response

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

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 5 GB and must have names that do not exceed 1024 bytes after URL encoding. If you try to upload an object larger than 5 GB, a 413 (Request Entity Too Large) code will be returned. However, objects larger than 5 GB may be segmented and 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 4.32. 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 4.33. 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: b0df8e8254d152d8fd28f3c5e0404a10
Content-type: text/html
Content-Length: 512000
```

```
[ ... ]
```

4.3.2. Create/Update Object

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

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

You can cause an object to expire after a certain date by using the `X-Delete-At` or `X-Delete-After` headers during an object **PUT** operation. When Cloud Files detects one of

these headers, the system automatically stops serving that object at the specified time and shortly after the expiration date, it removes the object from the storage system.

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

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

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

Example 4.34. 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 411 (Length Required) denotes a missing `Content-Length` or `Content-Type` header in the request. If the MD5 checksum of the data written to the storage system does NOT match the (optionally) supplied ETag value, a 422 (Unprocessable Entity) response is returned.

Example 4.35. 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 5 GB 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 4.36. Upload Segment of a Large Object

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

s

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

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

Example 4.37. 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 4.38. 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 than 5 GB with this method, the server will close the TCP/IP connection after 5 GB and purge the customer data from the system. Users must take responsibility for ensuring the data they transfer will be less than 5 GB or for splitting it into 5 GB chunks, each in its own storage object. If you have files that are larger than 5 GB 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 4.39. 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 4.40. 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.

The object must be compressed before it is uploaded. Cloud Files does not do any automatic compression. The content-encoding header is there to allow the client to set the metadata appropriately.

Example 4.41. 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 browser prompts to save the file rather than displaying it using default browser settings.

Example 4.42. 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
```

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

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

The X-Delete-At header requires a Unix Epoch timestamp, in integer form; for example: 1348691905 represents Wed, 26 Sep 2012 20:38:25 GMT. By setting the header to a specific Epoch time, you indicate when you want the object to expire, not be served, and be deleted completely from the storage system.

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

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

Example 4.43. Delete At Example

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

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

Example 4.44. Delete After Example

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

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

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>`. Unlike the first method, this method does not require a `Content-Length` header.

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

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

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

Your account will not be charged when you copy or move your objects within the same Data Center using the internal network host name. The name will be the Cloud Files storage URL with "snet-" pre-pended to it. You may locate your Data Center within the control panel. The Data Center name can be in the Storage URL returned when authenticating to Cloud Files.

4.3.4. Delete Object

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

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

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

Example 4.45. 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 code 404 (Not Found) is returned when the object does not exist.

Example 4.46. 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 4.47. 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.

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

Example 4.48. 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 or to assign headers not already assigned such as **X-Delete-At** or **X-Delete-After** for expiring objects. You cannot use the **POST** operation to change any of the object's other headers such as **Content-Type**, **ETag**, etc. It is not used to upload storage objects (see **PUT**). Also refer to [copying an object](#) when you need to update metadata or other headers such as **Content-Type** or **CORS** headers.

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

Example 4.49. 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 if the requested object does not exist.

Example 4.50. 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 the Content Delivery Network (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. See the section on authentication for more details.

5.1. CDN Account Services

The following operations may be performed at the account level of the CDN account. The operations 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

<code>limit</code>	For an integer value <i>n</i> , limits the number of results to <i>n</i> values.
<code>marker</code>	Given a string value <i>x</i> , return object names greater in value than the specified marker. Only strings using UTF-8 encoding are valid.
<code>format</code>	Specify either <code>json</code> or <code>xml</code> to return the respective serialized response.
<code>enabled_only</code>	Set to <code>true</code> 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, 08 Sep 2011 14:35:45 GMT
Server: Apache/2.2.14 (Ubuntu)
Transfer-Encoding: chunked
Content-Type: text/plain
```

```
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: Thu, 08 Sep 2011 14:41:29 GMT
Server: Apache/2.2.14 (Ubuntu)
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",
    "cdn_streaming_uri": "https://c2.r2.stream.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: Fri, 09 Sep 2011 17:57:28 GMT
Server: Apache/2.2.14 (Ubuntu)
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>
    <cdn_streaming_url>
      https://c2.r2.stream.cf1.rackcdn.com
    </cdn_streaming_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
```

When you CDN-enable a container, all the objects within it become available through the CDN. Similarly, once a container is CDN-enabled, any objects added to it in the storage service become CDN-enabled. Once you CDN-enable a container, its publically-available URL can be found under the header `X-CDN-URI`, and its objects may be accessed at `X-CDN-URI/objectName`. By knowing this pattern, you may pre-generate the URL for an object before it is added to the container.

When you enable a container in the CDN service, you automatically generate URLs for SSL and streaming usage. They are listed under the `X-CDN-SSL-URI` and `X-CDN-STREAMING-URI` headers, respectively.

5.2.1. CDN-Enable a Container

Before a container can be CDN-enabled, it must exist in the storage system. To CDN-enable the container, you will **PUT** it to the publicURL created under Cloud Files during Authentication. (In the examples below, this URL is referred to as `cdn.clouddrive.com`.) When a container is CDN-enabled, any objects stored in that container are publicly accessible over a CDN by combining the container's CDN URL with the object name (`X-CDN-URI/objectName`).

Any CDN-accessed objects will be cached in the CDN for specified amount of time called the TTL, or Time to Live. The default TTL value is 259200 seconds, or 72 hours. Each time the object is accessed after a TTL expiration, the CDN refetches and caches the object for the TTL duration.

You may specify the TTL for an object by including the HTTP header `X-TTL: integer_seconds`. Setting the TTL is the same as setting the HTTP `Expires` and `Cache-Control` headers for the cached object. The minimum TTL is 15 minutes (900 seconds) and the maximum is 50 years (range of 900 to 1577836800 seconds). Setting a TTL for a long time, however, does not guarantee that the content will stay populated on CDN edge servers for the entire five-year period. The most popular objects stay cached based on the edge location's 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: 259200
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: Wed, 07 Sep 2011 19:04:30 GMT
Server: Apache/2.2.14 (Ubuntu)
Content-Type: text/plain
X-CDN-URI: http://c10171.r71.cf0.rackcdn.com
X-CDN-SSL-URI: http://c10171.ssl.cf0.rackcdn.com
X-CDN-STREAMING-URI: http://c10171.r71.stream.cf0.rackcdn.com
```

In order to remove the container from the CDN, change the `X-Log-Retention` flag to `False`, as in the request below. Please note, however, that objects remain public until their TTL expires.

Example 5.11. Container CDN-Disable 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-CDN-Enabled: False
```

5.2.2. List a CDN-Enabled Container's 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 URL, TTL, enabled status, and log retention status are returned in the response headers. Its CDN URL 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 5.12. 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 5.13. 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-CDN-STREAMING-URI: http://c10171.stream.cf0.rackcdn.com
X-TTL: 86400
X-Log-Retention: True
```

5.2.3. Purge CDN-Enabled Containers

Rackspace can perform operations against a CDN-enabled container. You can either wait for the TTL to expire or you can request the purge of an entire container. Once you have made the request via a support ticket, the system purges the object from the CDN, and sends an email to the address (or multiple addresses) that you indicate through the ticket. The email address notification is optional.

5.2.4. Update CDN-Enabled Container Metadata

You may use the **POST** request against a CDN-enabled container to adjust CDN attributes. You may not, however, set the arbitrary header metadata that is allowed under the storage service. The only metadata that may be changed on the CDN is `X-Log-Retention`, `X-CDN-enabled`, and `X-TTL`.

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 5.14. 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 204 (No Content) indicates success; 404 (Not Found) is returned if the requested container was not found. The CDN URL, the CDN SSL URL, and the CDN Streaming URL are returned in the HTTP headers, `X-CDN-URL`, `X-CDN-SSL-URL`, and `X-CDN-STREAMING-URL`.

Example 5.15. 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-URL: http://c10171.r71.cf0.rackcdn.com
X-CDN-SSL-URL: https://c10171.ssl.cf0.rackcdn.com
X-CDN-STREAMING-URL: http://c10171.r71.stream.cf0.rackcdn.com
```

5.2.5. CDN-Enabled Containers Served via SSL

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

Example 5.16. 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 204 (No Content) indicates success; 404 (Not Found) is returned if the requested container was not found. The CDN SSL URL is returned in the HTTP header, `X-CDN-SSL-URL`, along with the CDN streaming URL, `X-CDN-STREAMING-URL`.

Example 5.17. 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-URL: http://c10171.r71.cf0.rackcdn.com
X-CDN-SSL-URL: https://c10171.ssl.cf0.rackcdn.com
X-CDN-STREAMING-URL: http://c10171.r71.stream.cf0.rackcdn.com
X-CDN-Enabled: True
X-TTL: 259000
X-Log-Retention: False
Connection: close
Content-Type: text/plain; charset=UTF-8
```

5.2.6. Streaming CDN-Enabled Containers

HEAD operations against a CDN-enabled container can also return a streaming URL. Like SSL, streaming is always enabled, and you use the streaming URL for links to content that you want to stream, such as video.

Example 5.18. CDN-Enabled Container Metadata Requests (Streaming Enabled)

```
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 204 (No Content) indicates success; 404 (Not Found) is returned if the requested container was not found. The CDN SSL URL is returned in the HTTP header, `X-CDN-SSL-URL`, along with the CDN streaming URL, `X-CDN-STREAMING-URL`.

Example 5.19. CDN-Enabled Container Metadata (Streaming Enabled)

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

5.3. CDN Object Services

This section documents the ReST operations against the CDN management service that can be performed on objects. Any objects stored in a CDN-enabled container are publicly accessible over a CDN by combining the container's CDN URL 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).

5.3.1. Purge CDN-Enabled Objects

You can perform **DELETE** operations against a CDN-enabled object when you find it absolutely necessary to remove the object from public access and you cannot wait for the TTL to expire. The situation for object purges should be limited to times when it could have serious personal, business, or security consequences if it remained in the CDN. For example, suppose someone published your company's quarterly earnings too early. Perhaps the timing of the publish was incorrect, and you need to remove it from public view. You can manually purge CDN-enabled objects 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: the method for purging individual objects is available through the API, and the method for purging entire containers is available by filing a support ticket.

Example 5.20. 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, user2@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 object was never CDN-enabled, and it returns a 403 (Forbidden) status code if an authorization problem occurs. The CDN URL is returned in the HTTP header, `X-CDN-URI`. Purging a large number of objects may take a long time, so please be patient while waiting for a response.

Example 5.21. Purge CDN-Enabled 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
```

6. Public Access to Your Cloud Files Account

This section describes ways you can allow others to post or retrieve objects from your Cloud Files account. With the methods described here, users do not need your password or login information in order to have access to your account.

6.1. TempURL

The Temporary URL feature (TempURL) allows you to create limited-time Internet addresses which allow you to grant limited access to your Cloud Files account. Using TempURL, you may allow others to retrieve or put objects in your Cloud Files account for as long or as short a time as you wish. Access to the TempURL is independent of whether or not your account is CDN-enabled. Even if you don't CDN-enable a directory, you can still grant temporary public access through a TempURL.

This feature is useful if you want to allow a limited audience to download a file from your Cloud Files account or website. You can give out the TempURL and know that after a specified time, no one will be able to access your object through the address. Or, if you want to allow your audience to upload objects into your Cloud Files account, you can give out a TempURL. After the specified time expires, no one will be able to upload to the address.

Additionally, you need not worry about time running out when someone downloads a large object. If the time expires while a file is being retrieved, the download will continue until it is finished. Only the link will expire.

When you create a TempURL, Cloud Files validates a GET- or PUT-accessible URL which is time-limited.

6.1.1. Set Account Metadata Key

To create a Temporary URL, you must first set the metadata header `X-Account-Meta-Temp-URL-Key` on your Cloud Files account to a key that only you know. This key can be any arbitrary sequence as it is for encoding your account.

Once the key is set, you may not change it while you still want others to be able to access your TempURL. If you change it, the TempURL becomes invalid (within 60 seconds, which is the cache time for a key) and others will not be allowed to access it.

Example 6.1. Set Account Metadata Key for Public Access

```
POST /<api version>/<account> HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
X-Account-Meta-Temp-Url-Key: <your key>
```

Any 2xx response indicates success.

6.1.2. Create the TempURL

After the metadata is set, you must create an HMAC-SHA1 (RFC 2104) signature. When you generate the TempURL, you determine which method of access you will grant users, **GET** or **PUT**. You also determine the path to the object or container you are granting access to. If you allow **PUT** access to your Cloud Files account, you must indicate the object prefix that will be used for each file uploaded. Lastly, set the time in UNIX epoch notation, which is when the TempURL expires.

In the Python example below, a URL is generated for an object (my_cat.jpg), which will be available for 60 seconds.

Example 6.2. Create TempURL

```
import hmac
from hashlib import sha1
from time import time
method = 'GET'
expires = int(time() + 60)
path = '/v1/AUTH_account/container/my_cat.jpg'
key = 'mykey'
hmac_body = '%s\n%s\n%s' % (method, expires, path)
sig = hmac.new(key, hmac_body, sha1).hexdigest()
```

Be certain to use the full path, from the /v1/ onward.

In this example, the signature becomes da39a3ee5e6b4b0d3255bfef95601890afd80709 and expires is 1323479485. On your website, you would provide a link to the below URL:

```
https://storage.cloudrive.com/v1/AUTH_account/container/my_cat.jpg?
temp_url_sig=da39a3ee5e6b4b0d3255bfef95601890afd80709&
temp_url_expires=1323479485
```

If you do not provide users with the exact TempURL, they will get 401 (Unauthorized) status errors. **HEAD** queries are allowed if **GET** or **PUT** are allowed.

6.2. FormPost

FormPost lets you offer your website audience a way to upload objects to your Cloud Files account through a web form. FormPost works by translating a browser form request into a object PUT in Cloud Files. Once you enable FormPost on your account, you need only create the form in your website using the guidelines below.

As with all objects in Cloud Files, the object file size limit is 5 gb. If your users try to upload an object larger than 5 gb, they will get a file size error.

6.2.1. Set Account Metadata Key

To allow FormPost actions on your Cloud Files account, you must first set the metadata header `X-Account-Meta-Temp-URL-Key` on your Cloud Files account to a key that only you know. This key can be any arbitrary sequence as it is for encoding your account.

Once the key is set, you may not change it while you still want others to access your account. If you change it, the actions from a FormPost become invalid (within 60 seconds, which is the cache time for a key).

Example 6.3. Set Account Metadata Key for Public Access

```
POST /<api version>/<account> HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
X-Account-Meta-Temp-Url-Key: <your key>
```

Any 2xx response indicates success.

6.2.2. Create the Form

To communicate between your website and your Cloud Files account, create a form using the following format in your website:

Example 6.4. Layout of Web Form

```
<form action="<CF-url>" method="POST" enctype="multipart/form-data">
  <input type="hidden" name="redirect" value="<redirect-url>" />
  <input type="hidden" name="max_file_size" value="<bytes>" />
  <input type="hidden" name="max_file_count" value="<count>" />
  <input type="hidden" name="expires" value="<unix-timestamp>" />
  <input type="hidden" name="signature" value="<hmac>" />
  <input type="file" name="file1" /><br />
  <input type="submit" />
</form>
```

Required: Form `action` is the Cloud Files URL (CF-url) to the destination where files will be uploaded. For instance, `https://storage.clouddrive.com/v1/CF_xer7_34/container`. The name of each uploaded object will have the `<CF-url>` appended to the front of it. Your users upload directly to root of the container.

You may also include a prefix to separate uploads, such as assigning each user a certain prefix: `https://storage.clouddrive.com/v1/CF_xer7_34/container/user_prefix`.

Required: The form method must be "POST" and the `enctype` must be set as "multipart/form-data".

Optional: The `redirect` attribute is the URL of the page that displays on your website after the form processes. The URL will have status and message query parameters added to it, indicating the HTTP status code for the upload (2xx is success) and a possible message for further information if there is an error, such as "max_file_size exceeded".

Required: The `max_file_size` attribute must be included and indicates the largest single file upload allowed, in bytes. Although you may limit the size of the individual file uploaded to your Cloud Files account here, the `max_file_size` may not exceed 5 gb.

Required: The `max_file_count` attribute indicates the maximum number of files that can be uploaded with the form.

Required: The `expires` attribute is the Unix timestamp when the form is invalidated. This gives your website users a limited time to have the form open. Time must be in Unix epoch format.

Required: The `signature` attribute is the HMAC-SHA1 signature of the form. Here is sample code for computing the signature in Python:

Example 6.5. Generate Signature for Form Post

```
import hmac
from hashlib import sha1
from time import time
path = '/v1/account/container/object_prefix'
redirect = 'https://myserver.com/some-page'
max_file_size = 104857600
max_file_count = 10
expires = int(time() + 600)
key = 'mykey'
hmac_body = '%s\n%s\n%s\n%s\n%s' % (path, redirect,
    max_file_size, max_file_count, expires)
signature = hmac.new(key, hmac_body, sha1).hexdigest()
```

Be certain to use the full path in your Cloud Files account, from the `/v1/` onward.

Key is the value of the `X-Account-Meta-Temp-URL-Key` header set for the account.

The `max_file_count` used in generating the signature must be the same as that in the web form.

Required: The `type="file"` field defines the form file field. At least one entry is required to allow your users to select and upload a file, but additional fields may be added for multiple files. The number of entries should not, however, exceed the `max_file_count`. Each `type="file"` field must have a different name.

Note that the `type="file"` field(s) must be at the end of the form code in order for Cloud Files to process the uploads properly.

7. Examples and Troubleshooting

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

Remember that object and container names must be URL-encoded and UTF-8 encoded. Refer to [a full list of the requirements](#) if you see problems. Object names 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.

Also, tokens expire after 24 hours. Be sure you request a new token programmatically only when one you have is expired.

7.1. 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 (**GET**, **HEAD**, etc.)
- D Dump HTTP response headers to stdout.
- H HEADER Specify an HTTP header in the request.

7.2. Authentication with cURL

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. Note that 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 where the account was created. This example uses the US-based URL <https://lon.auth.api.rackspacecloud.com/v1.0>.

Example 7.1. cURL Authenticate

```
curl -D - \  
-H "X-Auth-Key: a86850deb2742ec3cb41518e26aa2d89" \  
-H "X-Auth-User: jdoe" \  
https://lon.auth.api.rackspacecloud.com/v1.0
```

```
https://auth.api.rackspacecloud.com/v1.0
```

```
HTTP/1.1 204 No Content
Server: Apache/2.2.13 (Red Hat)
Date: Wed, 07 Sep 2011 18:19:26 GMT
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 **GET**, **HEAD**, **DELETE**, **POST** and **PUT** requests on the storage and CDN services.

While an authentication token lasts, you can continue to perform requests. However, once a token expires, it will return an HTTP error code 401 (Unauthorized). Given that a token is good for 24 hours, even long-running jobs do not need to re-authenticate on every request. You won't need to request another `X-Auth-Token` again until the existing one expires. At that point, you must obtain another authorization token. As a best practice example, here is some pseudo-code for re-authenticating. The best scalable process flow would be:

1. Begin requests by going to `auth.api.rackspace.com` for an `X-Auth-Token`.
2. Send request `X-Storage-URL` using the `X-Auth-Token` obtained in Step 1.
3. Repeat step 2 using the same `X-Auth-Token` retrieved in Step 1 until either the job finishes or you get a result code of 401 (Unauthorized).
 - If the job finishes, you can allow the token to expire with no further action.
 - If result code is 401 then send a request to `auth.api.rackspacecloud.com` for a new `X-Auth-Token`.

A Python-based example of how to check for errors and re-authenticate upon receiving an error can be found in the OpenStack Swift project in `client.py`, which is freely available.

7.3. Determining Storage Usage with cURL

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 7.2. 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
X-Account-Object-Count: 4943
X-Account-Bytes-Used: 25603957646
X-Account-Container-Count: 151
Accept-Ranges: bytes
Content-Length: 0
X-Trans-Id: tx15d1b08e3c1540at8cceda42acc723e4
Date: Wed, 07 Sep 2011 18:48:15 GMT
```

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.

7.4. Creating a Storage Container with cURL

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 7.3. 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
Content-Length: 18
Content-Type: text/html; charset=UTF-8
X-Trans-Id: txs56dc5b74f91419480ba485348057bfd
Date: Wed, 07 Sep 2011 18:52:30 GMT
```

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

7.5. Uploading a Storage Object with cURL

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 7.4. cURL Upload Storage Object

```
curl -X PUT -T screenies/wowl.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
```

7.6. CDN-Enabling the Container with cURL

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, not the authorization system.

Example 7.5. 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/1.1 201 Created
Date: Wed, 07 Sep 2011 19:00:44 GMT
Server: Apache/2.2.14 (Ubuntu)
X-CDN-URL: http://c1935.r35.stg2.rackcdn.com
X-CDN-SSL-URL: https://c1935.ssl.stg2.rackcdn.com
X-CDN-STREAMING-URL: http://c1935.r35.stream.stg2.rackcdn.com
Content-Length: 0
Content-Type: text/plain
```

When the container is CDN-enabled, the service returns its public URL in the `X-CDN-URL` header of the response, plus the SSL URL in the `X-CDN-SSL-URL` header of the response. Now you can combine this URL with the object name to access the file via the CDN, or use the `https://` URL 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 convention is only valid on UNIX-like systems).

Example 7.6. 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
```

7.7. 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.